Package ‘copBasic’

June 19, 2023

Title General Bivariate Copula Theory and Many Utility Functions
Version 2.2.0
Date 2023-06-19
Author William Asquith

Description Extensive functions for bivariate copula (bicopula) computations and related operations for bicopula theory. The lower, upper, product, and select other bicopula are implemented along with operations including the diagonal, survival copula, dual of a copula, co-copula, and numerical bicopula density. Level sets, horizontal and vertical sections are supported. Numerical derivatives and inverses of a bicopula are provided through which simulation is implemented. Bicopula composition, convex combination, asymmetry extension, and products also are provided. Support extends to the Kendall Function as well as the Lmoments thereof. Kendall Tau, Spearman Rho and Footrule, Gini Gamma, Blomqvist Beta, Hoeffding Phi, Schweizer-Wolff Sigma, tail dependency, tail order, skewness, and bivariate Lmoments are implemented, and positive/negative quadrant dependency, left (right) increasing (decreasing) are available. Other features include Kullback-Leibler Divergence, Vuong Procedure, spectral measure, and Lcomoments for inference, maximum likelihood, and AIC, BIC, and RMSE for goodness-of-fit.

Maintainer William Asquith <william.asquith@ttu.edu>

Repository CRAN

Depends R (>= 4.0.0)
Imports lmomco, randtoolbox
Suggests copula
License GPL-2

Date/Publication 2023-06-19 16:50:02 UTC

R topics documented:
copBasic-package .......................................................... 4
aicCOP ................................................................. 13
AMHcop ................................................................. 15
asCOP ................................................................. 18
bicCOP ................................................................. 20
R topics documented:

bilnoms .......................................................... 22
blomatrixCOP .................................................... 27
blomCOP .......................................................... 31
blomCOPss ......................................................... 34
breveCOP ........................................................ 37
CLcop .............................................................. 39
cOCOP ............................................................... 40
composite1COP ................................................. 42
composite2COP .................................................. 44
composite3COP .................................................. 46
convex2COP ....................................................... 48
convexCOP ......................................................... 50
COP ................................................................. 52
copBasic.fitpara .................................................. 56
COPinv .............................................................. 59
COPinv2 ............................................................. 61
densityCOP ........................................................ 62
densityCOPplot ................................................... 64
derCOP ............................................................... 67
derCOP2 .............................................................. 69
derCOPinv ........................................................... 71
derCOPinv2 .......................................................... 75
diagCOP ............................................................. 76
diagCOPatf .......................................................... 78
duCOP ................................................................. 80
EMPIRcop .......................................................... 83
EMPIRcopdf ......................................................... 88
EMPIRgrid .......................................................... 89
EMPIRgridder ..................................................... 93
EMPIRgridder2 .................................................... 94
EMPIRgridderinv ................................................. 95
EMPIRgridderinv2 ............................................... 98
EMPIRmed.regress ............................................... 99
EMPIRmed.regress2 ............................................ 100
EMPIRqua.regress .............................................. 100
EMPIRqua.regress2 ............................................ 106
EMPIRsim ........................................................ 107
EMPIRsimv ......................................................... 109
FGMcop ........................................................... 111
footCOP ............................................................ 113
FRECHETcop ..................................................... 115
gEVcop ........................................................... 118
GHcop ............................................................. 119
giniCOP ............................................................ 126
GLcop ............................................................... 129
glueCOP ........................................................... 131
gridCOP ............................................................ 133
hoefCOP ............................................................ 135
<table>
<thead>
<tr>
<th>Library</th>
<th>Topics Documented</th>
</tr>
</thead>
<tbody>
<tr>
<td>HRcop</td>
<td>139</td>
</tr>
<tr>
<td>isCOP.LTD</td>
<td>140</td>
</tr>
<tr>
<td>isCOP.permssym</td>
<td>141</td>
</tr>
<tr>
<td>isCOP.PQD</td>
<td>142</td>
</tr>
<tr>
<td>isCOP.radsym</td>
<td>145</td>
</tr>
<tr>
<td>isCOP.RTI</td>
<td>147</td>
</tr>
<tr>
<td>isfuncCOP</td>
<td>149</td>
</tr>
<tr>
<td>JOcopB5</td>
<td>150</td>
</tr>
<tr>
<td>joeskewCOP</td>
<td>152</td>
</tr>
<tr>
<td>joint.curvesCOP</td>
<td>157</td>
</tr>
<tr>
<td>joint.curvesCOP2</td>
<td>160</td>
</tr>
<tr>
<td>jointCOP</td>
<td>162</td>
</tr>
<tr>
<td>kfuncCOP</td>
<td>167</td>
</tr>
<tr>
<td>kfuncCOPinv</td>
<td>174</td>
</tr>
<tr>
<td>kfuncCOPl</td>
<td>176</td>
</tr>
<tr>
<td>kullCOP</td>
<td>179</td>
</tr>
<tr>
<td>lcomCOP</td>
<td>184</td>
</tr>
<tr>
<td>lcomCOPpv</td>
<td>187</td>
</tr>
<tr>
<td>lcomoms2.ABcop2parameter</td>
<td>190</td>
</tr>
<tr>
<td>level.curvesCOP</td>
<td>196</td>
</tr>
<tr>
<td>level.curvesCOP2</td>
<td>198</td>
</tr>
<tr>
<td>level.setCOP</td>
<td>199</td>
</tr>
<tr>
<td>level.setCOP2</td>
<td>200</td>
</tr>
<tr>
<td>M</td>
<td>201</td>
</tr>
<tr>
<td>med.regressCOP</td>
<td>202</td>
</tr>
<tr>
<td>med.regressCOP2</td>
<td>205</td>
</tr>
<tr>
<td>mleCOP</td>
<td>206</td>
</tr>
<tr>
<td>N4212cop</td>
<td>211</td>
</tr>
<tr>
<td>P</td>
<td>212</td>
</tr>
<tr>
<td>PARETOcop</td>
<td>214</td>
</tr>
<tr>
<td>PLACKETTcop</td>
<td>215</td>
</tr>
<tr>
<td>PLACKETTpar</td>
<td>217</td>
</tr>
<tr>
<td>PLACKETTsim</td>
<td>219</td>
</tr>
<tr>
<td>prod2COP</td>
<td>221</td>
</tr>
<tr>
<td>psepolar</td>
<td>223</td>
</tr>
<tr>
<td>PSP</td>
<td>225</td>
</tr>
<tr>
<td>qua.regressCOP</td>
<td>226</td>
</tr>
<tr>
<td>qua.regressCOP.draw</td>
<td>228</td>
</tr>
<tr>
<td>qua.regressCOP2</td>
<td>230</td>
</tr>
<tr>
<td>ReineckeWell266</td>
<td>231</td>
</tr>
<tr>
<td>ReineckeWells</td>
<td>232</td>
</tr>
<tr>
<td>RFcop</td>
<td>233</td>
</tr>
<tr>
<td>rhobevCOP</td>
<td>234</td>
</tr>
<tr>
<td>rhoCOP</td>
<td>236</td>
</tr>
<tr>
<td>rmseCOP</td>
<td>239</td>
</tr>
<tr>
<td>sectionCOP</td>
<td>240</td>
</tr>
<tr>
<td>semicorCOP</td>
<td>243</td>
</tr>
</tbody>
</table>
copBasic-package

Description

The copBasic package is oriented around bivariate copula theory and mathematical operations closely follow the standard texts of Nelsen (2006) and Joe (2014) as well as select other references. Another recommended text is Salvadori et al. (2007) and is cited herein, but about half of that excellent book concerns univariate applications. The primal objective of copBasic is to provide a basic application programming interface (API) to numerous results shown by authoritative texts on copulas. It is intended that the package will help other copula students in self study, potential course work, and applied circumstances.

Notes on copulas that are supported. The author has focused on pedagogical aspects of copulas, and this package is a diary of sorts. Originally, the author did not implement many copulas in the copBasic in order to deliberately avoid redundancy to that support such as it exists on the R CRAN. Though as time has progressed, other copulas have been added occasionally based on needs of the user community, need to show some specific concept in the general theory, or test algorithms. For example, the Clayton copula (ClCOP) is a late arriving addition to the copBasic package (c.2017), which was added to assist a specific user.

Helpful Navigation of Copulas Implemented in the copBasic Package Some entry points to the copulas implemented are listed in the Table of Copulas:

<table>
<thead>
<tr>
<th>Name</th>
<th>Symbol</th>
<th>Function</th>
<th>Concept</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lower-bounds copula</td>
<td>$W(u,v)$</td>
<td>$W$</td>
<td>copula</td>
</tr>
<tr>
<td>Independence copula</td>
<td>$\Pi(u,v)$</td>
<td>$P$</td>
<td>copula</td>
</tr>
</tbody>
</table>

Index
The language and vocabulary of copulas is formidable. The author (Asquith) has often emphasized “vocabulary” words in italics, which is used extensively and usually near the opening of function-by-function documentation to identify vocabulary words, such as survival copula (see surCOP). This syntax tries to mimic and accentuate the word usage in Nelsen (2006) and Joe (2014).

The italics then are used to draw connections between concepts. In conjunction with the summary of functions in copBasic-package, the extensive cross referencing to functions and expansive key-word indexing should be beneficial. The author had no experience with copulas prior to a chance happening upon Nelsen (2006) in c.2008. The copBasic package is a personal tour de force in self-guided learning. Hopefully, this package and user’s manual will be helpful to others.

A few comments on notation herein are needed. A bold math typeface is used to represent a copula such as \( \Pi \) (see P) for the independence copula. The syntax \( \mathbb{R} \times \mathbb{R} \equiv \mathbb{R}^2 \) denotes the orthogonal domain of two real numbers, and \([0, 1] \times [0, 1] \equiv \mathcal{I} \times \mathcal{I} \equiv \mathcal{I}^2 \) denotes the orthogonal domain on the unit square of probabilities. Limits of integration \([0, 1]\) or \([0, 1]^2\) involving copulas are thus shown as \( \mathcal{I} \) and \( \mathcal{I}^2 \), respectively.

The random variables \( X \) and \( Y \) respectively denote the horizontal and vertical directions in \( \mathbb{R}^2 \). Their probabilistic counterparts are uniformly distributed random variables on \([0, 1]\), are respectively denoted as \( U \) and \( V \), and necessarily also are the respective directions in \( \mathcal{I}^2 \) (\( U \) denotes the horizontal, \( V \) denotes the vertical). Often realizations of these random variables are respectively \( x \) and \( y \) for \( X \) and \( Y \) and \( u \) and \( v \) for \( U \) and \( V \).

There is an obvious difference between nonexceedance probability \( F \) and its complement, which is exceedance probability defined as \( 1 - F \). Both \( u \) and \( v \) herein are in nonexceedance probability. Arguments to many functions herein are \( u = u \) and \( v = v \) and are almost exclusively nonexceedance but there are instances for which the probability arguments are \( u = 1 - u = u' \) and \( v = 1 - v = v' \).

### Helpful Navigation of the copBasic Package

Some other entry points into the package are listed in the following table:

<table>
<thead>
<tr>
<th>Name</th>
<th>Symbol</th>
<th>Function</th>
<th>Concept</th>
</tr>
</thead>
<tbody>
<tr>
<td>Upper-bounds copula</td>
<td>( M(u, v) )</td>
<td>( M )</td>
<td>copula</td>
</tr>
<tr>
<td>Fréchet Family copula</td>
<td>( FF(u, v) )</td>
<td>( FRECHETcop )</td>
<td>copula</td>
</tr>
<tr>
<td>Ali–Mikhail–Haq copula</td>
<td>( AMH(u, v) )</td>
<td>( AMHcop )</td>
<td>copula</td>
</tr>
<tr>
<td>Clayton copula</td>
<td>( CL(u, v) )</td>
<td>( CLcop )</td>
<td>copula</td>
</tr>
<tr>
<td>Farlie–Gumbel–Morgenstern (generalized)</td>
<td>( FGM(u, v) )</td>
<td>( FGMcop )</td>
<td>copula</td>
</tr>
<tr>
<td>Gumbel–Hougaard copula</td>
<td>( GH(u, v) )</td>
<td>( GHcop )</td>
<td>copula</td>
</tr>
<tr>
<td>Hüsler–Reiss copula</td>
<td>( HR(u, v) )</td>
<td>( HRcop )</td>
<td>copula</td>
</tr>
<tr>
<td>Joe B5 copula</td>
<td>( B5(u, v) )</td>
<td>( J0copB5 )</td>
<td>copula</td>
</tr>
<tr>
<td>Nelsen eq.4-2-12 copula</td>
<td>( N4212cop(u, v) )</td>
<td>( N4212cop )</td>
<td>copula</td>
</tr>
<tr>
<td>Pareto copula</td>
<td>( PA(u, v) )</td>
<td>( PLcop )</td>
<td>copula</td>
</tr>
<tr>
<td>Plackett copula</td>
<td>( PL(u, v) )</td>
<td>( PLcop )</td>
<td>copula</td>
</tr>
<tr>
<td>PSP copula</td>
<td>( PSP(u, v) )</td>
<td>( PSP )</td>
<td>copula</td>
</tr>
<tr>
<td>Raftery copula</td>
<td>( RF(u, v) )</td>
<td>( RFcop )</td>
<td>copula</td>
</tr>
<tr>
<td>g-EV copula (Gaussian extreme value)</td>
<td>( gEV(u, v) )</td>
<td>( gEVcop )</td>
<td>copula</td>
</tr>
<tr>
<td>t-EV copula (t-distribution extreme value)</td>
<td>( tEV(u, v) )</td>
<td>( tEVcop )</td>
<td>copula</td>
</tr>
</tbody>
</table>
Several of the functions listed above are measures of “bivariate association.” Two of the measures (Kendall Tau, tauCOP; Spearman Rho, rhoCOP) are widely known. R provides native support for their sample estimation of course, but each function can be used to call the cor() function in R.
for parallelism to the other measures of this package. The other measures (Blomqvist Beta, Gini Gamma, Hoefding Phi, Schweizer–Wolff Sigma, Spearman Footrule) support sample estimation by specially formed calls to their respective functions: `bloomCOP`, `giniCOP`, `hoefCOP`, `wolfCOP`, and `footCOP`. Gini Gamma (`giniCOP`) documentation (also `joeskewCOP`) shows extensive use of theoretical and sample computations for these and other functions.

Concerning goodness-of-fit and although not quite the same as copula properties (such as “correlation”) per se as the coefficients aforementioned in the prior paragraph, three goodness-of-fit metrics of a copula compared to the empirical copula, which are all based the mean square error (MSE), are `aicCOP`, `bicCOP`, and `rmseCOP`. This triad of functions is useful for making decisions on whether a copula is more favorable than another to a given dataset. However, because they are genetically related by using MSE and if these are used for copula fitting by minimization, the fits will be identical. A statement of “not quite the same” is made because the previously described copula properties are generally defined as types of deviations from other copulas (such as $P$). Another goodness-of-fit statistic is `statTn`, which is based on magnitude summation of fitted copula difference from the empirical copula. These four (`aicCOP`, `bicCOP`, `rmseCOP`, and `statTn`) collectively are relative simple and readily understood measures. These bulk sample statistics are useful, but generally thought to not capture the nuances of tail behavior (`semicorCOP` and `taildepCOP` might be useful).

**Bivariate skewness** measures are supported in the functions `joeskewCOP` (nuskewCOP and nustarCOP) and `uvlmoms` (uvskew). Extensive discussion and example computations of bivariate skewness are provided in the `joeskewCOP` documentation. Lastly, so-called bivariate $L$-moments and bivariate $L$-comoments of a copula are directly computable in `bilmoms` (though that function using Monte Carlo integration is deprecated) and `lcomCOP` (direct numerical integration). The `lcomCOP` function is the theoretical counterpart to the sample L-comoments provided in the `inmomco` package.

Bivariate random simulation by several functions is identified in the previous table. The `copBasic` package explicitly uses only conditional simulation also known as the conditional distribution method for random variate generation following Nelsen (2006, pp. 40–41) (see also `simCOPmicro`, `simCOP`). The numerical derivatives (derCOP and derCOP2) and their inversions (derCOPinv and derCOPinv2) represent the foundation of the conditional simulation. There are other methods in the literature and available in other R packages, and a comparison of some methods is made in the Examples section of the Gumbel–Hougaard copula (GHcop).

Several functions in `copBasic` make the distinction between $V$ with respect to $(wrt) U$ and $U$ wrt $V$, and a guide for the nomenclature involving wrt distinctions is listed in the following table:

<table>
<thead>
<tr>
<th>Name</th>
<th>Symbol</th>
<th>Function</th>
<th>Concept</th>
</tr>
</thead>
<tbody>
<tr>
<td>Copula inversion</td>
<td>$V$ wrt $U$</td>
<td><code>COPinv</code></td>
<td>copula operator</td>
</tr>
<tr>
<td>Copula inversion</td>
<td>$U$ wrt $V$</td>
<td><code>COPinv2</code></td>
<td>copula operator</td>
</tr>
<tr>
<td>Copula derivative</td>
<td>$\delta C/\delta u$</td>
<td><code>derCOP</code></td>
<td>copula operator</td>
</tr>
<tr>
<td>Copula derivative</td>
<td>$\delta C/\delta v$</td>
<td><code>derCOP2</code></td>
<td>copula operator</td>
</tr>
<tr>
<td>Copula derivative inversion</td>
<td>$V$ wrt $U$</td>
<td><code>derCOPinv</code></td>
<td>copula operator</td>
</tr>
<tr>
<td>Copula derivative inversion</td>
<td>$U$ wrt $V$</td>
<td><code>derCOPinv2</code></td>
<td>copula operator</td>
</tr>
<tr>
<td>Joint curves</td>
<td>$t \mapsto C(u = U, v)$</td>
<td><code>joint.curvesCOP</code></td>
<td>copula theory</td>
</tr>
<tr>
<td>Joint curves</td>
<td>$t \mapsto C(u, v = V)$</td>
<td><code>joint.curvesCOP2</code></td>
<td>copula theory</td>
</tr>
<tr>
<td>Level curves</td>
<td>$t \mapsto C(u = U, v)$</td>
<td><code>level.curvesCOP</code></td>
<td>copula theory</td>
</tr>
<tr>
<td>Level curves</td>
<td>$t \mapsto C(u, v = V)$</td>
<td><code>level.curvesCOP2</code></td>
<td>copula theory</td>
</tr>
<tr>
<td>Level set</td>
<td>$V$ wrt $U$</td>
<td><code>level.setCOP</code></td>
<td>copula theory</td>
</tr>
<tr>
<td>Level set</td>
<td>$U$ wrt $V$</td>
<td><code>level.setCOP2</code></td>
<td>copula theory</td>
</tr>
<tr>
<td>Median regression</td>
<td>$V$ wrt $U$</td>
<td><code>med.regressCOP</code></td>
<td>copula theory</td>
</tr>
<tr>
<td>Median regression</td>
<td>$U$ wrt $V$</td>
<td><code>med.regressCOP2</code></td>
<td>copula theory</td>
</tr>
</tbody>
</table>
The previous two tables do not include all of the myriad of special functions to support similar operations on empirical copulas. All empirical copula operators and utilities are prepended with `EMPIR` in the function name. An additional note concerning package nomenclature is that an appended “2” to a function name indicates \( U \) \( \text{wrt} \) \( V \) (e.g. `EMPIRgridderv2` for an inversion of the partial derivatives \( \delta C/\delta v \) across the grid of the empirical copula).

Some additional functions to compute often salient features or characteristics of copulas or bivariate data, including functions for bivariate inference or goodness-of-fit, are listed in the following table:

<table>
<thead>
<tr>
<th>Name</th>
<th>Symbol ( \text{wrt} ) Function</th>
<th>Concept</th>
</tr>
</thead>
<tbody>
<tr>
<td>Left-tail decreasing ( V )</td>
<td><code>isCOP.LTD</code></td>
<td>bivariate association</td>
</tr>
<tr>
<td>Left-tail decreasing ( U )</td>
<td><code>isCOP.LTD</code></td>
<td>bivariate association</td>
</tr>
<tr>
<td>Right-tail increasing ( V )</td>
<td><code>isCOP.RTI</code></td>
<td>bivariate association</td>
</tr>
<tr>
<td>Right-tail increasing ( U )</td>
<td><code>isCOP.RTI</code></td>
<td>bivariate association</td>
</tr>
<tr>
<td>Pseudo-polar representation</td>
<td><code>psepol</code></td>
<td>extremal dependency</td>
</tr>
<tr>
<td>Tail concentration function</td>
<td><code>tailconCOP</code></td>
<td>bivariate tail association</td>
</tr>
<tr>
<td>Tail (lower) dependency ( \lambda_L )</td>
<td><code>taildepCOP</code></td>
<td>bivariate tail association</td>
</tr>
<tr>
<td>Tail (upper) dependency ( \lambda_U )</td>
<td><code>taildepCOP</code></td>
<td>bivariate tail association</td>
</tr>
<tr>
<td>Tail (lower) order ( \kappa_L )</td>
<td><code>tailordCOP</code></td>
<td>bivariate tail association</td>
</tr>
<tr>
<td>Tail (upper) order ( \kappa_U )</td>
<td><code>tailordCOP</code></td>
<td>bivariate tail association</td>
</tr>
<tr>
<td>Neg’ly quadrant dependency</td>
<td><code>isCOP.PQD</code></td>
<td>bivariate association</td>
</tr>
<tr>
<td>Pos’ly quadrant dependency</td>
<td><code>isCOP.PQD</code></td>
<td>bivariate association</td>
</tr>
<tr>
<td>Permutation symmetry</td>
<td><code>isCOP.perm</code></td>
<td>copula symmetry</td>
</tr>
<tr>
<td>Radial symmetry</td>
<td><code>isCOP.rad</code></td>
<td>copula symmetry</td>
</tr>
<tr>
<td>Skewness (Joe, 2014)</td>
<td><code>uvskew</code></td>
<td>bivariate skewness</td>
</tr>
<tr>
<td>Kullback–Leibler Divergence</td>
<td><code>kullCOP</code></td>
<td>bivariate inference</td>
</tr>
<tr>
<td>KL sample size</td>
<td><code>kullCOP</code></td>
<td>bivariate inference</td>
</tr>
<tr>
<td>The Vuong Procedure</td>
<td><code>vuongCOP</code></td>
<td>bivariate inference</td>
</tr>
<tr>
<td>Spectral measure</td>
<td><code>spectralmeas</code></td>
<td>extremal dependency inference</td>
</tr>
<tr>
<td>Stable tail dependence</td>
<td><code>sttabtaildepf</code></td>
<td>extremal dependency inference</td>
</tr>
<tr>
<td>L-comoments (samp. distr.)</td>
<td><code>lcomCOPpv</code></td>
<td>experimental bivariate inference</td>
</tr>
</tbody>
</table>

The Table of Probabilities that follows lists important relations between various joint probability concepts, the copula, nonexceedance probabilities \( u \) and \( v \), and exceedance probabilities \( u' \) and \( v' \). A compact summary of these probability relations has obvious usefulness. The notation \([\ldots]\) is to read as \([\ldots \text{ and} \ldots]\), and the \([\ldots|\ldots]\) is to be read as \([\ldots \text{ given} \ldots]\).

<table>
<thead>
<tr>
<th>Probability <code>\text{and}</code> Symbol</th>
<th>Convention</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \text{Pr}[ U \leq u, V \leq v ] )</td>
<td>( C(u, v) ) — The copula, ( \text{COP} )</td>
</tr>
<tr>
<td>( \text{Pr}[ U &gt; u, V &gt; v ] )</td>
<td>( C(u', v') ) — The survival copula, ( \text{surCOP} )</td>
</tr>
<tr>
<td>( \text{Pr}[ U \leq u, V &gt; v ] )</td>
<td>( u - C(u, v') )</td>
</tr>
</tbody>
</table>

The Type of Probabilities that follows lists important relations between various joint probability concepts, the copula, nonexceedance probabilities \( u \) and \( v \), and exceedance probabilities \( u' \) and \( v' \). A compact summary of these probability relations has obvious usefulness. The notation \([\ldots]\) is to read as \([\ldots \text{ and} \ldots]\), and the \([\ldots|\ldots]\) is to be read as \([\ldots \text{ given} \ldots]\).
The function `jointCOP` has considerable demonstration in its Note section of the joint and joint or relations shown through simulation and counting scenarios. Also there is a demonstration in the Note section of function `duCOP` on application of the concepts of joint and joint or conditions, and importantly joint mutually exclusive or conditions.

Permutation asymmetry can be added to a copula by `breveCOP`. One, two, or more copulas can be “composited,” “combined,” or “multiplied” in interesting ways to create highly unique bivariate relations and as a result, complex dependence structures can be formed. The package provides three main functions for copula composition: `composite1COP` composites a single copula with two compositing parameters, `composite2COP` composites two copulas with two compositing parameters, and `composite3COP` composites two copulas with four compositing parameters. Also two copulas can be combined through a weighted convex combination using `convexCOP` with a single weighting parameter, and even $N$ number of copulas can be combined by weights using `convexCOP`.

So-called “gluing” two copula by a parameter is provided by `glueCOP`. Finally, multiplication of two copulas to form a third is supported by `prod2COP`. All seven functions for compositing, combining, or multiplying copulas are compatible with joint probability simulation (`simCOP`), measures of association (e.g. $\rho_C$), and presumably all other copula operations using `copBasic` features.

### Useful Copula Relations by Visualization

There are a myriad of relations amongst variables computable through copulas, and these were listed in the Table of Probabilities earlier in this documentation. There is a script located in the inst/doc directory of the `copBasic` sources titled CopulaRelations_BaseFigure_inR.txt. This script demonstrates, using the PSP copula, relations between the copula (`cop`), survival copula (`surCOP`), joint survival function of a copula (`surfuncCOP`), co-copula (`coCOP`), and dual of a copula (`duCOP`)...
function (duCOP). The script performs simulation and manual counts observations meeting various criteria in order to compute their empirical probabilities. The script produces a base figure, which after extending in editing software, is suitable for educational description and is provided at the end of this documentation.

A Review of “Return Periods” using Copulas

Risk analyses of natural hazards are commonly expressed as annual return periods $T$ in years, which are defined for a nonexceedance probability $q$ as $T = 1/(1 - q)$. In bivariate analysis, there immediately emerge two types of return periods representing $T_{q;\text{coop}}$ and $T_{q;\text{dual}}$ conditions between nonexceedances of the two hazard sources (random variables) $U$ and $V$. It is usual in many applications for $T$ to be expressed equivalently as a probability $q$ in common for both variables.

Incidentally, the $Pr[U > u \mid V > v]$ and $Pr[V > v \mid U > u]$ probabilities also are useful for conditional return period computations following Salvadori et al. (2007, pp. 159–160) but are not further considered here. Also the $F_K(w)$ (Kendall Function or Kendall Measure of a copula) is the core tool for secondary return period computations (see kfuncCOP).

Let the copula $C(u, v; \Theta)$ for nonexceedances $u$ and $v$ be set for some copula family (formula) by a parameter vector $\Theta$. The copula family and parameters define the joint coupling (loosely meant the dependency/correlation) between hazards $U$ and $V$. If “failure” occurs if either or both hazards $U$ and $V$ are at probability $q$ threshold ($u = v = 1 - 1/T = q$) for $T$-year return period, then the real return period of failure is defined using either the copula $C(q, q; \Theta)$ or the co-copula $C^*(q', q'; \Theta)$ for exceedance probability $q' = 1 - q$ is

$$T_{q;\text{coop}} = \frac{1}{1 - C(q, q; \Theta)} = \frac{1}{C^*(1 - q, 1 - q; \Theta)} \quad \text{and} \quad \frac{1}{T_{q;\text{coop}}} \equiv \frac{1}{\text{cooperative risk}}.$$  

Or in words, the hazard sources collaborate or cooperate to cause failure. If failure occurs, however, if and only if both hazards $U$ and $V$ occur simultaneously (the hazards must “dually work together” or be “conjunctive”), then the real return period is defined using either the dual of a copula (function) $\tilde{C}(q, q; \Theta)$, the joint survival function $\overline{C}(q, q; \Theta)$, or survival copula $\hat{C}(q', q'; \Theta)$ as

$$T_{q;\text{dual}} = \frac{1}{1 - \tilde{C}(q, q; \Theta)} = \frac{1}{\overline{C}(q, q; \Theta)} = \frac{1}{\hat{C}(q', q'; \Theta)} \quad \text{and} \quad \frac{1}{T_{q;\text{dual}}} \equiv \frac{1}{\text{complement of dual protection}}.$$  

Numerical demonstration is informative. Salvadori et al. (2007, p. 151) show for a Gumbel–Hougaard copula (GHcop) having $\Theta = 3.055$ and $T = 1,000$ years ($q = 0.999$) that $T_{q;\text{coop}} = 797.1$ years and that $T_{q;\text{dual}} = 1,341.4$ years, which means that average return periods between “failures” are

$$T_{q;\text{coop}} \leq T \leq T_{q;\text{dual}} \quad \text{and thus} \quad 797.1 \leq T \leq 1314.4 \text{ years.}$$  

With the following code, these bounding return-period values are readily computed and verified using the prob2T() function from the lmomco package along with copBasic functions COP (generic functional interface to a copula) and duCOP (dual of a copula):
An early source (in 2005) by some of those authors cited on p. 151 of Salvadori et al. (2007; their citation “[67]”) shows $T_{q;\text{dual}} = 798$ years—a rounding error seems to have been committed. Finally just for reference, a Gumbel–Hougaard copula having $\Theta = 3.055$ corresponds to an analytical Kendall Tau (see GHcop) of $\tau \approx 0.673$, which can be verified through numerical integration available from \texttt{tauCOP} as:

\begin{verbatim}
tauCOP(cop=GHcop, para=3.055, brute=TRUE) # 0.6726542
\end{verbatim}

Thus, a “better understanding of the statistical characteristics of [multiple hazard sources] requires the study of their joint distribution” (Salvadori et al., 2007, p. 150).

**Interaction of copBasic to Copulas in Other Packages**

Originally, the \texttt{copBasic} package was not intended to be a port of the numerous bivariate copulas or over re-implementation other bivariate copulas available in R though as the package passed its 10th year in 2018, the original intent changed. It is useful to point out a demonstration showing an implementation of the Gaussian copula from the \texttt{copula} package, which is shown in the \texttt{Note} section of \texttt{med.regressCOP} in a circumstance of ordinary least squares linear regression compared to median regression of a copula as well as prediction limits of both regressions. Another demonstration in context of maximum pseudo-log-likelihood estimation of copula parameters is seen in the \texttt{Note} section \texttt{mleCOP}, and also see “API to the copula package” or “package copula (comparison to)” entries in the Index of this user manual.
Simulated value from parameterless "PSP" copula with a sample size of 1,000
The monospace typeface indicates commands in R using copBasic syntax
The slanted typeface indicates theoretical values from the copula and sample estimates from the data—
For this example, the survival copula has probability Pr[U > u and V > v] = 0.262 and the simulations have
258 occurrences in the respectively shown hatched region.

Author(s)
William Asquith <william.asquith@ttu.edu>

References
Examples

```r
## Not run:
# Nelsen (2006, p. 75, exer. 3.15b) provides for a nice test of copBasic features.
"mcdurv" <- function(u,v, theta) {
  ifelse(u > theta & u < 1-theta & v > theta & v < 1 - theta,
    return(M(u,v) - theta), # Upper bounds copula with a shift
    return(W(u,v))) # Lower bounds copula
}

"MCDURV" <- function(u,v, para=NULL) {
  if(is.null(para)) stop("need theta")
  if(para < 0 | para > 0.5) stop("theta ! in [0,1/2]")
  return(asCOP(u, v, f=mcdurv, para))
}

"afunc" <- function(t) { # a sample size = 1,000 hard wired
  return(cov(simCOP(n=1000, cop=MCDURV, para=t, ploton=FALSE, points=FALSE))[1,2])
}

set.seed(6234) # setup covariance based on parameter "t" and the "root" parameter
print(uniroot(afunc, c(0,0.5))) # "t" by simulation = 0.1023742
# Nelsen reports that if theta approx. 0.103 then covariance of U and V is zero.
# So, one will have mutually completely dependent uncorrelated uniform variables!

# Let us check some familiar measures of association:
rhoCOP( cop=MCDURV, para=0.1023742) # Spearman Rho = 0.005854481 (near zero)
tauCOP( cop=MCDURV, para=0.1023742) # Kendall Tau = 0.2648521
wolfCOP(cop=MCDURV, para=0.1023742) # S & W Sigma = 0.4690174 (less familiar)
D <- simCOP(n=1000, cop=MCDURV, para=0.1023742) # Plot mimics Nelsen (2006, fig. 3.11)
# Lastly, open research problem. L-comoments (matrices) measure high dimension of
# variable comovements (see lmomco package)---"method of L-comoments" for estimation?
lmomco::lcomom2(simCOP(n=1000, cop=MCDURV, para=0), nmom=5) # Perfect neg. corr.
lmomco::lcomom2(simCOP(n=1000, cop=MCDURV, para=0.5), nmom=5) # Perfect pos. corr.
# T2 (L-correlation), T3 (L-coskew), T4 (L-cokurtosis), and T5 matrices result. For
# Theta = 0 or 0.5 see the matrix symmetry with a sign change for L-coskew and T5 on
# the off diagonals (offdiags). See unities for T2. See near zero for offdiag terms
# in T2 near zero. But then see that T4 off diagonals are quite different from those
# for Theta 0.1024 relative to 0 or 0.5. As a result, T4 has captured a unique
# property of U vs V.
## End(Not run)
```

---

The Akaike Information Criterion between a Fitted Coupla and an Empirical Copula

**Description**

Compute the Akaike information criterion (AIC) \( \text{AIC}_C \) (Chen and Guo, 2019, p. 29), which is computed using mean square error \( \text{MSE}_C \) as

\[
\text{MSE}_C = \frac{1}{n} \sum_{i=1}^{n} \left( \mathbf{C}_n(u_i, v_i) - \mathbf{C}_{\Theta_m}(u_i, v_i) \right)^2
\]

and
\[ \text{AIC}_C = 2m + n \log(\text{MSE}_C), \]

where \( C_n(u_i, v_i) \) is the empirical copula (empirical joint probability) for the \( i \)th observation, \( C_{\Theta_m}(u_i, v_i) \) is the fitted copula having \( m \) parameters in \( \Theta \). The \( C_n(u_i, v_i) \) comes from \texttt{EMPIRcop}. The \( \text{AIC}_C \) is in effect saying that the best copula will have its joint probabilities plotting on a 1:1 line with the empirical joint probabilities, which is an \( \text{AIC}_C = -\infty \). From the \( \text{MSE}_C \) shown above, the root mean square error \( \text{rmseCOP} \) and Bayesian information criterion (BIC) \( \text{bicCOP} \) can be computed. These goodness-of-fits can assist in deciding one copula favorability over another, and another goodness-of-fit using the absolute differences between \( C_n(u, v) \) and \( C_{\Theta_m}(u, v) \) is found under \texttt{statTn}.

Usage

\[
\text{aicCOP}(u, v, \text{cop=NULL, para=NULL, m=NA, ...})
\]

Arguments

- \( u \) Nonexceedance probability \( u \) in the \( X \) direction;
- \( v \) Nonexceedance probability \( v \) in the \( Y \) direction;
- \( \text{cop} \) A copula function;
- \( \text{para} \) Vector of parameters or other data structure, if needed, to pass to the copula;
- \( m \) The number of parameters in the copula, which is usually determined by length of \( \text{para} \) if \( m=\text{NA} \), but some complex compositions of copulas are difficult to authoritatively probe for total parameter lengths and mixing coefficients; and
- \( \ldots \) Additional arguments to pass to either copula (likely most commonly to the empirical copula).

Value

The value for \( \text{AIC}_C \) is returned.

Author(s)

W.H. Asquith

References


See Also

\texttt{EMPIRcop, bicCOP, rmseCOP}
Examples

```r
## Not run:
S <- simCOP(80, cop=GHcop, para=5) # Simulate some probabilities, but we
# must then treat these as data and recompute empirical probabilities.
U <- lmomco::pp(S$U, sort=FALSE); V <- lmomco::pp(S$V, sort=FALSE)
# The parent distribution is Gumbel-Hougaard extreme value copula.
# But in practical application we do not know that, but say we speculate
# that perhaps the Galambos extreme value might be the parent. Then maximum
# likelihood is used to fit the single parameter.
pGL <- mleCOP(U, V, cop=GLcop, interval=c(0,20))$par
aics <- c(aicCOP(U, V, cop=GLcop, para=pGL),
aicCOP(U, V, cop=P),
aicCOP(U, V, cop=PSP))
names(aics) <- c("GLcop", "P", "PSP")
print(aics) # We will see that the first AIC is the smallest as the
# Galambos has the nearest overall behavior than the P and PSP copulas.
## End(Not run)
```

### AMHcop

#### Description

The Ali–Mikhail–Haq copula (Joe, 2014, p. 168) is

\[
C_\Theta(u, v) = \text{AMH}(u, v) = \frac{uv}{1 - \Theta(1 - u)(1 - v)},
\]

where \( \Theta \in [-1, +1) \), where the right boundary, \( \Theta = 1 \), can sometimes be considered valid according to Mächler (2014). The copula \( \Theta \to 0 \) becomes the **independence copula** \( (\Pi(u, v); P) \), and the parameter \( \Theta \) is readily computed from a **Kendall Tau** \( (\text{tauCOP}) \) by

\[
\tau_C = \frac{3\Theta - 2}{3\Theta} - \frac{2(1 - \Theta)^2 \log(1 - \Theta)}{3\Theta^2},
\]

and by **Spearman Rho** \( (\text{rhoCOP}) \), through Mächler (2014), by

\[
\rho_C = \sum_{k=1}^{\infty} \frac{3\Theta^k}{(k+2)^2}.
\]

The support of \( \tau_C \) is \([5 - 8 \log(2))/3, 1/3]\) \( \approx [-0.1817258, 0.3333333] \) and the \( \rho_C \) is \([33 - 48 \log(2), 4\pi^2 - 39]\) \( \approx [-0.2710647, 0.4784176] \), which shows that this copula has a limited range of dependency.

#### Usage

```r
AMHcop(u, v, para=NULL, rho=NULL, tau=NULL, fit=c("rho", "tau"), ...)
```
Arguments

- **u**: Nonexceedance probability $u$ in the $X$ direction;
- **v**: Nonexceedance probability $v$ in the $Y$ direction;
- **para**: A vector (single element) of parameters—the $\Theta$ parameter of the copula. However, if a second parameter is present, it is treated as a logical to reverse the copula $(u + v - 1 + \text{AMH}(1 - u, 1 - v; \Theta))$;
- **rho**: Optional Spearman Rho from which the parameter will be estimated and presence of rho trumps tau;
- **tau**: Optional Kendall Tau from which the parameter will be estimated;
- **fit**: If para, rho, and tau are all NULL, the the $u$ and $v$ represent the sample. The measure of association by the fit declaration will be computed and the parameter estimated subsequently. The fit has not other utility than to trigger which measure of association is computed internally by the cor function in R; and

... Additional arguments to pass.

Value

Value(s) for the copula are returned. Otherwise if tau is given, then the $\Theta$ is computed and a list having

- **para**: The parameter $\Theta$, and
- **tau**: Kendall Tau.

and if para=NULL and tau=NULL, then the values within $u$ and $v$ are used to compute Kendall Tau and then compute the parameter, and these are returned in the aforementioned list.

Note

Mächler (2014) reports on accurate computation of $\tau_C$ and $\rho_C$ for this copula for conditions of $\Theta \to 0$ and in particular derives the following equation, which does not have $\Theta$ in the denominator:

$$\rho_C = \sum_{k=1}^{\infty} \frac{3\Theta^k}{(k+2)^2}.$$ 

The copula package provides a Taylor series expansion for $\tau_C$ for small $\Theta$ in the copula::tauAMH(). This is demonstrated here between the implementation of $\tau = 0$ for parameter estimation in the copBasic package to that in the more sophisticated implementation in the copula package.

```r
copula::tauAMH(AMHcop(tau=0)$para) # theta = -2.313076e-07
copula:::.rhoAmhCopula(1E-9) # 3.333333e-10 (cutoff based)
```

It is seen that the numerical approaches yield quite similar results for small $\tau_C$, and finally, a comparison to the $\rho_C$ is informative:

```r
rhoCOP(AMHcop, para=1E-9) # 3.333333e-10 (two nested integrations)
copula:::.rhoAmhCopula(1E-9) # 3.333333e-10 (cutoff based)
```

```r
theta <- seq(-1,1, by=.0001)
RHoa <- sapply(theta, function(t) rhoCOP(AMHcop, para=t))
RHOb <- sapply(theta, function(t) copula:::.rhoAmhCopula(t))
plot(10^theta, RHoa-RHOb, type="l", col=2)
```
The plot shows that the apparent differences are less than 1 part in 100 million—The \texttt{copBasic}
computation is radically slower though, but \texttt{rhoCOP} was designed for generality of copula family.

\section*{Author(s)}

W.H. Asquith

\section*{References}


\section*{See Also}

\texttt{P}

\section*{Examples}

```r
## Not run:
t <- 0.9 # The Theta of the copula and we will compute Spearman Rho.
di <- integrate(function(t) log(t)/(1-t), lower=1, upper=(1-t))$value
A <- di*(1+t) - 2*log(1-t) + 2*t*log(1-t) - 3*t # Nelsen (2007, p. 172)
rho <- 12*A/t^2 - 3 # 0.4070369
rhoCOP(AMHcop, para=t) # 0.4070369
sum(sapply(1:100,function(k) 3*t^k/choose(k+2,2)^2)) # Machler (2014)
# 0.4070369 (see Note section, very many tens of terms are needed)
## End(Not run)

## Not run:
layout(matrix(1:2, byrow=TRUE)) # Note: Kendall Tau is same on reversal.
s <- 2; set.seed(s); nsim <- 10000
UVn <- simCOP(nsim, cop=AMHcop, para=c(0.9, "FALSE" ), col=4)
mtext("Normal definition [default]") # '2nd' parameter could be skipped
set.seed(s) # seed is used to keep Rho/Tau inside attainable limits for the example
UVr <- simCOP(nsim, cop=AMHcop, para=c(0.9, "TRUE"), col=2)
mtext("Reversed definition")
AMHcop2(UVn[,1], UVn[,2], fit="rho")$rho # -0.2581653
AMHcop2(UVr[,1], UVr[,2], fit="rho")$rho # -0.2570689
rhoCOP(cop=AMHcop, para=-0.9) # -0.2483124
AMHcop2(UVn[,1], UVn[,2], fit="tau")$tau # -0.1731904
AMHcop2(UVr[,1], UVr[,2], fit="tau")$tau # -0.1724820
tauCOP(cop=AMHcop, para=-0.9) # -0.1663313
## End(Not run)
```
Description

This function is intended to document and then to extend a simple API to end users to aid in implementation of other copulas for use within the copBasic package. There is no need or requirement to use asCOP for almost all users. However, for the mathematical definition of some copulas, the asCOP function might help considerably. This is because there is a need for special treatment of \( u \) and \( v \) vectors of probability as each interacts with the vectorization implicit in \( R \). The special treatment is needed because many copulas are based on the operators such as \( \min() \) and \( \max() \). When numerical integration used by the \( \text{integrate()} \) function in \( R \) in some copula operators, such as \( \text{tauCOP} \) for the Kendall Tau of a copula, special accommodation is needed related to the inherent vectorization in \( R \) and how \( \text{integrate()} \) works.

Basically, the problem is that one can not strictly rely in all circumstances on what \( R \) does in terms of value recycling when \( u \) and \( v \) are of unequal lengths. The source code is straightforward. Simply put, if lengths of \( u \) and \( v \) are unity, then there is no concern, and even if the length of \( u \) (say) is unity and \( v \) is 21, then recycling of \( u \) would often be okay. The real danger is when \( u \) and \( v \) have unequal lengths and those lengths are each greater than unity—the \( R \) treatment can not be universally relied upon with the various numerics herein involving optimization and nested numerical integration.

The example shows how a formula definition of a copula that is not a copula already implemented by copBasic is set into a function deltacop and then used inside another function UsersCop that will be the official copula that is compatible with a host of functions in copBasic. The use of asCOP provides the length check necessary on \( u \) and \( v \), and the argument \( \ldots \) provides optional parameter support should the user’s formula require more settings.

Usage

\[
\text{asCOP}(u, v, f=NULL, \ldots)
\]

Arguments

- **\( u \)**: Nonexceedance probability \( u \) in the \( X \) direction;
- **\( v \)**: Nonexceedance probability \( v \) in the \( Y \) direction;
- **\( f \)**: A function for which the user desires to make as a copula; and
- **\( \ldots \)**: Additional arguments to pass to the function \( f \) (such as parameters, if needed, for the copula in the form of a list).

Value

The value(s) for the copula are returned.

Author(s)

W.H. Asquith
References


See Also

COP

Examples

## Not run:
# Concerning Nelsen (2006, exer. 3.7, pp. 64--65)
"trianglecop" <- function(u,v, para=NULL, ...) {
  # If para is set, then the triangle is rotated 90d clockwise.
  if(! is.null(para) & para == 1) { t <- u; u <- v; v <- t }
  if(length(u) > 1 | length(v) > 1) stop("only scalars for this function")
  v2<-v/2; if(0 <= u & u <= v2 & v2 <= 1/2) { return(u )
  } else if(0 <= v2 & v2 < u & u < 1-v2) { return(v2 )
  } else if(1/2 <= 1-v2 & 1-v2 <= u & u <= 1 ) { return(u+v-1)
  } else { stop("should not be here in logic")
}
"UsersCop" <- function(u,v, ...) { asCOP(u,v, f=trianglecop, ...) }

n=20000; UV <- simCOP(n=n, cop=UsersCop)
# The a-d elements of the problem now follow:
# (a) Pr[V = 1 - |2*U -1|] = 1 and Cov(U,V) = 0; so that two random variables
# can be uncorrelated but each is perfectly predictable from the other
mean(UV$V - (1 - abs(2*UV$U -1))) # near zero; Nelsen says == 0
cov(UV$U, UV$V) # near zero; Nelsen says == 0
# (b) Cop(m,n) = Cop(n,m); so that two random variables can be identically
# distributed, uncorrelated, and not exchangeable
EMPIRcop(0.95,0.17, para=UV) # = A
EMPIRcop(0.17,0.95, para=UV) # = B; then A != B
# (c) Pr[V - U > 0] = 2/3; so that two random variables can be identically
# distributed, but their difference need not be symmetric about zero
length(UV$V - UV$U > 0)/n # about 2/3; Nelsen says == 2/3
# the prior two lines yield about 1/2 for independence copula P()
# (d) Pr[X + Y > 0] = 2/3; so that uniform random variables on (-1,1) can each
# be symmetric about zero, but their sum need not be.
length(UV$V + UV$U > 0)/n # about 2/3; Nelsen says == 2/3
# End(Not run)

## Not run:
# Concerning Nelsen (2006, exam. 3.10, p. 73)
"shufflecop" <- # assume scalar arguments for u and v
function(u,v, para, ...) {
  m <- para$mixer; subcop <- para$subcop
  if(is.na(m) | m <= 0 | m >= 1) stop("m ! in [0,1]")
  if(u <= m) { return(
    subcop(1-m+u, v, para=para$para) -
  )
  } else if(m < u & v <= m) { return(
    subcop(u-m, v, para=para$para) -
  )
  } else if(m < v & u <= m) { return(
    subcop(u-m, 1-m+v, para=para$para) -
  )
  } else if(u <= m & v <= m) { return(
    subcop(u-m, v-m, para=para$para) -
  )
  } else if(u <= m & v > m) { return(
    subcop(u-m, m-v, para=para$para) -
  )
  } else if(m < u & v > m) { return(
    subcop(u-m, m-v, para=para$para) -
  )
  } else if(m < u & v <= m) { return(
    subcop(u-m, v-m, para=para$para) -
  )
  } else if(u <= m & v > m) { return(
    subcop(u-m, m-v, para=para$para) -
  )
  } else { stop("should not be here in logic")
}
bicCOP

The Bayesian Information Criterion between a Fitted Copula and an Empirical Copula

Description

Compute the Bayesian information criterion (BIC) $\text{BIC}_C$ (Chen and Guo, 2019, p. 29), which is computed using mean square error $\text{MSE}_C$ as

$$\text{MSE}_C = \frac{1}{n} \sum_{i=1}^{n} (C_n(u_i, v_i) - C_{\Theta_m}(u_i, v_i))^2$$

and

$$\text{BIC}_C = m \log(n) + n \log(\text{MSE}_C),$$

where $C_n(u_i, v_i)$ is the empirical copula (empirical joint probability) for the $i$th observation, $C_{\Theta_m}(u_i, v_i)$ is the fitted copula having $m$ parameters in $\Theta$. The $C_n(u_i, v_i)$ comes from $\text{EMPIRcop}$. The $\text{BIC}_C$ is in effect saying that the best copula will have its joint probabilities plotting on a 1:1
line with the empirical joint probabilities, which is an $\text{BIC}_C = -\infty$. From the MSE shown above, the root mean square error $\text{rmseCOP}$ and Akaike information criterion (AIC) $\text{aicCOP}$ can be computed. These goodness-of-fits can assist in deciding on one copula favorability over another, and another goodness-of-fit using the absolute differences between $C_n(u,v)$ and $C_{\Theta_m}(u,v)$ is found under $\text{statTn}$.

Usage

\begin{verbatim}
bicCOP(u,v, cop=NULL, para=NULL, m=NA, ...)
\end{verbatim}

Arguments

- **u** Nonexceedance probability $u$ in the $X$ direction;
- **v** Nonexceedance probability $v$ in the $Y$ direction;
- **cop** A copula function;
- **para** Vector of parameters or other data structure, if needed, to pass to the copula;
- **m** The number of parameters in the copula, which is usually determined by length of **para** if $m=\text{NA}$, but some complex compositions of copulas are difficult to authoritatively probe for total parameter lengths and mixing coefficients; and
- **...** Additional arguments to pass to either copula (likely most commonly to the empirical copula).

Value

The value for $\text{BIC}_C$ is returned.

Author(s)

W.H. Asquith

References


See Also

EMPIRcop, aicCOP, rmseCOP

Examples

```r
## Not run:
S <- simCOP(80, cop=GHcop, para=5) # Simulate some probabilities, but we
# must then treat these as data and recompute empirical probabilities.
U <- lmomco::pp(S$U, sort=FALSE); V <- lmomco::pp(S$V, sort=FALSE)
# The parent distribution is Gumbel-Hougaard extreme value copula.
# But in practical application we don’t know that but say we speculate that
# perhaps the Galambos extreme value might be the parent. Then maximum
# likelihood is used on that copula to fit the single parameter.
```
bilmoms

Bivariate L-moments and L-comoments of a Copula

Description

Attention: This function is deprecated in favor of lcomCOP, which uses only direct numerical integrate() on the integrals shown below. The bilmoms function is strictly based on Monte Carlo integration.

Compute the bivariate L-moments (ratios) \( \delta_{1;C}^{[12]} \) of a copula \( C(u, v; \Theta) \) and remap these into the L-comoment matrix counterparts (Serfling and Xiao, 2007; Asquith, 2011) including L-correlation (Spearman Rho), L-coskew, and L-cokurtosis. As described by Brahimi et al. (2015), the first four bivariate L-moments \( \delta_{k}^{[12]} \) for random variable \( X^{(1)} \) or \( U \) with respect to \( (wrt) \) random variable \( X^{(2)} \) or \( V \) are defined as

\[
\begin{align*}
\delta_{1;C}^{[12]} &= 2 \int \int \mathcal{I}_{2}(C(u, v)) \, du \, dv - \frac{1}{2}, \\
\delta_{2;C}^{[12]} &= \int \int (12v - 6)C(u, v) \, du \, dv - \frac{1}{2}, \\
\delta_{3;C}^{[12]} &= \int \int (60v^2 - 60v + 12)C(u, v) \, du \, dv - \frac{1}{2}, \quad \text{and} \\
\delta_{4;C}^{[12]} &= \int \int (280v^3 - 420v^2 + 180v - 20)C(u, v) \, du \, dv - \frac{1}{2},
\end{align*}
\]

where the bivariate L-moments are related to the L-comoment ratios by

\[
6\delta_{k}^{[12]} = \tau_{k+1}^{[12]} \quad \text{and} \quad 6\delta_{k}^{[21]} = \tau_{k+1}^{[21]},
\]

where in otherwords, “the third bivariate L-moment \( \delta_{3}^{[12]} \) is one sixth the L-cokurtosis \( \tau_{k+1}^{[12]} \).” The first four bivariate L-moments yield the first five L-comoments (there is no first order L-comoment ratio). The terms and nomenclature are not easy and also the English grammar adjective “ratios” is not always consistent in the literature. The \( \delta_{k;C}^{[12]} \) are ratios, and the returned bilcomoms element by element by this function holds matrices for the marginal means, marginal L-scales and L-coscales, and then the ratio L-comoments.

Similarly, the \( \delta_{k}^{[21]} \) are computed by switching \( u \to v \) in the polynomials within the above integrals multiplied to the copula in the system of equations with \( u \). In general, \( \delta_{k}^{[12]} \neq \delta_{k}^{[21]} \) for \( k > 1 \) unless in the case of permutation symmetric (isCOP.permsym) copulas. By theory, \( \delta_{1}^{[12]} = \delta_{1}^{[21]} = \rho_C / 6 \) where \( \rho_C \) is a Spearman Rho rhoCOP.
The integral for $\delta^{[12]}_{4C}$ does not appear in Brahimi et al. (2015) but this and the other forms are verified in the Examples and discussion in Note. The four $k \in \{1, 2, 3, 4\}$ for $U \text{ wrt } V$ and $V \text{ wrt } U$ comprise a full spectrum of system of seven (not eight) equations. One equation is lost because $\delta^{[12]}_{11} = \delta^{[21]}_{11}$.

Usage

```
bilmoms(cop=NULL, para=NULL, only.bilmoms=FALSE, n=1E5, sobol=TRUE, scrambling=0, ...)```

**Arguments**

- **cop**
  - A copula function;
- **para**
  - Vector of parameters or other data structure, if needed, to pass to the copula;
- **only.bilmoms**
  - A logical to trigger return of the $\delta_k$ and skip L-comoment computation;
- **n**
  - The Monte Carlo integration size. The default seems to be at least an order of magnitude greater than needed for many applied problems;
- **sobol**
  - A logical triggering Sobol sequences for the Monte Carlo integration instead of the bivariate uniform distribution (independence). The Sobol sequences are dependent on the `sobol()` function of the `randtoolbox` package, and the Sobol sequences canvas the $I^2$ domain for smaller $n$ values than required if statistical independence is used for the Monte Carlo integration. Note that `randtoolbox` at least at version 2.0.+ has “scrambling” of Sobol sequences temporarily disabled, and hence `scrambling=0` as default for `bilmoms`;
- **scrambling**
  - The argument of the same name for `randtoolbox::sobol`;
- **...**
  - Additional arguments to pass to the `densityCOP` function.

**Value**

An R list of the bivariate L-moments is returned.

- **bilmomUV**
  - The bivariate L-moments $\delta^{[12]}_k$ of $U$ with respect to $V$ for $k \in \{1, 2, 3, 4\}$;
- **bilmomVU**
  - The bivariate L-moments $\delta^{[21]}_k$ of $V$ with respect to $U$ for $k \in \{1, 2, 3, 4\}$;
- **error.rho**
  - An “error” term in units of $\delta^{[12k-21]}_1$ used to judge whether the sample size for the Monte Carlo integration is sufficient based on a comparison to the Spearman Rho from direct numerical integration (not Monte Carlo based) using `rhoCOP` of the copula. Values for `error.rho < 1E-5` seem to be sufficient to judge whether $n$ is large enough;
- **bilcomoms**
  - If not only.bilmoms, another R list holding the L-comoments (see Note) computed by simple remapping of the $\delta^{[1]}_k$ and parallel in structure to the function `lcomoms2()` of the `lmomco` package; and
- **source**
  - An attribute identifying the computational source of the bivariate L-moments and bivariate L-comoments: “bilmoms.”
Note

The mapping of the bivariate L-moments to their L-comoment matrix counterparts is simple but nuances should be discussed and the meaning of the error.rho needs further description. The extra effort to form L-comoment matrices (Serfling and Xiao, 2007; Asquith, 2011) is made so that output matches the structure of the sample L-comoment matrices from the lcomoms2() function of the lmomco package.

Concerning the triangular or tent-shaped copula of Nelsen (2006, exer. 3.7, pp. 64–65) for demonstration, simulate from the triangular copula a sample of size \( m = 20,000 \) and compute some sample L-comoments using the following CPU intensive code. The function asCOP completes the vectorization needed for non-Monte Carlo integration for rhoCOP.

```r
"trianglecop" <- function(u,v, para=NULL, ...) {
  # If para is set, then the triangle is rotated 90d clockwise.
  if(! is.null(para) & & para == 1) { t <- u; u <- v; v <- t } 
  if(length(u) > 1 | length(v) > 1) stop("only scalars for this function")
  v2<-v/2; if(0 <= u & u <= v2 & v2 <= 1/2) { return(u ) }
  else if(0 <= v2 & v2 < u & u < 1-v2) { return(v2 ) }
  else if(1/2 <= 1-v2 & 1-v2 <= u & u <= 1 ) { return(u+v-1) }
  else { stop("should not be here in logic") }
}
"TriCop" <- function(u,v, ...) { asCOP(u,v, f=trianglecop, ...) }

m=20000; SampleUV <- simCOP(n=m, cop=TriCop, graphics=FALSE)
samLC <- lcomoms2(SampleUV, nmom=5)
theoLC <- bilmoms(cop=TriCop)
```

The Error in Rho Computation—The \( \rho_{C} \) of the copula by numerical integration is computed internally to bilmoms as

\[
\text{rhoC} \leftarrow \text{rhoCOP(cop=TriCop)} \# -1.733858e-17
\]

and used to compute the error.rho for bilmoms (see next code snippet). The \( \rho_{C} \) is obviously zero for this copula. Therefore, the bivariate association of TriCop is zero and thus is an example of a perfectly dependent situation yet of zero correlation. The bivariate L-moments and L-comoments of this copula are computed as

\[
\text{mean(replicate(20, bilmoms(cop=TriCop)$error.rho))} \# 7.650723e-06
\]

where the error.rho is repeated trials appears firmly \(<1e-5\), which is near zero (\( \epsilon_{\rho} \approx 0 \)). The error.rho term is defined by taking the first bivariate L-moment and numerically integrated \( \rho_{C} \) through rhoCOP and computing the terms

\[
\epsilon_{\rho}^{[12]} = |\delta_{1;C}^{[12]} - (\rho_{C}/6)|,
\]

\[
\epsilon_{\rho}^{[21]} = |\delta_{1;C}^{[21]} - (\rho_{C}/6)|,
\]

and

\[
\epsilon_{\rho} = \frac{\epsilon_{\rho}^{[12]} + \epsilon_{\rho}^{[21]}}{2},
\]

where the error.rho = \( \epsilon_{\rho} \), and values near zero are obviously favorable because this indicates that the Monte Carlo integration sample size \( n \) argument is sufficiently large to effectively canvas the
\(I^2\) domain. For the situation here, the theoretical \(\rho_C = 0\), but for \(n = 100\), the error\.rho ≈ 0.006 (e.g. bilmoms(n=100, cop=TriCop)$error\.rho) through a 20-unit replication, which is a hint that 100 samples are not large enough and that should be obvious.

The reasoning behind using the error\.rho between conventional numerical integration and the Monte Carlo integration (error\.rho) is that \(\rho_C\) is symmetrical. This choice of “convergence” assessment reduces somewhat the sample size needed for Monte Carlo integration into single number representing error.

**Discussion of Theoretical L-comoments**—The theoretical L-comoments in the format structure of the sample L-comoments by the `lcomoms2()` function of the lmomco package are formed by the bilmoms function, and the theoretical values are shown below in sequence with details listed by L-comoment. Now we extract the L-comoment matrices and show the first L-comoment matrix (the matrix of the means):

\[
\text{theoLClcm} <- \text{theoLC}\$bilcomoms
def print(theoLClcm$L1)
\begin{bmatrix}
[1,] 0.4999939 & NA \\
[2,] NA & 0.5000032
\end{bmatrix}
\]

where the diagonal should be filled with 1/2, if the \(n\) is suitably large, because 1/2 is the mean of the marginal uniform random variables. By definition the secondary diagonal has NAs. The values shown above are extremely close supporting the idea that default \(n\) is large enough. The matrix of means is otherwise uninformative.

The second L-comoment matrix (L-scales and L-coscales) is

\[
\text{print(theoLClcm$L2)}
\begin{bmatrix}
[1,] 1.666677e-01 & -3.466202e-06 \\
[2,] -3.466209e-06 & 1.666674e-01
\end{bmatrix}
\]

where the diagonal should be filled with 1/6, if the \(n\) is suitably large, because 1/6 is the univariate L-scale of the marginal uniform random variables. These values further support that default \(n\) is large enough. The diagonal is computed from the univariate L-moments of the margins of the Monte Carlo-generated edges and is otherwise uninformative. The secondary diagonal is a rescaling of the \(\delta_i^{[\cdot]}\) by the univariate L-moments of the margins to form L-coscales (nonratios). The copula is perfectly dependent but uncorrelated; so the secondary diagonal has near zeros.

The second L-comoment ratio matrix (coefficient of L-variations and L-correlations) is

\[
\text{print(theoLClcm$T2)}
\begin{bmatrix}
[1,] 1.000000e+00 & -2.079712e-05 \\
[2,] -2.079712e-05 & 1.000000e+00
\end{bmatrix}
\]

where the diagonal by definition has unities (correlation is unity for a variable on itself) but the secondary diagonal for the L-correlations has near zeros because again the copula is uncorrelated, and the secondary diagonal is computed from the \(\delta_i^{[\cdot]}\). These L-correlations are the Spearman Rho values computed external to the algorithms within rhoCOP.

The third L-comoment ratio matrix (L-skews and L-coskews) is
where the diagonal by definition should have zero zeros because the univariate L-skew of a uniform variable is zero. These values further support that default $n$ is large enough. The secondary diagonal holds L-coskews. The copula has L-coskew of $U \, \text{wrt} \, V$ of numerically near zero (symmetry) but measurable asymmetry of L-coskew of $V \, \text{wrt} \, U$ of $\tau_{3}^{[21]} \approx -0.75$.

The fourth L-comoment ratio matrix (L-kurtosis and L-cokurtosis) is

\begin{verbatim}
print(theoLClcm$T4)
[,1] [,2]
[1,] -2.623665e-06 -3.325177e-05
[2,] -2.162954e-04 -2.811630e-06
\end{verbatim}

where the diagonal by definition should have zero zeros because the univariate L-kurtosis of a uniform variable is zero—it has no peakedness. These values further support that default $n$ is large enough. The secondary diagonal holds L-cokurtoses and are near zero for this particular copula.

The fifth L-comoment ratio matrix (unnamed) is

\begin{verbatim}
print(theoLClcm$T5)
[,1] [,2]
[1,] 1.813344e-06 -1.296025e-04
[2,] 1.246436e-01 1.475012e-06
\end{verbatim}

where the diagonal by definition should have zero zeros because the univariate L-kurtosis of a uniform variable is zero—such a random variable has no asymmetry. These values further support that default $n$ is large enough. The secondary diagonal holds the fifth L-comoment ratios. The copula has $\tau_{5}^{[12]} = 0$ of $U \, \text{wrt} \, V$ of numerically near zero (symmetry) but measurable fifth-order L-comoment asymmetry of $V \, \text{wrt} \, U$ of $\tau_{5}^{[21]} \approx 0.125$.

**Comparison of Sample and Theoretical L-comoments**—The previous section shows theoretical values computed as $\tau_{3}^{[21]} \approx -0.75$ and $\tau_{5}^{[21]} \approx 0.125$ for the two L-comoments substantially away from zero. As sample L-comoments these values are `samLC$T3[2,1]` = $\hat{\tau}_{3}^{[21]} \approx -0.751$ and `samLC$T5[2,1]` = $\hat{\tau}_{5}^{[21]} \approx 0.123$. **CONCLUSION:** The sample L-comoment algorithms in the `lmomco` package are validated.

**Author(s)**

W.H. Asquith

**References**


See Also
lcomCOP, uvlmoms

Examples

```r
## Not run:
bilmoms(cop=PSP, n=10000, para=NULL, sobol=TRUE)$bilcomoms$T3
# results: Tau3[12]=-0.132, Tau3[21]=-0.132 (Monte Carlo)
lcomCOP(cop=PSP, para=NULL, orders=3)
# results: Tau3[12]=-0.129, Tau3[21]=-0.129 (direct integration)
## End(Not run)

## Not run:
# This stopped running sometime before June 2023. IS THIS IN COP()?
para <- list(alpha=0.5, beta=0.93, para1=4.5, cop1=GLcop, cop2=PSP)
bilmoms(cop=composite2COP, n=10000, para=para, sobol=TRUE)$bilcomoms$T3
# results: Tau3[12]=0.154, Tau3[21]=-0.0691 (Monte Carlo)
lcomCOP(cop=composite2COP, para=para, orders=3)
# results: Tau3[12]=0.156, Tau3[21]=-0.0668 (direct integration)
## End(Not run)

## Not run:
UVsim <- simCOP(n=20000, cop=composite2COP, para=para, graphics=FALSE)
samLcom <- lmomco::lcomoms2(UVsim, nmom=5) # sample algorithm
# results: Tau3[12]=0.1489, Tau3[21]=-0.0679 (simulation)
## End(Not run)
```

`blomatrixCOP`  
A Matrix of Blomqvist-like Betas of a Copula

Description

Compute the Blomqvist-like Betas matrix $\beta^c_2$-matrix of a copula, which is defined at presumably strategic points within $I^2$, as (for as `blomCOPss=FALSE` argument)

$$\beta^c_2 = \frac{C(u^c, v^c)}{\Pi(1/2, 1/2)} - 1,$$

where the $u^c$ and $v^c$ are of two types of gridded locations in $I^2$ space and if $u^c = 1/2$ and $v^c = 1/2$, then central location of the matrix is Blomqvist Beta (`blomCOP`). The definition of $\beta^c_2$ is such that the matrix is entirely zero for the independence copula ($\Pi(u, v)$) when $C(u^c, v^c) = \Pi(u, v)$ at the medial location $u, v = 1/2$. Also, the definition here might be unique to the copBasic package. The decile version (`blommatrixCOPdec`) of this function uses $u^c \in (1, 5, 9)/10$ and $v^c \in (1, 5, 9)/10$. Whereas, the quartile version (`blommatrixCOPiqr`) of this function uses $u^c \in (25, 50, 75)/100$.
and \( v^\circ \in (25, 50, 75)/100 \). If as.blomCOPss=TRUE argument is set (default operation), then the coordinate locations in the matrix become the \( \beta^\circ_C \) of blomCOPss. As a rule \( \beta^\circ_C \neq \beta^\circ_C \).

Usage

blomatrixCOPdec(cop=NULL, para=NULL, as.sample=FALSE, as.blomCOPss=TRUE,
ctype=c("weibull", "hazen", "1/n",
"bernstein", "checkerboard"), ...)
blomatrixCOPiqr(cop=NULL, para=NULL, as.sample=FALSE, as.blomCOPss=TRUE,
ctype=c("weibull", "hazen", "1/n",
"bernstein", "checkerboard"), ...)

Arguments

cop A copula function;
para Vector of parameters or other data structure, if needed, to pass to the copula;
as.sample A logical controlling whether an optional R data.frame in para is used to compute the \( \beta^\circ_C \)-matrix at which point the ctype argument will be passed to multiple calls of EMPIRcop;
as.blomCOPss A logical to trigger blomCOPss for each of the \((u, v)\) locations where for the blomCOPss calls \( \beta^\circ_C(u, v)\): \( u \mapsto (u, u) \mapsto u \) and \( v \mapsto (v, v) \mapsto v \);
cctype Argument of the same as EMPIRcop; and
... Additional arguments to pass to the copula.

Value

The matrix for \( \beta^\circ_C \) is returned depending on whether the decile or quartile version has been called.

Author(s)

W.H. Asquith

See Also

blomCOP, blomCOPss

Examples

round(blomatrixCOPdec(cop=P), digits=8); round(blomatrixCOPiqr(cop=P), digits=8)
# U|V=0.10 U|V=0.50 U|V=0.90
# U|V=0.90 0 0 0
# U|V=0.50 0 0 0
# U|V=0.10 0 0 0

round(blomatrixCOPdec(cop=PSP, as.blomCOPss=TRUE), digits=8)
# U|V=0.10 U|V=0.50 U|V=0.90
# U|V=0.90 0.4736842 0.8181818 0.5153268
# U|V=0.50 0.8181818 0.3333333 0.6459330
# U|V=0.10 0.8901099 0.4736842 0.1708292
```r
round(blomatrixCOPdec(cop=PSP, as.blomCOPss=FALSE), digits=8)
# U|V=0.10  U|V=0.50  U|V=0.90
# U|V=0.90  0.09890110 0.81818182 4.2631579
# U|V=0.50  0.05263158 0.33333333 0.8181818
# U|V=0.10  0.01010101 0.05263158 0.0989011

## Not run:
set.seed(1)
td <- c(0.10, 0.50, 0.90)
UVn <- simCOP(n=5000, cop=glueCOP, col=8,
  para=list(glue=0.4, cop1 =PLcop, cop2=PLcop,
  para1=PLpar(rho=-0.5), para2=PLpar(rho=+0.5)),
  para1=PLpar(rho=-0.5), para2=PLpar(rho=+0.5)))
points(td, rep(td[3], 3), cex=2, lwd=2, pch=3, col="red")
points(td, rep(td[2], 3), cex=2, lwd=2, pch=3, col="red")
points(td, rep(td[1], 3), cex=2, lwd=2, pch=3, col="red")
print(blomatrixCOPdec(as.sample=TRUE, para=UVn, ctype="weibull"))
# U|V=0.10  U|V=0.50  U|V=0.90
# U|V=0.90  -0.08222222 -0.580 -0.190
# U|V=0.50  0.30800000 0.112 0.264
# U|V=0.10  0.84000000 0.620 0.262

BMdn <- blomatrixCOPdec(cop=glueCOP,
  para=list(glue=0.4, cop1 =PLcop, cop2=PLcop,
  para1=PLpar(rho=-0.5), para2=PLpar(rho=+0.5)),
  para1=PLpar(rho=-0.5), para2=PLpar(rho=+0.5)))
print(round(BMdn, digits=8))
# U|V=0.10  U|V=0.50  U|V=0.90
# U|V=0.90  -0.08110464 -0.5569028 -0.2053772
# U|V=0.50  0.33449815 0.1202744 0.2424668
# U|V=0.10  0.75217766 0.6013719 0.2424668

## End(Not run)

## Not run:
set.seed(1); nsim <- 2000
para.pop <- list( cop1=GHcop, cop2=PLcop, alpha=0.359,
  para1=c(4.003, 1.099), para2=0.882, beta=0.292)
UVs <- simCOP(nsim, cop=composite2COP, para=para.pop)
print("GIVEN THIS SAMPLE")
Rho <- rhoCOP(as.sample=TRUE, para=UVs) # Spearman Rho
BMn <- blomatrixCOPdec(as.sample=TRUE, para=UVs, ctype="weibull")
parafn <- function(k) {
  c(exp(k[1])+1, exp(k[2]), exp(k[3]), pnorm(k[4]), pnorm(k[5]))
}
parafn_list <- function(k) {
  k <- parafn(k)
  list(cop1=GHcop, para1=c(k[1], k[2]), alpha=k[4],
    cop2=PLcop, para2=k[4], beta=k[5])
}
BLOM_ofun <- function(para, statmat=NULL, parafn=NULL, rho=NA) {
  para <- parafn(para)
```
new.para <- list(cop1=GHcop, para1=para[1:2], alpha=para[4],
  cop2=PLcop, para2=para[3], beta=para[5])
bm <- blomatrixCOPdec(cop=composite2COP, para=new.para)
err <- sum((statmat - bm)^2) + (rhoCOP(cop=composite2COP, para=new.para) - rho)^2
# print(c(para, err))
return(err)
}

run1 <- function(graphics=TRUE, nsim=0) {
  par.init <- c(log(1), log(1), log(1), qnorm(0.5), qnorm(0.5))
  rt <- optim(par.init, BLOM_ofun, statmat=BMn, parafn=parafn, rho=Rho)
  para <- parafn(rt$par)
  para.fit <- list(cop1=GHcop, cop2=PLcop, alpha=para[4],
    para1=para[1:2], para2=para[3], beta=para[5])
  uv <- simCOP(nsim, cop=composite2COP, para=para.fit, graphics=graphics, col=2)
  rmse <- round(rmseCOP(uv[,1], uv[,2], ctype="weibull",
    cop=composite2COP, para=para.fit), digits=8)
  if(graphics) mtext(paste0("RMSE(run1)=" , rmse))
  return(list(rmse=rmse, para=para.fit))
}

system.time(RUN1 <- run1(nsim=nsim))

par.init <- c(log(RUN1$para$para1[1]), log(RUN1$para$para1[2]),
  log(RUN1$para$para2), qnorm(RUN1$para$alpha), qnorm(RUN1$para$beta))

RMSE_ofun <- function(para, parafn=NULL) {
  para <- parafn(para)
  new.para <- list(cop1=GHcop, para1=para[1:2], alpha=para[4],
    cop2=PLcop, para2=para[3], beta=para[5])
  new.rmse <- rmseCOP(UVs[,1], UVs[,2], cop=composite2COP, para=new.para)
  # print(c(para, new.rmse))
  return(new.rmse)
}

run2 <- function(graphics=TRUE, nsim=0, par.init=NULL) {
  if(is.null(par.init)) {
    par.init <- c(log(1), log(1), log(1), qnorm(0.5), qnorm(0.5))
  }
  rt <- optim(par.init, RMSE_ofun, parafn=parafn)
  para <- parafn(rt$par)
  para.fit <- list(cop1=GHcop, cop2=PLcop, alpha=para[4],
    para1=para[1:2], para2=para[3], beta=para[5])
  uv <- simCOP(nsim, cop=composite2COP, para=para.fit, graphics=graphics, col=4)
  rmse <- round(rmseCOP(uv[,1], uv[,2], ctype="weibull",
    cop=composite2COP, para=para.fit), digits=8)
  if(graphics) mtext(paste0("RMSE(run2)=" , rmse))
  return(list(rmse=rmse, para=para.fit))
}

system.time(RUN2.1 <- run2(nsim=nsim, par.init=par.init))


GIVN <- c(para.pop$alpha, para.pop$para1, para.pop$beta, para.pop$para2)
FIT1 <- c(RUN1$para$alpha, RUN1$para$para1, RUN1$para$beta, RUN1$para$para2)
FIT1 <- round(FIT1, digits=3)
FIT2.1 <- c(RUN2.1$para$alpha, RUN2.1$para$para1, RUN2.1$para$beta, RUN2.1$para$para2)
FIT2.1 <- round(FIT2.1, digits=3)
FIT2.2 <- c(RUN2.2$para$alpha, RUN2.2$para$para1, RUN2.2$para$beta, RUN2.2$para$para2)
FIT2.2 <- round(FIT2.2, digits=3)
nms <- c("what", "alpha", "para1", "para2")
GIVN <- c("given", GIVN); FIT2.1 <- c("by_Blom1", FIT2.1)
FIT1 <- c("by_RMSE", FIT1); FIT2.2 <- c("by_Blom2", FIT2.2)
names(GIVN) <- nms; names(FIT1) <- nms
names(FIT2.1) <- nms; names(FIT2.2) <- nms
RESL <- cbind(data.frame(GIVN), data.frame(FIT1), data.frame(FIT2.1), data.frame(FIT2.2))
print(RESL) #
## End(Not run)

---

**blomCOP**  
*The Blomqvist Beta of a Copula*

**Description**

Compute the *Blomqvist Beta* $\beta_C$ of a copula (Nelsen, 2006, p. 182), which is defined at the middle or center of $I^2$ as

$$\beta_C = 4 \times C \left( \frac{1}{2}, \frac{1}{2} \right) - 1,$$

where the $u = v = 1/2$ and thus shows that $\beta_C$ is based on the median joint probability. The Blomqvist Beta is also called the *medial correlation coefficient*. Nelsen also reports that “although, the Blomqvist Beta depends only on the copula only through its value at the center of $I^2$, but that $[\beta_C]$ nevertheless often provides an accurate approximation to both Spearman Rho $\rho_C$ and Kendall Tau $\tau_C$. Kendall Tau $\tau_C$, Gini Gamma $\gamma_C$, and Spearman Rho $\rho_C$ in relation to $\beta_C$ satisfy the following inequalities (Nelsen, 2006, exer. 5.17, p. 185):

$$\frac{1}{4}(1 + \beta_C)^2 - 1 \leq \tau_C \leq 1 - \frac{1}{4}(1 - \beta_C)^2,$$

$$\frac{3}{16}(1 + \beta_C)^3 - 1 \leq \rho_C \leq 1 - \frac{3}{16}(1 - \beta_C)^3,$$

and

$$\frac{3}{8}(1 + \beta_C)^2 - 1 \leq \tau_C \leq 1 - \frac{3}{8}(1 - \beta_C)^2.$$

A curious aside (Joe, 2014, p. 164) about the *Gaussian copula* is that Blomqvist Beta is equal to Kendall Tau ($\tau_C$) (see *Note* in med.regressCOP for a demonstration). Finally, a version of Blomqvist Beta defined outside the median is provided by *blomCOPss*.

**Usage**

```r
blomCOP(cop=NULL, para=NULL, as.sample=FALSE, 
ctype=c("joe", "weibull", "hazen", "1/n", 
"bernstein", "checkerboard"), ...)
```
Arguments

cop A copula function;
para Vector of parameters or other data structure, if needed, to pass to the copula;
as.sample A logical controlling whether an optional \( \mathbb{R} \) data.frame in para is used to compute the \( \hat{\beta}_C \) (see Note);
cctype Argument of the same as EMPIRcop with the exception of the "joe" specific to the documentation here. The other choices trigger and are given over to the empirical copula; and
... Additional arguments to pass to the copula or down to EMPIRcop if a sample version had been requested.

Value

The value for \( \hat{\beta}_C \) or sample \( \hat{\beta}_n \) is returned.

Note

The sample \( \hat{\beta}_n \) is most efficiently computed (Joe, 2014, p. 57) by

\[
\hat{\beta}_n = \frac{2}{n} \sum_{i=1}^{n} 1 \left( [r_{i1} - (1 + n)/2] \times [r_{i2} - (1 + n)/2] \geq 0 \right) - 1,
\]

where \( r_{i1}, r_{i2} \) are the ranks of the data for \( i = 1, \ldots, n \), and \( 1(.) \) is an indicator function scoring 1 if condition is true otherwise zero. However, the Joe sample estimator is not fully consistent (or vice versa) with the various versions of the empirical copula, \( C_n \) (EMPIRcop) (see the last example in Examples). Also, the nature of even and odd sample sizes controls how the median is computed and the issue of samples lying on the median lines in \( U \) and \( V \) (Genest et al., 2013). The argument ctype supports triggers to the \( C_n \) in lieu of the Joe sample estimator shown in this documentation.

Author(s)

W.H. Asquith

References


See Also

blomCOPss, blomatrixCOPdec, blomatrixCOPiqr, footCOP, giniCOP, hoefCOP, rhoCOP, tauCOP, wolfCOP, joeskewCOP, uvlmoms
Examples

blomCOP(cop=PSP) # 1/3 precisely

## Not run:
# Nelsen (2006, exer. 5.17, p. 185) : All if(...) are TRUE
B <- blomCOP(cop=N4212cop, para=2.2); Bp1 <- 1 + B; Bm1 <- 1 - B
G <- giniCOP(cop=N4212cop, para=2.2); a <- 1/4; b <- 3/16; c <- 3/8
R <- rhoCOP(cop=N4212cop, para=2.2)
K <- tauCOP(cop=N4212cop, para=2.2, brute=TRUE) # numerical issues without brute
if( a*Bp1^2 - 1 <= K & K <= 1 - a*Bm1^2 ) print("TRUE") #
if( b*Bp1^3 - 1 <= R & R <= 1 - b*Bm1^3 ) print("TRUE") #
if( c*Bp1^2 - 1 <= G & G <= 1 - c*Bm1^2 ) print("TRUE") #
## End(Not run)

## Not run:
# A demonstration of a special feature of blomCOP for sample data.
# Joe (2014, p. 60; table 60) has 0.749 for GHcop(tau=0.5); n*var(hatB) as n-->infinity
set.seed(1)
theta <- GHcop(tau=0.5)$para; B <- blomCOP(cop=GHcop, para=theta); n <- 1000
H <- sapply(1:1000, function(i) { # Let us test that with pretty large sample size:
           blomCOP(para=rCOP(n=n, cop=GHcop, para=theta), as.sample=TRUE) })
print(n*var(B-H)) # For 1,000 sims of size n : 0.789, nearly matches Joe's result
## End(Not run)

## Not run:
# Joe (2014, p. 57) says that sqrt(n)(B-HatBeta) is Norm(0, 1 - B^2)
set.seed(1)
n <- 10000; B <- blomCOP(cop=PSP) # Beta = 1/3
H <- sapply(1:100, function(i) { message(i,"-", appendLF=FALSE)
bloomCOP(para=rCOP(n=n, cop=PSP), as.sample=TRUE) })
lmomco::parnor(lmomco::lmoms(sqrt(n)*(H-B))) # mu = -0.038; sigma = 0.970
# Joe (2014) : sqrt(1-B^2) == standard deviation (sigma) : (1-(1/3)^2) approx 0.973
## End(Not run)

## Not run:
nn <- 200; set.seed(1)
UV <- simCOP(n=nn+1, cop=PSP, graphics=FALSE)
for(n in nn:(nn+1)) {
  if(as.logical(n %% 2)) { # in source \ percent \ percent for latex
    message(\"Blomquist Betas for an odd sample size n=\", n)
    uv <- UV
  } else {
    message(\"Blomquist Betas for an even sample size n=\", n)
    uv <- UV[-(nn+1), ] # remove the last and 'odd' indexed value to make even
  }
}
message(c(" Joe2014: ", blomCOP(as.sample=TRUE, para=uv, ctype="joe")))
message(c(" Weibull: ", blomCOP(as.sample=TRUE, para=uv, ctype="weibull")))
message(c(" Hazen: ", blomCOP(as.sample=TRUE, para=uv, ctype="hazen")))
message(c(" 1/n: ", blomCOP(as.sample=TRUE, para=uv, ctype="1/n")))
message(c(" Bernstn: ", blomCOP(as.sample=TRUE, para=uv, ctype="bernstein")))
message(c(" ChckBrd: ", blomCOP(as.sample=TRUE, para=uv, ctype="checkerboard")))
}
# Blomquist Betas for an even sample size n=200
# Joe2014: 0.32
# Weibull: 0.32
# Hazen: 0.32
# 1/n: 0.32
# Bernstn: 0.323671819423416
# CheckBrk: 0.32

# Blomquist Betas for an odd sample size n=201
# Joe2014: 0.323383084577114
# Weibull: 0.333333333333333
# Hazen: 0.333333333333333
# 1/n: 0.293532338308458
# Bernstn: 0.327747577290946
# CheckBrk: 0.313432835820896

## End(Not run)

### blomCOPss

**Blomqvist (Schmid–Schmidt) Betas of a Copula**

**Description**

Compute the Blomqvist (Schmid–Schmidt) Betas $\beta^\diamond_C$ (Schmid and Schmidt, 2007) defined for arbitrary dimension $d$ of a copula $C(u_1, \cdots, u_d; \Theta)$ (COP) for parameters $\Theta$. The copula survival function is $\overline{C}(u_1, \cdots, u_d; \Theta)$ (surfuncCOP). The Beta, though the copBasic package is built around bivariate copula only, is defined as

$$\beta^\diamond_C = h_d(u, v) \left[ (C(u) + \overline{C}(v)) - g_d(u, v) \right],$$

where $h_d$ and $g_d$ are norming constants defined below. The superscript $\diamond$ (diamond) is chosen for copBasic because of the alliteration to “dimension.” The bold face for $u$ and $v$ are denote that these arguments are vectors of length $d$ reflecting “cutting points” on nonexceedence probabilities in each of the dimensions. The $u$ functions as the arguments $(u, v)$ pair used in copula of this package and represents the first cutting point for a $P[U \leq u, V \leq v] = C(u, v)$, and $v$ functions as the arguments $u, v$ pair for this package and represents the second cutting point for a $P[U > u, V > v] = 1 - u - v + C(u, v) = \overline{C}(u, v)$. This notation of vectored (bold face) and nonvectored “u” and “v” is a little obtuse but as the properties of $\beta^\diamond_C$ are summarized clarity for the reader is anticipated. In short, the $u$ will reference the coordinate pairs in the lower right quadrant and the $v$ will reference the coordinate pairs in the upper right quadrant.

The norming constant $h_d$ is defined as

$$h_d(u, v) = \frac{1}{(\min(u_1, \cdots, u_d) + \min(1 - v_1, \cdots, 1 - v_d) - g_d(u, v))},$$

and $g_d$ is defined as

$$g_d(u, v) = \prod_{i=1}^{d} u_i + \prod_{i=1}^{d} (1 - v_i),$$

where the cutting points $u$ and $v$ are in a domain $D : \{ (u, v) \} \in [0, 1]^d$ given $u \leq v$ and $u > 0$ or $v < 1$. The reader must careful remember that these $u$ and $v$ are vectors of probabilities.
The norming constants provide for $-1 \leq \beta_C^{\circ} \leq +1$. Using the function argument defaults for $d = 2$ dimensions $u = (1,1)/2$ for $uu$ and $v = (1,1)/2$ for $vv$, results in (1) $\beta_C^{\circ} = 1$ if $C = M$ comonotonicity copula \((M)\) \((\text{blomCOPss(cop=M)} \Rightarrow 1)\), (2) $\beta_C^{\circ} = 0$ if $C = P$ independence copula \((P)\) \((\text{blomCOPss(cop=P)} \Rightarrow 0)\), and (3) if $C = W$ countermonotonicity copula \((W)\) \((\text{blomCOPss(cop=W)} \Rightarrow -1)\).

Schmid and Schmidt (2007) list three important cases extending the $M$ and $P$ examples. First, $\beta_C^{\circ}(1/2, 1/2) = \beta_C(1/2, 1/2)$, which is Blomqvist Beta \((\beta_C(1/2, 1/2))\) \((\text{blomCOP})\) and measures overall dependence.

Second, $\beta_C^{\circ}(u, v)$ with $u < 1/2 < v$, which measures dependence in the tail regions. (Note that the author of \texttt{copBasic} thinks “regions” as a plural is need in the previous sentence; Schmid and Schmidt (2007) use the singular “region.” This is potentially important as seemingly simultaneous tail dependency in the lower and upper perspectives would be provided. More discussion is provided in \texttt{Examples}.)

Third and presumably very important in practical applications, $\lim_{p \downarrow 0} \beta_C^{\circ}(p, 1) = \lambda_C^{U_L}$ for $p = u = (p, \cdots, p)$ measures lower-tail dependence. This measure is equal to the lower-tail dependence parameter $\lambda_C^{L_U} = \lambda_C^{U_L}$ without some of the computational nuances required as $\lambda_C^{L_U}$ is defined at \texttt{taildepCOP}.

Schmid and Schmidt (2007) do not list how the upper-tail dependence parameter $\lambda_C^{U_L}$ could be computed in terms of $\beta_C^{\circ}$.

\texttt{Usage}

```
blomCOPss(cop=NULL, para=NULL, uu=rep(0.5, 2), vv=rep(0.5, 2), trap.nan=TRUE,
         as.sample=FALSE, ctype=c("weibull", "hazen", "1/n",
             "bernstein", "checkerboard"), ...)
```

\texttt{Arguments}

- \texttt{cop} A copula function;
- \texttt{para} Vector of parameters or other data structure, if needed, to pass to the copula;
- \texttt{uu} The vector for $u$ and the defaults with $vv$ as such for same operation as \texttt{blomCOP} \((\beta_C^{\circ}(1/2, 1/2))\);
- \texttt{vv} The vector for $v$ and the defaults with $uu$ as such for same operation as \texttt{blomCOP} \((\beta_C^{\circ}(1/2, 1/2))\);
- \texttt{trap.nan} A logical to trigger 0 if \((0,0)\) is NaN or if \((1,1)\) is NaN. This feature is present on a package-specific purpose because the \texttt{PSP} copula deliberately retains edge NaN as a stress case;
- \texttt{as.sample} A logical controlling whether an optional \texttt{R} data.frame in \texttt{para} is used to compute the $\beta_C^{\circ}$ at which point the \texttt{ctype} argument will be passed to \texttt{EMPIRcop};
- \texttt{ctype} Argument of the same as \texttt{EMPIRcop}; and
- \texttt{...} Additional arguments to pass to the copula.
Value

The $\beta_C$ is returned.

Note

Sample estimation of the $\beta_C$ is possible. The `as.sample` triggers internally a call to the empirical copula ($C_n$) (`EMPIRcop`) for the `ctype` for the copula and its survival function form. Expansive more details are provided by `taildepCOP` (section Note::DEMONSTRATION (Tail Dependence)). A comparison of the $\lambda_C^{U}$ and $\lambda^{U}_{\beta C}$ is made.

Author(s)

W.H. Asquith

References


See Also

`blomCOP, blomatrixCOP, taildepCOP`

Examples

```r
blomCOP( cop=PSP) # [1] 0.3333333
blomCOPss(cop=PSP) # [1] 0.3333333

## Not run:
# The calls below for blomCOPss() are technically the same for sample versions.
UV <- simCOP(1000, cop=PSP, graphics=FALSE) # HatBeta(0.1,0.9) = 0.277
blomCOPss(para=UV, cop=EMPIRcop, uu=c(0.1,0.1), vv=c(0.90,0.90))
blomCOPss(para=UV, as.sample=TRUE, uu=c(0.1,0.1), vv=c(0.90,0.90)) #
## End(Not run)

## Not run:
set.seed(1)
para <- c(3, 6) # define parameters of two-parameter GHcop
UV <- simCOP(1000, cop=GHcop, para=para) # simulate to show general structure

# compute the tail dependencies from having into the limits
taildepCOP(cop=GHcop, para=para, plot=TRUE)
# lower tail dependency = 0.96222
# upper tail dependency = 0.74008
# The two parameters influence how strongly the tail dependencies are.

# Schmid and Schmidt (2007, eq. 24) define the lower-tail dependency in terms of
# the Beta and p--->0 Beta(c(p,p), c(1,1)). Let's compute these and produce content
# suitable to show on the tail-dependency plot that the assertion for the lower
# dependency by Beta() is correct, which it is and then extend to the upper-tail
```
# dependency parameter that the authors seem to not have defined.
usr <- par()$usr[1:2]  # grab horizontal edges of the plot, and set up the
uuLO <- rep(pnorm(usr[1]), 2)  # the uu for the lower tail and the vv for the upper
vvUP <- rep(pnorm(usr[2]), 2)  # tail and then plot both with overplotting symbols

# lower-tail estimate and see how it plots along the value from taildepCOP()
SchmidsL <- blomCOPss(cop=GHcop, para=para, uu=uuLO, vv=c(1,1))
points(usr[1], SchmidsL, col="darkgreen", cex=2, pch=1, lwd=2)
points(usr[1], SchmidsL, col="darkgreen", cex=2, pch=3, lwd=2)
points(usr[1], SchmidsL, col="darkgreen", cex=2, pch=4, lwd=2)

# upper-tail estimate and see how it plots along the value from taildepCOP()
SchmidsU <- blomCOPss(cop=GHcop, para=para, uu=c(0,0), vv=vvUP)
points(usr[2], SchmidsU, col="darkgreen", cex=2, pch=1, lwd=2)
points(usr[2], SchmidsU, col="darkgreen", cex=2, pch=3, lwd=2)
points(usr[2], SchmidsU, col="darkgreen", cex=2, pch=4, lwd=2)

# SchmidsL lower tail dependency = 0.962224
# SchmidsU upper tail dependency = 0.740079
# The author has an expectation that the SchmidsL and SchmidsU values are
# more reliable than those stemming from taildepCOP() because of the limiting
# behavior (or its implementation therein) compared to direct computation by
# blomCOPss().

# Mow for sake of curiosity, let us see how the trajectory of the Blomqvist
# (Schmid--Schmidt) Betas at arriving at the tail dependencies as p-->0|1.
# It is very informative that the trajectories of blomCOPss() and taildepCOP()
# as each hones towards the two dependency parameters are different and this
# highlights the fact that the computational underpinnings are different.
psl <- pnorm(seq(0, usr[1], by=-diff(range(c(0, usr[1]))) / 1000))
lines(qnorm(psl), sapply(psl, function(p) {
  blomCOPss(cop=GHcop, para=para, uu=rep(p, 2), vv=c(1,1)) } ),
col="darkgreen", lty=2, lwd=2)

psu <- pnorm(seq(0, usr[2], by= diff(range(c(0, usr[2]))) / 1000))
lines(qnorm(psu), sapply(psu, function(p) {
  blomCOPss(cop=GHcop, para=para, uu=c(0,0), vv=rep(p, 2)) } ),
col="darkgreen", lty=2, lwd=2) #
## End(Not run)

---

**Add Asymmetry to a Copula**

**Description**

Adding permutation asymmetry (Chang and Joe, 2020, p. 1596) (isCOP.permsym) is simple for a bivariate copula family. Let $C$ be a copula with respective vectors of parameters $\Theta_C$, then the permutation asymmetry is add through an asymmetry parameter $\beta \in (-1, +1)$ for $-1 \leq \beta < 0$ by

$$\tilde{C}_{\beta, \Theta}(u, v) = v^{-\beta} \cdot C(u, v^{(1+\beta)}; \Theta),$$

for $0 \leq \beta \leq +1$ by

$$\tilde{C}_{\beta, \Theta}(u, v) = u^{+\beta} \cdot C(u^{(1-\beta)}, v; \Theta).$$
Usage

breveCOP(u, v, para, ...)

Arguments

u  Nonexceedance probability $u$ in the $X$ direction;

v  Nonexceedance probability $v$ in the $Y$ direction;

para A special parameter list (see Note); and

... Additional arguments to pass to the copula.

Value

Value(s) for the copula are returned.

Note

The following descriptions list in detail the structure and content of the para argument where cop1 and cop and para1 and para are respectively synonymous to have some structural similarity to the various copula constructors (compositors) of the copBasic package:

beta — The $\beta$ asymmetry parameter;

cop — Function of the copula $C$;

cop1 — Function of the coupla $C$;

para — Vector of parameters $\Theta_C$ for $C$; and

para1 — Vector of parameters $\Theta_C$ for $C$.

Author(s)

W.H. Asquith

References


See Also

COP, convex2COP, convexCOP, composite1COP, composite2COP, composite3COP, FRECHETcop, glueCOP

Examples

```
para <- list(beta=0.24, cop1=FRECHETcop, para1=c(0.4, 0.56))
breveCOP(0.87, 0.35, para=para) # 0.282743
```

## Not run:

# Notice the argument cop and para name adjustments to show that
# translation exists inside the function to have use flexibility.
The Clayton Copula

**Description**

The **Clayton copula** (Joe, 2014, p. 168) is

\[
C_\Theta(u, v) = \text{CL}(u, v) = \max\left\{\left(u^{-\Theta} + v^{-\Theta} - 1; 0\right)\right\}^{-1/\Theta},
\]

where \(\Theta \in [-1, \infty)\), \(\Theta \neq 0\). The copula, as \(\Theta \to -1^+\) limits, to the **countermonotonicity copula** \(W(u, v); W\), as \(\Theta \to 0\) limits to the **independence copula** \(\Pi(u, v); \Pi\), and as \(\Theta \to \infty\), limits to the **comonotonicity copula** \(M(u, v); M\). The parameter \(\Theta\) is readily computed from a **Kendall Tau** (\(\tau_{COP}\)) by \(\tau_C = \Theta / (\Theta + 2)\).

**Usage**

\[
\text{CLcop}(u, v, \text{para=NULL, \tau=}NULL, \ldots)
\]

**Arguments**

- \(u\) Nonexceedance probability \(u\) in the \(X\) direction;
- \(v\) Nonexceedance probability \(v\) in the \(Y\) direction;
- \(\text{para}\) A vector (single element) of parameters—the \(\Theta\) parameter of the copula;
- \(\tau\) Optional Kendall Tau; and
- \(\ldots\) Additional arguments to pass.
Value

Value(s) for the copula are returned. Otherwise if \( \tau \) is given, then the \( \Theta \) is computed and a list having

\[
\text{para} \quad \text{The parameter } \Theta, \text{ and} \\
\tau \quad \text{Kendall Tau.}
\]

and if \( \text{para} = \text{NULL} \) and \( \tau = \text{NULL} \), then the values within \( u \) and \( v \) are used to compute Kendall Tau and then compute the parameter, and these are returned in the aforementioned list.

Author(s)

W.H. Asquith

References


See Also

\( \text{M, P, W} \)

Examples

# Lower tail dependency of Theta = pi --> 2^(-1/pi) = 0.8020089 (Joe, 2014, p. 168) 
\text{taildepCOP(cop=CLcop, para=pi)$lambdaL} # 0.80201

---

\text{coCOP} \quad \text{The Co-Copula Function}

Description

Compute the \textit{co-copula (function)} from a copula (Nelsen, 2006, pp. 33–34), which is defined as

\[
\Pr[U > u \text{ or } V > v] = C^*(u', v') = 1 - C(u', v'),
\]

where \( C^*(u', v') \) is the co-copula and \( u' \) and \( v' \) are exceedance probabilities and are equivalent to \( 1 - u \) and \( 1 - v \) respectively. The co-copula is the expression for the probability that either \( U > u \) \textbf{or} \( V > v \) when the arguments to \( C^*(u', v') \) are exceedance probabilities, which is unlike the \textit{dual of a copula (function)} (see \texttt{duCOP}) that provides \( \Pr[U \leq u \text{ or } V \leq v] \).

The co-copula is a function and not in itself a copula. Some rules of copulas mean that \( C(u, v) + C^*(u', v') \equiv 1 \) or in \texttt{copBasic} syntax that the functions \( \text{COP}(u, v) + \text{coCOP}(u, v) \) equal unity if the exceedance argument to \texttt{coCOP} is set to FALSE.

The function \texttt{coCOP} gives “risk” against failure if failure is defined as either hazard source \( U \) or \( V \) occurring by themselves or if both occurred at the same time. Expressing this in terms of an annual probability of occurrence \( (q) \), one has

\[
q = 1 - \Pr[U > u \text{ or } V > v] = C^*(u', v') \text{ or}
\]
in R code 

\[
\text{q} \leftarrow \text{coCOP}(u, v, \text{exceedance}=\text{FALSE}, \ldots)
\]

So, in yet other words and as a mnemonic: A co-copula is the probability of exceedance if the hazard sources collaborate or cooperate to cause failure. Also, \(q\) can be computed by 

\[
\text{q} \leftarrow \text{coCOP}(1 - u, 1 - v, \text{exceedance}=\text{TRUE}, \ldots)
\]

**Usage**

\[
\text{coCOP}(u, v, \text{cop=NULL, para=NULL, exceedance=TRUE, \ldots})
\]

**Arguments**

- **u**  
  Exceedance probability \((u' = 1 - u)\) in the \(X\) direction;

- **v**  
  Exceedance probability \((v' = 1 - v)\) in the \(Y\) direction;

- **cop**  
  A copula function;

- **para**  
  Vector of parameters or other data structure, if needed, to pass to the copula;

- **exceedance**  
  A logical controlling the probability direction. Are \(u\) and \(v\) values given really \(u'\) and \(v'\), respectively? If \(\text{FALSE}\), then the complements of the two are made internally and the nonexceedances can thus be passed; and

- **...**  
  Additional arguments to pass to the copula.

**Value**

The value(s) for the co-copula are returned.

**Note**

The author (Asquith) finds the use of exceedance probabilities delicate in regards to Nelsen’s notation. The coCOP function and surCOP have the exceedance argument to serve as a reminder that the co-copula as defined in the literature uses exceedance probabilities as its arguments, although the arguments as code \(u\) and \(v\) do not mimic the overline nomenclature (\(\overline{}\)) of the exceedance (survival) probabilities.

**Author(s)**

W.H. Asquith

**References**


**See Also**

COP, surCOP, duCOP
Examples

u <- 1 - runif(1); v <- 1 - runif(1) # as exceedance, in order to reinforce the
# change to exceedance instead of nonexceedance that otherwise dominates this package
message("Exceedance probabilities u' and v' are ','. u, '' and ',', v)
coCOP(u,v,cop=PLACKETTCop, para=10) # Positive association Plackett

# computation using manual manipulation to nonexceedance probability
1 - COP(cop=PSP,(1-u),(1-v))
# computation using internal manipulation to nonexceedance probability
coCOP(cop=PSP, u, v)

# Next demonstrate COP + coCOP = unity.
"MOcop.formula" <- function(u,v, para=para, ...) { # Marshall-Olkin copula
alpha <- para[1]; beta <- para[2]; return(min(v*u*(1-alpha), u*v*(1-beta)))
}
"MOcop" <- function(u,v, ...) { asCOP( u, v, f=MOcop.formula, ...) }
u <- 0.2; v <- 0.75; ab <- c(1.5, 0.3)
COP(u,v, cop=MOcop, para=ab) + coCOP(1-u,1-v, cop=MOcop, para=ab) # UNITY

composite1COP  Composition of a Single Symmetric Copula with Two Compositing Parameters

Description

The composition of a single copula (Salvadori et al., 2006, p. 266, prop. C.3) is created by the
following result related to “composition of copulas” in that reference. Suppose $C(u,v)$ is a sym-
metric copula (see COP) with parameters $\Theta$ and $C \neq \Pi$ (for $\Pi$ see P), then a family of generally
asymmetric copulas $C_{\alpha,\beta};\Theta$ with two compositing parameters $0 < \alpha, \beta < 1$, and $\alpha \neq \beta$, which
also includes just the copula $C(u,v)$ as a limiting case for $\alpha = \beta = 0$ and is given by

$$C_{\alpha,\beta}(u,v) = u^\alpha v^\beta \cdot C(u^{1-\alpha}, v^{1-\beta}).$$

The composite1COP function provides the means for permutation asymmetry from a permuta-
sion symmetric copula as also described by Joe (2017, p. 124) but do so in a more general way through
the provision of two and not just one parameter. Joe’s description is supported herein if one of the $\alpha$
or $\beta$ is held at zero. Very loosely, the $\alpha > 0$ kicks probability density down towards the lower right
corner, whereas $\beta > 0$ kicks density up towards the upper left corner. Finally, the composite2COP
function is based on a slightly more general result (see composite2COP for further details of copula
composition).

Usage

composite1COP(u, v, para, ...)
Arguments

- **u**: Nonexceedance probability \( u \) in the \( X \) direction;
- **v**: Nonexceedance probability \( v \) in the \( Y \) direction;
- **para**: A special parameter list (see Note); and
- ... Additional arguments to pass to the copula.

Value

Value(s) for the composited copula are returned.

Note

The following descriptions list in detail the structure and content of the para argument:

- **alpha**: The \( \alpha \) compositing parameter;
- **beta**: The \( \beta \) compositing parameter;
- **cop1**: Function of the copula \( C(u,v) \); and
- **para1**: Vector of parameters \( \Theta_C \) for \( C(u,v) \).

For the para argument, the same nomenclature as used for composite2COP is used with obviously cop2 and para2 dropped for composite1COP. The cop1 and para1 names remain enumerated for composite1COP so that the para argument of the more general composite2COP function could be used directly in composite1COP. Albeit, the second copula and its parameters would not be used. A more complex (extended) composition in composite3COP extends this basic parameter structure.

Author(s)

W.H. Asquith

References


See Also

COP, breveCOP, composite2COP, composite3COP, convexCOP, glueCOP

Examples

```r
## Not run:
alpha <- 0.24; beta <- 0.23; Theta1 <- NA;
# W() does not use a parameter, but show how a parameter would be set if needed.
para <- list(alpha=alpha, beta=beta, cop1=W, para1=Theta1)
t <- composite1COP(0.4, 0.6, para)
```
if( t != W(0.4, 0.6)) message("Not equal as expected")

# Next use this as a chance to check logic flow through the various
# "compositing" operators and their as needed dispatch to COP()
my.para <- list(cop1=GHcop, para1=exp(+1.098612) + 1,
    cop2=PLcop, para2=exp(-1.203973),
    alpha=0.5, beta=0.25,
    kappa=0.1, gamma=0.1,
    weights=c(0.95, 0.05))

# uses cop1/2, para1/2, only weights
nustarCOP(cop=convexCOP, para=my.para) # 0.8570434

# uses cop1/2, para1/2, only alpha
nustarCOP(cop=convex2COP, para=my.para) # 0.2697063

# uses cop1, para1, only alpha / beta
nustarCOP(cop=composite1COP, para=my.para) # 0.5103119

# uses cop1/2, para1/2, only alpha / beta
nustarCOP(cop=composite2COP, para=my.para) # 0.0714571

# uses cop1/2, para1/2, only alpha, beta, kappa, gamma
nustarCOP(cop=composite3COP, para=my.para) # 0.0792634

## End(Not run)

---

**composite2COP**  
*Composition of Two Copulas with Two Compositing Parameters*

**Description**

The *composition of two copulas* (Salvadori *et al.*, 2007, p. 266, prop. C.3) provides for more sophisticated structures of dependence between variables than many single parameter copula can provide. Further, *asymmetrical copulas* are readily obtained from *symmetrical copulas*. Let $A$ and $B$ be copulas with respective parameters $\Theta_A$ and $\Theta_B$, then

$$C_{\alpha,\beta}(u,v) = A(u^\alpha, v^\beta) \cdot B(u^{1-\alpha}, v^{1-\beta}),$$

defines a family of copulas $C_{\alpha,\beta;\Theta_A,\Theta_B}$ with two *compositing parameters* $\alpha, \beta \in [0,1]$. In particular if $\alpha = \beta = 1$, then $C_{1,1} = A$, and if $\alpha = \beta = 0$, then $C_{0,0} = B$. For $\alpha \neq \beta$, the $C_{\alpha,\beta}$ is in general asymmetric that is $C(u,v) \neq C(v,u)$ for some $(u,v) \in I^2$.

It is important to stress that copulas $A_{\Theta_A}$ and $B_{\Theta_B}$ can be of different families and each copula parameterized accordingly by the vector of parameters $\Theta_A$ and $\Theta_B$. This is an interesting feature in the context of building complex structures when pursuing asymmetric measures of dependency such as the $L$-comoments. Do the copulas $A$ and $B$ need be symmetric? The Salvadori reference makes no stated restriction to that effect. Symmetry of the copula $C$ is required for the situation that follows, however.

It is possible to simplify the construction of an asymmetric copula from a symmetric copula by the following. Let $C(u,v)$ be a symmetric copula, $C \neq \Pi$ (for $\Pi$ see $P$). A family of asymmetric
copulas $C_{\alpha,\beta}$ with two composition parameters $0 < \alpha, \beta < 1$, and $\alpha \neq \beta$ that also includes $C(u,v)$ as a limiting case and is given by

$$C_{\alpha,\beta}(u,v) = u^\alpha v^\beta \cdot C(u^{1-\alpha}, v^{1-\beta}).$$

The composite2COP function is based on the more general result given in the former rather than the later mathematical definition to provide additional flexibility. For simpler case of composition involving only one copula, composite1COP is available, and a more complex (extended) composition is available in composite3COP.

Usage

```
composite2COP(u, v, para, ...)
```

Arguments

- **u** Nonexceedance probability $u$ in the $X$ direction;
- **v** Nonexceedance probability $v$ in the $Y$ direction;
- **para** A special parameter list (see Note); and
- **...** Additional arguments to pass to the copulas.

Value

Value(s) for the composited copula is returned.

Note

The following descriptions list in detail the structure and content of the `para` argument:

- **alpha** — The $\alpha$ compositing parameter;
- **beta** — The $\beta$ compositing parameter;
- **cop1** — Function of the first copula $A$;
- **cop2** — Function of the second copula $B$;
- **para1** — Vector of parameters $\Theta_A$ for $A$; and
- **para2** — Vector of parameters $\Theta_B$ for $B$.

The `para` argument of this function also can be passed to composite1COP; albeit, the second copula and its parameters would not be used. A more complex (extended) composition in composite3COP extends this basic parameter structure.

Author(s)

W.H. Asquith

References

The (extended) composition of two copulas (Salvadori et al., 2006, p. 266, prop. C.4) provides for even more sophisticated structures of dependence between variables than two-copula composition in `composite2COP`. Let $A$ and $B$ be copulas with respective parameters $\Theta_A$ and $\Theta_B$, then

\begin{align*}
\text{composite3COP} \quad \text{(Extended) Composition of Two Copulas with Four Compositing Parameters}
\end{align*}

### Examples

```r
alpha <- 0.24; beta <- 0.23; Theta1 <- NA; Theta2 <- NA
# The W() and PSP() copulas do not take parameters, but example shows how the
# parameters would be set should either or both of the copulas require parameters.
para <- list(alpha=alpha, beta=beta, cop1=W, cop2=PSP, para1=Theta1, para2=Theta2)
print(composite2COP(0.4, 0.6, para)) # 0.2779868

# In this example, the N4212cop uses "3" as its parameter value.
para <- list(alpha=alpha, beta=beta, cop1=W, cop2=N4212cop, para1=Theta1, para2=3)
print(composite2COP(0.4, 0.6, para)) # 0.3387506

## Not run:
# This example does a great job of showing a composited copula with a near singularity,
# but with leakage of chance to the upper left. The example is also critical because
# it shows that gridCOP is returning a matrix in the proper orientation relative to
# the level.curvesCOP and simCOP functions. Example is cross-ref'ed from gridCOP() docs.
layout(matrix(1:2,byrow=TRUE))
para <- list(alpha=0.5, beta=0.90, cop1=M, cop2=N4212cop, para1=NA, para2=1.4)
image(gridCOP(cop=composite2COP, para=para, delta=0.01), col=terrain.colors(30),
     xlab="U, NONEXCEEDANCE PROBABILITY", ylab="V, NONEXCEEDANCE PROBABILITY")
D <- simCOP(n=2000, cop=composite2COP, para=para, ploton=FALSE, pch=4, col=4, cex=0.75)
level.curvesCOP(cop=composite2COP, para=para, ploton=FALSE, delt=0.05)
# Theoretical composited copula, level curves, and simulation

emp <- EMPIRgrid(para=D, deluv=0.05) # CPU heavy
image(emp$empcop, col=terrain.colors(30) ) # image orientation is correct!
# Depending on balance between sample size, deluv, delu, and delt, one or more:
# Error in uniroot(func, interval = c(0, 1), u = u, LHS = t, cop = cop, :
# f() values at end points not of opposite sign
# warnings might be triggered. This is particularly true because of the flat derivative
# above the near singularity in this example composited copula.
points(D$U, D$V, pch=4, col=4, cex=0.75)
level.curvesCOP(cop=EMPIRcop, para=D, ploton=FALSE, delu=0.02, delt=0.05)
mtext("Empirical copula from n=2000 simulation")#
## End(Not run)
```
\[ C_{\alpha,\beta,\kappa,\gamma}(u, v) = u^\kappa v^\gamma \cdot A([u^{1-\kappa}]^\alpha, [v^{1-\gamma}]^\beta) \cdot B([u^{1-\kappa}]^{1-\alpha}, [v^{1-\gamma}]^{1-\beta}), \]

defines a family of copulas \( C_{\alpha,\beta,\kappa,\gamma} \) with **four compositing parameters** \( \alpha, \beta, \kappa, \gamma \in (0, 1) \).

It is important to stress that copulas \( A_{\Theta_A} \) and \( B_{\Theta_B} \) can be of different families and each parameterized accordingly by the vectors of parameters \( \Theta_A \) and \( \Theta_B \).

**Usage**

\[ \text{composite3COP}(u, v, \text{para}, \ldots) \]

**Arguments**

- \( u \) Nonexceedance probability \( u \) in \( X \) direction;
- \( v \) Nonexceedance probability \( v \) in \( Y \) direction;
- \( \text{para} \) A special parameter list (see **Note**); and
- \( \ldots \) Additional arguments to pass to \text{composite2COP}.

**Value**

A value for the composited copula is returned.

**Note**

The following descriptions list in detail the structure and content of the \text{para} argument:

- \text{alpha} — The \( \alpha \) compositing parameter;
- \text{beta} — The \( \beta \) compositing parameter;
- \text{kappa} — The \( \kappa \) compositing parameter;
- \text{gamma} — The \( \gamma \) compositing parameter;
- \text{cop1} — Function of the first copula \( A \);
- \text{cop2} — Function of the second copula \( B \);
- \text{para1} — Vector of parameters \( \Theta_A \) for \( A \); and
- \text{para2} — Vector of parameters \( \Theta_B \) for \( B \).

The first example produces two plots. These are extremely informative for many nuances of copula theory. Whereas it is difficult in prose to describe, users are strongly encouraged that once full understanding of connection of red and green between the easier to understand bivariate plot and the plot showing the sections and derivatives of the sections is achieved that much of copula theory will be mastered—get a copy of Nelsen (2006) and (or) Salvadori *et al.* (2007).

**Author(s)**

W.H. Asquith
convex2COP

Convex Combination of Two Copulas

Description

The convex composition of two copulas (Joe, 2014, p. 155) provides for some simple complexity extension between copula families. Let A and B be copulas with respective vectors of parameters \( \Theta_A \) and \( \Theta_B \), then the convex combination of these copulas is

\[
C^\times_C(u, v) = \alpha A(u, v; \Theta_A) - (1 - \alpha) B(u, v; \Theta_B),
\]

where \( 0 \leq \alpha \leq 1 \). The generalization of this function for \( N \) number of copulas is provided by \texttt{convexCOP}.

References


See Also

\texttt{COP}, \texttt{breveCOP}, \texttt{simCOP}, \texttt{composite1COP}, \texttt{composite2COP}, \texttt{convexCOP}, \texttt{glueCOP}, \texttt{simcomposite3COP}

Examples

```r
## Not run:
para <- list(cop1=PLACKETTcop, cop2=N4212cop,
              para1=10*runif(1, min=-5, max=5), para2=runif(1, min=1, max=100),
              alpha=runif(1), beta=runif(1), kappa=runif(1), gamma=runif(1))
txts <- c("Alpha=", round(para$alpha, digits=4),
           "Beta=", round(para$beta, digits=4),
           "Kappa=", round(para$kappa, digits=4),
           "Gamma=", round(para$gamma, digits=4),
           "Theta1=", round(para$para1[1], digits=5),
           "Theta2=", round(para$para2[1], digits=2))
layout(matrix(1:2, byrow=TRUE))
D <- simCOP(n=300, cop=composite3COP, para=para, cex=0.5, col=rgb(0,0,0,0.2), pch=16)
mtext(paste(txts,collapse=""))

f <- round(runif(1), digits=2)
ftxt <- c("Sectionals (thick) and derivatives (thin) at f="f," nonexceedance prob.")
segments(f,0,f,1, lwd=2); segments(0,f,1,f, col=2, lwd=2)
t <- sectionCOP(f,cop=composite3COP,para=para, wrtV=TRUE, dercop=TRUE, col=3)
t <- sectionCOP(f,cop=composite3COP,para=para, wrtV=TRUE, dercop=TRUE, col=2, lwd=4)

mtext(paste(ftxt, collapse=""))#
## End(Not run)
```
convex2COP

Usage

convex2COP(u,v, para, ...)

Arguments

u   Nonexceedance probability $u$ in the $X$ direction;

v   Nonexceedance probability $v$ in the $Y$ direction;

para  A special parameter list (see Note); and

...  Additional arguments to pass to the copula.

Value

Value(s) for the convex combination copula is returned.

Note

The following descriptions list in detail the structure and content of the para argument:

alpha — The $\alpha$ compositing parameter;

cop1 — Function of the first copula $A$;

cop2 — Function of the second copula $B$;

para1 — Vector of parameters $\Theta_A$ for $A$; and

para2 — Vector of parameters $\Theta_B$ for $B$.

Author(s)

W.H. Asquith

References


See Also

COP, breveCOP, convexCOP, composite1COP, composite2COP, composite3COP, FRECHETcop, glueCOP

Examples

para <- list(alpha=0.24, cop1=FRECHETcop, para1=c(0.4, 0.56),
             cop2=PSP, para2=NA)
convex2COP(0.87,0.35, para=para) # 0.3188711

## Not run:
# Suppose we have a target Kendall Tau of 1/3 and a Gumbel-Hougaard copula seems
# attractive but the GH has just too much upper tail dependency for comfort. We
# think from data analysis that an upper tail dependency that is weaker and near
# 2/10 is the better. Let us convex mix in a Plackett copula and optimize.
TargetTau <- tauCOP(cop=GHcop, para=1.5) # 1/3 (Kendall Tau)
taildepCOP( cop=GHcop, para=1.5, plot = TRUE)$lambdaU # 0.4126
TargetUpperTailDep <- 2/10
# **Serious CPU time pending for this example**
par <- c(-.10, 4.65) # Initial guess but the first parameter is in standard
# normal for optim() to keep us in the [0,1] domain when converted to probability.
# The guesses of -0.10 (standard deviation) for the convex parameter and 4.65 for
# the Plackett are based on a much longer search times as setup for this problem.
# The simplex for optim() is going to be close to the solution on startup.
"afunc" <- function(par) {
  para <- list(alpha=pnorm(par[1]), cop1=GHcop, para1=1.5,
                cop2=PLACKETTcop, para2=par[2])
  tau <- tauCOP(cop=convex2COP, para=para)
  taildep <- taildepCOP(cop=convex2COP, para=para, plot = FALSE)$lambdaU
  err <- sqrt((TargetTau - tau)^2 + (TargetUpperTailDep - taildep)^2)
  print(c(pnorm(par[1]), par[2], tau, taildep, err))
  return(err)
}
mysolution <- optim(par, afunc, control=list(abstol=1E-4))

para <- list(alpha=.4846902, cop1=GHcop, para1=1.5,
              cop2=PLACKETTcop, para2=4.711464)
UV <- simCOP(n=2500, cop=convex2COP, para=para, snv=TRUE) 
## End(Not run)

convexCOP

Convex Combination of an Arbitrary Number of Copulas

**Description**

The convex composition of \( N \) number of copulas (Salvadori et al., p. 132, 2007) provides for complexity extension between copula families. Let \( C_i \) be a copula with respective vector of parameters \( \Theta_i \), then the convex combination of these copulas is

\[
C_n^\omega(u,v) = \sum_{i=1}^{N} \omega_i C_i(u,v; \Theta_i),
\]

where \( \sum_{i=1}^{N} \omega_i = 1 \) for \( N \) number of copulas. The weights \( \omega \) are silently treated as \( 1/N \) if the weights element is absent in the \( \textbf{R} \) list argument \( \text{para} \).

**Usage**

\[
\text{convexCOP}(u,v, \text{para}, \ldots)
\]

**Arguments**

- \( u \) Nonexceedance probability \( u \) in the \( X \) direction;
- \( v \) Nonexceedance probability \( v \) in the \( Y \) direction;
- \( \text{para} \) A special parameter list (see \textbf{Note}); and
- \( \ldots \) Additional arguments to pass to the copula.
convexCOP

Value

Value(s) for the convex combination copula is returned.

Note

The following descriptions list in detail the structure and content of the para argument but please reference the Examples to see the i notation:

copi — The i th copula;
parai — Vector of parameters \( \Theta_i \); and
weights — Optional vector of weights whose sum will be rescaled to unity; default is \( 1/N \) for each weight.

Author(s)

W.H. Asquith

References


See Also

COP, breveCOP, convex2COP, composite1COP, composite2COP, composite3COP, glueCOP

Examples

# The copulas and parameters are named by sequence number appended to cop and para.
para1 <- list(cop1=GHcop, cop2=PLcop, para1=8, para2=.03, weights=c(.8,.2))
para2 <- list(cop1=GHcop, cop2=PLcop, para1=8, para2=.03, alpha=0.8)
H <- convexCOP( 0.6,0.4, para=para1)
G <- convex2COP(0.6,0.4, para=para2)
if( abs(H-G) <= 1e-6 ) message("They are equal.")

## Not run:
# A convex combination of three copulas. A GHcop with strong positive association and
# a Plackett with strong negative association, and independence. The weights favor the
# GHcop but a little outlier and expansive spread is superimposed on the core trend.
para <- list(cop1=GHcop, cop2=PLcop, cop3=P,
            para1=8, para2=.03, para3=NA, weights=c(40,7,10))
UV <- simCOP(1000, cop=convexCOP, para=para, lwd=0.8) #
## End(Not run)
The Copula

Description

Compute the copula or joint distribution function through a copula as shown by Nelsen (2006, p. 18) is the joint probability

\[ \Pr[U \leq u, V \leq v] = C(u, v). \]

The copula is an expression of the joint probability that both \( U \leq u \) and \( V \leq v \).

A copula is a type of dependence function that permits straightforward characterization of dependence from independence. Joe (2014, p. 8) comments that “copula families are usually given as cdfs [cumulative distribution functions].” A radially symmetric or permutation symmetric copula is one such that \( C(u, v) = C(v, u) \) otherwise the copula is asymmetric.

The copula inversions \( t = C(u=U, v) \) or \( t = C(u, v=V) \) for a given \( t \) and \( U \) or \( V \) are provided by \( \text{COPinv} \) and \( \text{COPinv2} \), respectively. A copula exists in the domain of the unit square \( \mathbb{I}^2 = [0, 1] \times [0, 1] \) and is a grounded function meaning that \( C(u, 0) = 0 = C(0, v) \) and thus \( C(0, 0) = 0 \), and other properties of a copula are that

\[ C(u, 1) = u \text{ and } C(1, v) = v \text{ and } C(1, 1) = 1. \]

Copulas can be combined with each other (\( \text{convexCOP}, \text{convex2COP}, \text{composite1COP}, \text{composite2COP}, \text{composite3COP}, \text{and glueCOP} \)) to form more complex and sophisticated dependence structures. Also copula multiplication—a special product of two copulas—yields another copula (see \( \text{prod2COP} \)).

Perhaps the one of the more useful features of this function is that in practical applications it can be used to take a copula formula and reflect or rotated it in fashions to attain association structures that the native definition of the copula can not acquire. The terminal demonstration in the Examples demonstrates this for the Raftery copula (\( \text{RFcop} \)).

Usage

\[
\text{COP}(u, v, \text{cop=NULL, para=NULL, reflect=c("cop", "surv", "acute", "grave", "1", "2", "3", "4"), ...})
\]
Arguments

- **u**: Nonexceedance probability $u$ in the $X$ direction;
- **v**: Nonexceedance probability $v$ in the $Y$ direction;
- **cop**: A copula function with vectorization as in `asCOP`;
- **para**: Vector of parameters or other data structures, if needed, to pass to the copula;
- **reflect**: The reflection of the copula form (see [Note](#note)) and the default "cop" or "1" is the usual copula definition (also see `simCOPmicro`). The numbered values correspond, respectively, to the named values; and
- **...**: Additional arguments to pass to the copula.

Value

Value(s) for the copula are returned.

Note

**REFLECTIONS OF VARIABLES (ROTATIONS OF THE COPULA)**—The copula of $(1 - U, 1 - V)$ is the survival copula ($\hat{C}(u, v); \text{surCOP}$) and is defined as

$$
\Pr[U > u, V > v] = \hat{C}(u, v) = u + v - 1 + C(1 - u, 1 - v) \rightarrow \text{"surv"},
$$

whereas, following the notation of Joe (2014, p. 271), the copula of $(1 - U, V)$ is defined as

$$
\Pr[U > u, V \leq v] = \check{C}(u, v) = v - C(1 - u, v) \rightarrow \text{"acute"}, \text{and}
$$

the copula of $(U, 1 - V)$ is defined as

$$
\Pr[U \leq u, V > v] = \tilde{C}(u, v) = u - C(u, 1 - v) \rightarrow \text{"grave"}.
$$

Here it is useful to stress the probability aspects that change with the reflections, but this section ends with the reflections themselves being graphically highlighted. The Examples stress simple variations on the probability aspects.

To clarify the seemingly clunky nomenclature—Joe (2014) does not provide “names” for $\check{C}(u, v)$ or $\tilde{C}(u, v)$—the following guidance is informative:

1. "surv" or $\hat{C}(u, v)$ is a reflection of $U$ and $V$ on the horizontal and vertical axes, respectively,
2. "acute" or $\check{C}(u, v)$ is a reflection of $U$ on the horizontal axis, and
3. "grave" or $\tilde{C}(u, v)$ is a reflection of $V$ on the vertical axis.

The names "acute" and "grave" match those used in the Rd-format math typesetting instructions. Users are directed to the documentation of `simCOPmicro` for further discussion because the `COP` function is expected to be an early entry point for new users interested in the copBasic API.

For the copBasic package and in order to keep some logic brief and code accessible for teaching and applied circumstances, reflections of copulas using analogs to the reflect argument are only natively supported in the `COP` and `simCOPmicro` functions. The interfaces of copBasic should already be flexible enough for users to adapt and (or) specially name reflections of copulas for deployment. A caveat is that some individual copula implementations might have some self-supporting infrastructure. The reflection can also be set within the para argument when it is a list (see Examples).
An example is warranted. Although the Gumbel–Hougaard copula (GHcop) can be reflected by COP and simCOPmicro and testing is made in the Note section of simCOPmicro, it is suggested that a user generally requiring say a horizontal reflection ru (or vertical reflection rv) of the Gumbel–Hougaard copula write a function named perhaps ruGHcop (or rvGHcop).

Such functions, consistent with the mathematics at the beginning of this Note, can be used throughout functions of copBasic using the cop arguments. The author (Asquith) eschews implementing what is perceived as too much flexibility and overhead for the package to support the three reflection permutations universally across all copula functions of the package. This being said, COP can take an R list for the para argument for rotation/reflection:

```r
set.seed(14)
UV3 <- simCOP(20, cop=COP, pch=16, col=3,
       para=list(cop=GLcop, para=pi+1, reflect="3"))
set.seed(14)
UV2 <- simCOP(20, cop=COP, pch=16, col=4, ploton=FALSE,
       para=list(cop=GLcop, para=pi+1, reflect="2"))
arrows(x0=UV3[,1], y0=UV3[,2], x=UV2[,1], y=UV2[,2])
```

and this type of interface is similar to composite1COP as the following rotation and then asymmetric construction shows:

```r
UV <- simCOP(1000, cop=composite1COP,
       para=list(cop1=COP,
                 para1=c(cop=GHcop, para=pi+1, reflect="4"),
                 alpha=0.1, beta=0.3))
```

Author(s)

W.H. Asquith

References


See Also

coCOP, duCOP, surCOP, surfuncCOP

Examples

```r
u <- runif(1); v <- runif(1)
COP(cop=W,u,v); COP(cop=P,u,v); COP(cop=M,u,v); COP(cop=PSP,u,v)
FF <- 0.75 # 75th percentile, nonexceedance
GG <- 0.20 # 25th percentile, nonexceedance
bF <- 1 - FF; bG <- 1 - GG # exceedance
# What is the probability that both X and Y are less than
# 75th and 20th percentiles, respectively?
COP(cop=P, FF, GG) # 0.15
```
# What is the probability that both X and Y are greater than
# 75th and 20th percentiles, respectively?
surCOP(cop=P, bF, bG) # 0.20
# What is the probability that either X or Y are less than
# the 75th and 20th percentiles, respectively?
duCOP(cop=P, FF, GG) # 0.8
# What is the probability that either X or Y are greater than
# the 75th and 20th percentiles, respectively?
coCOP(cop=P, bF, bG) # 0.85

# Repeat for the PSP copula:
# What is the probability that both X and Y are less than
# 75th and 20th percentiles, respectively?
COP(cop=PSP, FF, GG) # 0.1875
# What is the probability that both X and Y are greater than
# 75th and 20th percentiles, respectively?
surCOP(cop=PSP, bF, bG) # 0.2375
# What is the probability that either X or Y are less than
# the 75th and 20th percentiles, respectively?
duCOP(cop=PSP, FF, GG) # 0.7625
# What is the probability that either X or Y are greater than
# the 75th and 20th percentiles, respectively?
coCOP(cop=PSP, bF, bG) # 0.8125

# Both of these summations equal unity
COP(cop=PSP, FF, GG) + coCOP(cop=PSP, bF, bG) # 1
surCOP(cop=PSP, bF, bG) + duCOP(cop=PSP, FF, GG) # 1

FF <- 0.99 # 99th percentile, nonexceedance
GG <- 0.50 # 50th percentile, nonexceedance
bF <- 1 - FF # nonexceedance
bG <- 1 - GG # nonexceedance
# What is the probability that both X and Y are less than
# 99th and 50th percentiles, respectively?
COP(cop=P, FF, GG) # 0.495
# What is the probability that both X and Y are greater than
# 99th and 50th percentiles, respectively?
surCOP(cop=P, bF, bG) # 0.005
# What is the probability that either X or Y are less than
# the 99th and 50th percentiles, respectively?
duCOP(cop=P, FF, GG) # 0.995
# What is the probability that either X or Y are greater than
# the 99th and 50th percentiles, respectively?
coCOP(cop=P, bF, bG) # 0.505

# Not run:
# MAJOR EXAMPLE FOR QUICKLY MODIFYING INHERENT ASSOCIATION STRUCTURES
p <- 0.5 # Reasonable strong positive association for the Raftery copula
"RFcop1" <- function(u,v, para) COP(u,v, cop=RFcop, para=para, reflect="1")
"RFcop2" <- function(u,v, para) COP(u,v, cop=RFcop, para=para, reflect="2")
"RFcop3" <- function(u,v, para) COP(u,v, cop=RFcop, para=para, reflect="3")
"RFcop4" <- function(u,v, para) COP(u,v, cop=RFcop, para=para, reflect="4")
d <- 0.01 # Just to speed up the density plots a bit
densityCOPplot(RFcop1, para=p, contour.col=1, deluv=d) # the Raftery in the literature
densityCOPplot(RFcop2, para=p, contour.col=1, deluv=d, ploton=FALSE)
densityCOPplot(RFcop3, para=p, contour.col=1, deluv=d, ploton=FALSE)
densityCOPplot(RFcop4, para=p, contour.col=1, deluv=d, ploton=FALSE)
# Now some text into the converging tail to show the reflection used.
text(-2,-2, "reflect=1", col=2); text(+2,+2, "reflect=2", col=2)
text(+2,-2, "reflect=3", col=2); text(-2,+2, "reflect=4", col=2)

# To show how the reflection can be alternatively specified and avoid in this case
# making four Raftery functions, pass by a list para argument. Also, demonstrate
# that cop1 --> cop and para1 --> para are the same in use of the function. This
# provides some nomenclature parallel to the other compositing functions.
densityCOPplot(COP, para=list(reflect=1, cop1=RFcop, para=p ), deluv=d,
contour.col=1, drawlabels=FALSE)
densityCOPplot(COP, para=list(reflect=2, cop1= RFcop, para=p), deluv=d,
contour.col=2, drawlabels=FALSE, ploton=FALSE)
densityCOPplot(COP, para=list(reflect=3, cop1=RFcop, para=p), deluv=d,
contour.col=3, drawlabels=FALSE, ploton=FALSE)
densityCOPplot(COP, para=list(reflect=4, cop1=RFcop, para=p), deluv=d,
contour.col=4, drawlabels=FALSE, ploton=FALSE) #
## End(Not run)

## Not run:
# Similar example to previous, but COP() can handle the reflection within a
# parameter list ,and the reflect, being numeric here, is converted to
# character internally.
T12 <- CLcop(tau=0.67)$para # Kendall Tau of 0.67
T12 <- list(cop=CLcop, para=T12, reflect=2) # reflected to upper tail dependency
UV <- simCOP(n=1000, cop=COP, para=T12) #
## End(Not run)

copBasic.fitpara A Single or Multi-Parameter Optimization Engine (Beta Version)

Description

An example of a general implementation of a parameter optimization scheme using core features of the copBasic package. Because of the general complexity of the objectives for this function, the interface shown here is considered an “beta version” and nomenclature is subject to possibly sweeping changes in the future.

Usage

copBasic.fitpara.beta(uv=NULL, popstat=NULL, statf=NULL, cop=NULL,
paradim=1, interval=NULL, par.init=NULL, ...)

Arguments

uv An R two column matrix or data.frame of a sample of nonexceedance probabilities u and v;
copBasic.fitpara

popstat

The population statistic(s) that will be used if `uv` is `NULL`;

statf

A function responsible at the minimum for computation of the theoretical values of the population statistic(s) that the optimization will revolve around: This function, if supporting an `as.sample` interface (e.g., `hoefCOP`) and if `uv` has been provided, will be dispatched to compute the population statistic(s);

cop

A copula function that is passed along to `statf` though of course the `statf` function can decide whether to use this argument or not;

paradim

The dimension of the parameters. In reality, the default triggers uni-dimensional root solving using the `uniroot()` function in `R` or otherwise the `optim()` function in `R` is used for multi-dimensional minimization with subtle changes in setup (see source code). Alternative logic could have been used but it is felt that this is the most logical for future adaptations;

interval

The interval argument by the same name for the `uniroot()` function;

par.init

The initial parameter vector for the `optim()` function; and

... Additional arguments to pass.

Value

A vector of the values for the parameter variable is returned

Note

**One-Parameter Optimization**—A demonstration of one-dimensional parameter optimization using the Gini Gamma (`giniCOP`), which is a measure of bivariate association. There is no native support for Gini Gamma (and most of the other measures of association) in regards to being a parameter estimator at the copula function interface level in `copBasic`. (A converse example is one provided internally by the `GHcop` copula.)

```r
set.seed(24); n <- 230 # sample size to draw from Galambos copula but a # different copula was chosen for the fitting.
sampleUV <- simCOP(n=n, cop=GLcop, para=1.5) # a random sample
para <- copBasic.fitpara.beta(uv=sampleUV, statf=giniCOP,
   interval=c(.1,200), cop=HRcop) # 1.959521
```

**Three-Parameter Optimization**—A demonstration of multi-dimensional parameter optimization using the Gini Gamma (`giniCOP`), Nu-Skew (`nuskewCOP`), and Nu-Star (`nustarCOP`). This is substantially more complicated to implement. The Hüsler–Reiss copula (`HRcop`) is chosen both as part of the sample simulation for the study as well as the copula as part of the modeling. Using composition by the `composite1COP`, first establish a parent three parameter copula and simulate from it to create a bivariate sample in `sampleUV` that will be used for demonstration. A standard normal variate graphic of the simulation is generated by `simCOP` as well—later, additional results will be superimposed.

```r
n <- 610; set.seed(1999) # Seed value will be used again (see below)
pop.para <- list(cop1=HRcop, para1=4, alpha=.14, beta=0.35)
sampleUV <- simCOP(n=n, cop=composite1COP, para=pop.para, col=3, snv=TRUE)
```
A custom objective function `objfunc` to serve as the `statf` for the `copBasic.fitpara.beta` call. The objective function has the `as.sample` interface (e.g. `giniCOP`) implemented for sample estimation. The most subtle feature of function presumably is the use of the `pnorm()` function in R for the \( \alpha \) and \( \beta \) parameters to keep each parameter in the range \( \alpha, \beta \in (0, 1) \). Another subtly, which affects the choice of other copulas from `HRcop`, is how the parameter range of \( \Theta \) (the `para[1]` variable) is controlled—here the parameter remains untransformed but the lower domain edge is truncated by the return of infinity (`return(Inf)`). The `getstat` argument is only for short circuiting the objective so that `objfunc` can be used to compute the three statistics after the optimal parameters are found.

```r
"objfunc" <- function(para, stat=NA, as.sample=FALSE, getstat=FALSE, ...) {
  if(as.sample) {
    return(c( giniCOP(para=para, as.sample=TRUE),
              nuskewCOP(para=para, as.sample=TRUE),
              nustarCOP(para=para, as.sample=TRUE)))
  }
  para[1] <- ifelse(pmin(para[1] < 0, return(Inf), para[1]) # edge check
  para[2:3] <- pnorm(para[2:3]) # detransform
  para <- list(cop1=HRcop, para1=para[1], alpha=para[2], beta=para[3])
  hp <- c( giniCOP(composite1COP, para),
           nuskewCOP(composite1COP, para),
           nustarCOP(composite1COP, para))
  if(getstat) return(hp) # short circuit to get the statistics out
  dv <- stat; dv[dv == 0] <- 1 # changing dv "adapts" the error to
  return(sqrt(sum(((stat-hp)/dv)^2))) # trap division by zero
}
```

The parameter estimation proceeds in the following code. The sample statistics (or `target.stats`) are computed and subsequently passed to the optimization using the `popstat` argument. Notice also the use of the `qnorm()` function in R to transform the initial guess \( \alpha = \beta = 1/2 \) into a domain more easily handled by the optimizer (`optim()` function in R). The transformed optimal parameters are computed, and then the values of the three statistics for the fit are computed. Lastly, a `copBasic` parameter object `fit.para` is created, which can be used for later comparisons.

```r
txt <- c("GiniGamma", "NuSkew", "NuStar")
target.stats <- objfunc(sampleUV, as.sample=TRUE); names(target.stats) <- txt
raw.fit.para <- copBasic.fitpara.beta(popstat=target.stats, statf=objfunc,
                                       par.init=c(1, qnorm(0.5), qnorm(0.5)), cop=composite1COP, paradim=3)
fit.stats <- objfunc(raw.fit.para, getstat=TRUE); names(fit.stats) <- txt
fit.para <- list(cop1=HRcop, para1=raw.fit.para[1],
                 alpha=pnorm(raw.fit.para[2]), beta=pnorm(raw.fit.para[3]))
```

The optimization is completed. It is informative to see what the simulation of the fitted copula looks like. Note: this particular example suffers from identifiability problems between the parameters—meaning that local minima exist or that satisfactory solutions using different parameters than used to generate the random sample can occur. The same seed is used so that one-to-one comparison of points can visually be made with the `simCOP` function call.

```r
set.seed(1999) # This value will be used again (see below)
```
The visual comparison is qualitative. The tabular comparison of the sample statistics to those of the fitted model shows that perhaps an acceptable local minima has been attained in terms of “fit” but the subsequent comparison of the parameters of the population used to generate the sample and those of the fitted model seemingly depart substantially in the $\alpha \rightarrow 0$ parameter of the copula composition. The tail dependency parameters are similar, but further goodness-of-fit check is not made.

<table>
<thead>
<tr>
<th></th>
<th>GiniGamma</th>
<th>NuSkew</th>
<th>NuStar</th>
</tr>
</thead>
<tbody>
<tr>
<td>print(target.stats)</td>
<td>0.5219027</td>
<td>-0.1940361</td>
<td>0.6108319</td>
</tr>
<tr>
<td>print(fit.stats)</td>
<td>0.5182858</td>
<td>-0.1938848</td>
<td>0.6159566</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Alpha</th>
<th>Beta</th>
</tr>
</thead>
<tbody>
<tr>
<td>print(ls.str(pop.para))</td>
<td>4.00</td>
<td>0.14</td>
</tr>
<tr>
<td>print(ls.str(fit.para))</td>
<td>11.2</td>
<td>0.187</td>
</tr>
</tbody>
</table>

# Tail Dependency Parameters

<table>
<thead>
<tr>
<th>cop=composite1COP, para=pop.para)</th>
<th>lower=0 : upper=0.5838</th>
</tr>
</thead>
<tbody>
<tr>
<td>taildepCOP(cop=composite1COP, para=fit.para)</td>
<td>lower=0 : upper=0.5714(est.)</td>
</tr>
</tbody>
</table>

The demonstration ends with the comparison of the two asymmetrical density contours.

densityCOPplot(cop=composite1COP, para=pop.para, contour.col=3)
densityCOPplot(cop=composite1COP, para=fit.para, contour.col=2, ploton=FALSE)

**Author(s)**

W.H. Asquith

**Examples**

# See the Note section

---

**Description**

Compute the inverse of a copula for $V$ with respect to $U$ given $t$ or

$$t = C(u=U, v) \rightarrow C^{-1}(u=U, t) = v,$$

and solving for $v$. Nelsen (2006, p. 12) does not so name this function as an “inverse.” The COPinv function is internally used by `level.curvesCOP` and `level.curvesCOP2`. A common misapplication that will puzzle the user (including the developer after long breaks from package use) is that the following call and error message are often seen:
COPinv(.2, .25, cop=PSP)
Error in uniroot(func, interval = c(lo, 1), u = u, LHS = t, cop = cop, :
  f() values at end points not of opposite sign
[1] NA

This is a harmless error in the sense that COPinv is functioning properly. One can not invert a copula for \( u < t \) and for \( u = t \) the \( v = 1 \) because of fundamental copula properties. Also because of fundamental theory being involved, the author perceives that it is better to not silently trap the error but let errors cascade up the chain in a user’s application. This might represent suboptimal design and suggestions for improvement are invited.

Usage

COPinv(cop=NULL, u, t, para=NULL, ...)

Arguments

cop A copula function;
u Nonexceedance probability \( u \) in the \( X \) direction;
t Nonexceedance probability level \( t \);
para Vector of parameters or other data structures, if needed, to pass to the copula; and
... Additional arguments to pass.

Value

Value(s) for \( v \) are returned.

Author(s)

W.H. Asquith

References


See Also

COP, COPinv2, level.curvesCOP, level.curvesCOP2

Examples

COPinv(cop=N4212cop, para=2, u=0.5, t=0.2)
The Inverse of a Copula for U with respect to V

Description

Compute the inverse of a copula for U with respect to V given t or

\[ t = C(u, v=V) \rightarrow C^{-1}(v=V, t) = u, \]

and solving for u. Nelsen (2006, p. 12) does not so name this function as an “inverse.” The \texttt{COPinv2} function is internally used by \texttt{level.curvesCOP2}. A common misapplication that will puzzle the user (including the developer after long breaks from package use) is that the following call and error message are often seen:

\begin{verbatim}
COPinv2(.2, .25, cop=PSP)
Error in uniroot(func, interval = c(lo, 1), u = u, LHS = t, cop = cop, : f() values at end points not of opposite sign
[1] NA
\end{verbatim}

This is a harmless error in the sense that \texttt{COPinv2} is functioning properly. One can not invert a copula for \( v < t \) and for \( v = t \) the \( u = 1 \) because of fundamental copula properties. Also because of fundamental theory being involved, the author perceives that it is better to not silently trap the error but let errors cascade up the chain in a user’s application. This might represent suboptimal design and suggestions for improvement are invited.

Usage

\texttt{COPinv2(cop=NULL, v, t, para=NULL, ...)}

Arguments

- \texttt{cop} A copula function;
- \texttt{v} Nonexceedance probability \( v \) in the \( Y \) direction;
- \texttt{t} Nonexceedance probability in \( t \);
- \texttt{para} Vector of parameters or other data structures, if needed, to pass to the copula; and
- \texttt{...} Additional arguments to pass to the copula.

Value

Value(s) for \( u \) are returned.

Author(s)

W.H. Asquith
densityCOP

References

See Also
COP, COPinv, level.curvesCOP, level.curvesCOP2

Examples
# See those for COPinv as they are the same by analogy.

densityCOP	Density of a Copula

Description
Numerically estimate the copula density for a sequence of \( u \) and \( v \) probabilities for which each sequence has equal steps that are equal to \( \Delta(uv) \). The density \( c(u,v) \) of a copula \( C(u,v) \) is numerically estimated by

\[
c(u,v) = \left[ C(u_2,v_2) - C(u_2,v_1) - C(u_1,v_2) + C(u_1,v_1) \right] / \left[ \Delta(uv) \times \Delta(uv) \right],
\]

where \( c(u,v) \geq 0 \) (see Nelsen, 2006, p. 10; densityCOPplot). The joint density can be defined by the copula density for continuous variables and is the ratio of the joint density function \( f(x,y) \) for random variables \( X \) and \( Y \) to the product of the marginal densities \( f_X(x) \) and \( f_Y(y) \):

\[
c(F_X(x), F_Y(y)) = \frac{f(x,y)}{f_X(x)f_Y(y)},
\]

where \( F_X(x) \) and \( F_Y(y) \) are the respective cumulative distribution functions of \( X \) and \( Y \), and lastly \( u = F_X(x) \) and \( v = F_Y(y) \).

Usage

densityCOP(u,v, cop=NULL, para=NULL, deluv=.Machine$double.eps^0.25, truncate.at.zero=TRUE, the.zero=0, sumlogs=FALSE, ...)

Arguments

- \( u \) Nonexceedance probability \( u \) in the \( X \) direction;
- \( v \) Nonexceedance probability \( v \) in the \( Y \) direction;
- \( \text{cop} \) A copula function;
- \( \text{para} \) Vector of parameters or other data structure, if needed, to pass to the copula;
- \( \text{deluv} \) The change in the sequences \( \{u,v\} \mapsto \delta, \ldots, 1 - \delta; \delta = \Delta(uv) \) probabilities;
A density must be \( c(u, v) \geq 0 \), but because this computation is based on a rectangular approximation and not analytical, there exists a possibility that very small rectangles could result in numerical values in \( \mathbb{R} \) that are less than zero. This usually can be blamed on rounding. This logical if \( \text{TRUE} \) truncated computed densities to zero, and the default assumes that the user is providing a proper copula. A \( \text{FALSE} \) value is used by the function \texttt{isfuncCOP};

The value for “the zero” where a small number might be useful for pseudo-maximum likelihood estimation using \texttt{sumlogs};

Return the \( \sum \log c(u, v; \Theta) \) where \( \Theta \) are the parameters in \texttt{para} and this feature is provided for \texttt{mleCOP}; and

Additional arguments to pass to the copula function.

Value

Value(s) for \( c(u, v) \) are returned.

Note

The \texttt{copBasic} package does not currently have copula densities as analytical solutions implemented. This is because initial design decisions were entirely on cumulative distribution function (CDF) representations of the copula.

Author(s)

W.H. Asquith

References


See Also

\texttt{simCOP, densityCOPplot, kullCOP, mleCOP}

Examples

```r
## Not run:
# Joe (2014, p. 164) shows the closed form copula density for the Plackett.
"dPLACKETTcop" <- function(u,v,para) {
  uv <- u*v; upv <- u + v; eta <- para - 1
  A <- para*(1+eta*(upv - 2*uv)); B <- ((1+eta*upv)^2 - 4*para*eta*uv)^(3/2)
  return(A/B)
}
dPLACKETTcop(0.32, 0.74, para=1.3) # 0.9557124
densityCOP( 0.32, 0.74, cop=PLcop, para=1.3) # 0.9557153
## End(Not run)
## Not run:
```
# Joe (2014, p. 165) shows the corner densities of the Plackett as Theta.  
# copBasic uses numerical density estimation and not analytical formula.  
eps <- .Machine$double.eps  
densityCOP(0, 0, cop=PLcop, para=4) # 3.997073 (default eps^0.25)  
densityCOP(1, 1, cop=PLcop, para=4) # 3.997073 (default eps^0.25)  
densityCOP(1, 1, cop=PLcop, para=4, deluv=eps) # 0 (silent failure)  
densityCOP(1, 1, cop=PLcop, para=4, deluv=eps^0.5) # 4.5  
densityCOP(1, 1, cop=PLcop, para=4, deluv=eps^0.4) # 4.002069  
densityCOP(1, 1, cop=PLcop, para=4, deluv=eps^0.3) # 3.999513  
# So, we see that the small slicing does have an effect, the default of 0.25 is  
# intended for general application by being away enough from machine limits.  
## End(Not run)  
## Not run:  
# Joe (2014, p. 170) shows the closed form copula density for "Bivariate Joe/B5" copula  
"dJOEB5cop" <- function(u, v, para) {  
  up <- (1-u)^para; vp <- (1-v)^para; eta <- para - 1  
  A <- (up + vp - up*vp); B <- (1-u)^eta * (1-v)^eta; C <- (eta + A)  
  return(A^(-2 + 1/para) * B * C)  
}  
densityCOP(0.32, 0.74, cop=JOcopB5, para=1.3) # 0.9410653  
dJOEB5cop( 0.32, 0.74, para=1.3) # 0.9410973  
## End(Not run)  

---  

**densityCOPplot**  
Contour Density Plot of a Copula  

**Description**  

Generate a contour density plot after the advocation of Joe (2014, pp. 9–15). Such graphics are plots of scaled copula densities \((c^*(u, v), \text{bivariate herein})\) that are copula densities scaled to the standard normal distribution \(\sim \text{N}(0,1)\) margins. Joe (2014) repeatedly emphasizes such plots in contrast to the uniform distribution \(\sim \text{U}(0,1)\) margins. Nelsen (2006) does not discuss such scaling but seemingly Nelsen’s objectives for his book were somewhat different.  
The density of copula \(C(u, v)\) is numerically estimated by  
\[
c(u, v) = \frac{[C(u_2, v_2) - C(u_2, v_1) - C(u_1, v_2) + C(u_1, v_1)]}{\Delta(uv) \times \Delta(uv)},
\]
where \(c(u, v) \geq 0\) (see Nelsen, 2006, p. 10; **densityCOP**). Given a numerically estimated quantity \(c^*(u, v) = c(u, v) \times \phi^{-1}(u) \times \phi^{-1}(v)\) for copula density \(c(u, v)\), a grid of the \(c^*(u, v)\) values can be contoured by the `contour()` function in R. The density function of the N(0,1) is \(\phi(z)\) for standard normal variate \(z\) and the quantile function of the N(0,1) is \(\Phi^{-1}(t)\) for nonexceedance probability \(t\).  
A grid (matrix) of \(c(u, v)\) or \(c^*(u, v)\) is defined for sequence of \(u\) and \(v\) probabilities for which each sequence has equal steps that are equal to \(\Delta(uv)\). This function has as focus on plotting of the contour lines of \(c^*(u, v)\) but the R matrix of either \(c(u, v)\) or \(c^*(u, v)\) can be requested on return. For either matrix, the `colnames()` and `rownames()` (the R functions) are set equal to the sequence of \(u\) and \(v\), respectively. Neither the column or row names are set to the standard normal variates for the matrix of \(c^*(u, v)\), the names remain in terms of nonexceedance probability.
For plotting and other uses of normal scores of data, Joe (2014, p. 245) advocates that one should use the plotting position formula \( u_i = (i - 1/2)/n \) (Hazen plotting position) for normal scores \( z_i = \Phi^{-1}(u_i) \) in preference to \( i/(n + 1) \) (Weibull plotting position) because \( n^{-1} \sum_{i=1}^n z_i^2 \) is closer to unity. Other examples of Joe’s advocation for the Hazen plotting positions are available (Joe, 2014, pp. 9, 17, 245, 247–248).

Usage

```r
densityCOPplot(cop=NULL, para=NULL, deluv=0.002,
getmatrix=c("none", "cdenzz", "cden"), n=0,
ploton=TRUE, snv=TRUE, origins=TRUE,
contour.col=1, contour.lwd=1.5, ...)
```

Arguments

- **cop**: A copula function;
- **para**: Vector of parameters or other data structure, if needed, to pass to the copula;
- **deluv**: The change in the sequences \( \{u, v\} \mapsto \delta, \ldots, 1 - \delta; \delta = \Delta(uv) \) probabilities;
- **getmatrix**: A trigger on whether the density matrix is to be returned. The option cdenzz returns the density scaled by the two standard normal densities \( c^\star(u, v) \), whereas the option cden returns simply the copula density \( c(u, v) \);
- **ploton**: A logical to toggle on the plot;
- **snv**: A logical to toggle standard normal variates for the axes;
- **origins**: A logical to plot the origin lines, if and only if snv is true;
- **contour.col**: The color of the contour lines, which corresponds to the col argument of the contour function in R;
- **contour.lwd**: The width of the contour lines, which corresponds to the lwd argument of the contour function in R;
- **n**: An optional sample size for which simulation of this many values from the copula will be made by simCOP and drawn; and
- **...**: Additional arguments to pass to the copula function and to the contour function in R (e.g. to turn off labeling of contours add drawlabels=FALSE).

Value

This is a high-level function used for its side effects; an R matrix can be triggered, however, as a returned value.

Note

Joe (2014, p. 244) says “if [density] plots show multimodality, then multivariate techniques of mixture models, cluster analysis[,] and nonparametric functional data analysis might be more appropriate” than relatively straightforward parametric copula models.

Author(s)

W.H. Asquith
DensityCopplot

References


See Also

simCOP, densityCOP

Examples

```r
## Not run:
# Joe (2014, p. 5) names rMTCJ = reflected Mardia-Takahasi-Cook-Johnson copula
"rMTCJ" <- function(u, v, para, ...) {
  u + v - 1 + ((1-u)^(-para) + (1-v)^(-para) - 1)^(-1/para)
} # Survival Copula ("reflected" in Joe's terms)
densityCOPplot(cop=rMTCJ, para=1.0760, n=9000, snv=TRUE)
# The density plot matches that shown by Joe (2014, p. 11, fig. 1.2, lower left plot)
# for a Spearman Rho equaling 0.5. Let us compute then Rho:
rhoCOP(cop=rMTCJ, para=1.076075) # 0.4999958

# Now let us get really wild with a composition with two modes!
# This example also proves that the grid orientation is consistent with respect
# to the orientation of the simulations.
para <- list(alpha=0.15, beta=0.90, kappa=0.06, gamma=0.96,
  cop1=GHcop, cop2=PLACKETTcop, para1=5.5, para2=0.07)
densityCOPplot(cop=composite2COP, para=para, n=9000)

# Now, let us hack back to a contour density plot with U(0,1) and not N(0,1) margins
# just so show that capability exists, but emphasis of densityCOPplot is clearly
# on Joe's advocacy, because it does not have a default trigger to use U(0,1) margins.
set.seed(12)
H <- densityCOPplot(cop=PLACKETTcop, para=41.25, getmatrix="cdenzz", n=1000, snv=FALSE)
set.seed(12)
UV <- simCOP(cop=PLACKETTcop, para=41.25, n=1000, col=8, snv=FALSE)
U <- as.numeric(colnames(H)); V <- as.numeric(rownames(H))
contour(x=U, y=V, z=t(H), lwd=1.5, cex=2, add=TRUE, col=2) #
## End(Not run)

## Not run:
U <- rCOP(400, cop=PSP, pch=16, col=8, n=400)
CL <- mleCOP(UV, cop=CLcop, interval=c(1, 20))
JO <- mleCOP(UV, cop=JOcopB5, interval=c(0.1, 20))
PL <- mleCOP(UV, cop=PLcop, interval=c(0.1, 20))

AICs <- c(CL$AIC, JO$AIC, PL$AIC)
names(AICs) <- c("Clayton", "Joe(B5)", "Plackett")
print(round(AICs, digits=2))
# Clayton Joe(B5) Plackett
# -156.77 -36.91 -118.38
# So, we conclude Clayton must be the best fit of the three.
```
derCOP

Numerical Derivative of a Copula for V with respect to U

Description

Compute the numerical partial derivative of a copula, which is a conditional distribution function, according to Nelsen (2006, pp. 13, 40–41) with respect to \( u \):

\[
0 \leq \frac{\delta}{\delta u} C(u, v) \leq 1,
\]

or

\[
\Pr[V \leq v \mid U = u] = C_{2|1}(v \mid u) = \lim_{\Delta u \to 0} \frac{C(u + \Delta u, v) - C(u, v)}{\Delta u},
\]

which is to read as the probability that \( V \leq v \) given that \( U = u \) and corresponds to the derdir="left" mode of the function. For derdir="right", we have

\[
\Pr[V \leq v \mid U = u] = \lim_{\Delta u \to 0} \frac{C(u, v) - C(u - \Delta u, v)}{\Delta u},
\]

and for derdir="center" (the usual method of computing a derivative), the following results

\[
\Pr[V \leq v \mid U = u] = \lim_{\Delta u \to 0} \frac{C(u + \Delta u, v) - C(u - \Delta u, v)}{2\Delta u}.
\]

The “with respect to V” versions are available under derCOP2.

Copula derivatives (\( \delta C/\delta u \) or say \( \delta C/\delta v \) derCOP2) are non-decreasing functions meaning that if \( v_1 \leq v_2 \), then \( C(u, v_2) - C(u, v_1) \) is a non-decreasing function in \( u \), thus

\[
\frac{\partial}{\partial u} \frac{C(u, v_2) - C(u, v_1)}{\delta u}
\]

is non-negative, which means

\[
\frac{\delta C(u, v_2)}{\delta u} \geq \frac{\delta C(u, v_1)}{\delta u} \text{ for } v_2 \geq v_1.
\]
Usage

derCOP(cop=NULL, u, v, delu=.Machine$double.eps^0.50,
derdir=c("left", "right", "center"), ...)

Arguments

cop A copula function;
u Nonexceedance probability \( u \) in the \( X \) direction. If the length of \( u \) is unity, then
the length of \( v \) can be arbitrarily long. If the length of \( u \) is not unity, then the
length of \( v \) should be the same, and if not, then only the first value in \( v \) is silently
used;
v Nonexceedance probability \( v \) in the \( Y \) direction (see previous comment on \( u \));
delu The \( \Delta u \) interval for the derivative;
derdir The direction of the derivative as described above. Default is \( \text{left} \) but internally
any setting can be temporarily suspended to avoid improper computations (see
source code); and
... Additional arguments to pass such as the parameters often described in \( \text{para} \)
arguments of other copula functions. (The lack of \( \text{para} = \text{NULL} \) for \( \text{derCOP} \) and
\( \text{derCOP2} \) was either design oversight or design foresight but regardless it is too
late to enforce package consistency in this matter.)

Value

Value(s) for the partial derivative are returned.

Note

A known caveat of the current implementation of the copula derivative is that there is a chance that
the \( \Delta u \) will span a singularity or discontinuous (or nearly so) portion of a copula should it have a
property of singularity (or nearly so). The \( \text{delu} \) is chosen small so the chance is mitigated to be a
small change and certainly appear to work throughout the examples herein. It is not decided whether
a derivative from the positive side (\( \text{dir} = \text{"left"} \)), when failing should switch over to a computation
from the negative side (\( \text{dir} = \text{"right"} \)). The distinction is important for the computation of
the inverse of the derivative \( \text{derCOPinv} \) because the solution finder needs a sign reversal to work.

Author(s)

W.H. Asquith

References


See Also

derCOPinv, derCOP2
Examples

derCOP(cop=W, 0.4, 0.6); derCOP(cop=P, 0.4, 0.6); derCOP(cop=M, 0.4, 0.6)

lft <- derCOP(cop=PSP, 0.4, 0.6, derdir="left")
rgt <- derCOP(cop=PSP, 0.4, 0.6, derdir="right")
cnt <- derCOP(cop=PSP, 0.4, 0.6, derdir="center")
cat(c(lft,rgt,cnt,"\n"))
#stopifnot(all.equal(lft,rgt), all.equal(lft,cnt))

Let us contrive a singularity through this NOT A COPULA in the function "afunc".
"afunc" <- function(u,v, ...) return(ifelse(u <= 0.5, sqrt(u^2+v^2), P(u,v,...)))
lft <- derCOP(cop=afunc, 0.5, 0.67, derdir="left")
rgt <- derCOP(cop=afunc, 0.5, 0.67, derdir="right")
cnt <- derCOP(cop=afunc, 0.5, 0.67, derdir="center")
cat(c(lft,rgt,cnt,"\n")) # The "right" version is correct.

Description

Compute the numerical partial derivative of a copula, which is a conditional distribution function, according to Nelsen (2006, pp. 13, 40–41) with respect to v:

\[ 0 \leq \frac{\delta}{\delta v} C(u,v) \leq 1, \]

or

\[ \Pr[U \leq u \mid V = v] = C_1|2(u|v) = \lim_{\Delta v \to 0} \frac{C(u,v + \Delta v) - C(u,v)}{\Delta v}, \]

which is to read as the probability that U ≤ u given that V = v and corresponds to the derdir="left" mode of the function. For derdir="right", the following results

\[ \Pr[U \leq u \mid V = v] = \lim_{\Delta v \to 0} \frac{C(u,v) - C(u,v - \Delta v)}{\Delta v}, \]

and for derdir="center" (the usual method of computing a derivative), the following results

\[ \Pr[U \leq u \mid V = v] = \lim_{\Delta v \to 0} \frac{C(u,v + \Delta v) - C(u,v - \Delta v)}{2\Delta v}. \]

The “with respect to U” versions are under derCOP.

Copula derivatives (\(\delta C / \delta v\) or say \(\delta C / \delta u\) derCOP) are non-decreasing functions meaning that if \(u_1 \leq u_2\), then \(C(u_2, v) - C(u_1, v)\) is a non-decreasing function in \(v\), thus

\[ \frac{\delta(C(u_2, v) - C(u_1, v))}{\delta v} \]

is non-negative, which means

\[ \frac{\delta C(u_2, v)}{\delta v} \geq \frac{\delta C(u_1, v)}{\delta v} \text{ for } u_2 \geq u_1. \]
Usage

```r
derCOP2(cop=NULL, u, v, delv=.Machine$double.eps^0.50,
derdir=c("left", "right", "center"), ...)
```

Arguments

- `cop`: A copula function;
- `u`: Nonexceedance probability \( u \) in the \( X \) direction. If the length of \( u \) is unity, then the length of \( v \) can be arbitrarily long. If the length of \( u \) is not unity, then the length of \( v \) should be the same and if not only the first value in \( v \) will be silently used;
- `v`: Nonexceedance probability \( v \) in the \( Y \) direction (see previous comment on \( u \));
- `delv`: The \( \Delta v \) interval for the derivative;
- `derdir`: The direction of the derivative as described above. Default is `left` but internally any setting can be temporarily suspended to avoid improper computations (see source code); and
- `...`: Additional arguments to pass such as the parameters often described in `para` arguments of other copula functions. (The lack of `para=NULL` for `derCOP` and `derCOP2` was either design oversight or design foresight but regardless it is too late to enforce package consistency in this matter.)

Value

Value(s) for the partial derivative are returned.

Note

A known caveat of the current implementation of the copula derivative is that there is a chance that the \( \Delta v \) will span a singularity or discontinuous (or nearly so) portion of a copula should it have a property of singularity (or nearly so). The `delv` is chosen small so the chance is mitigated to be a small change and certainly seems to work throughout the examples herein. It is not decided whether a derivative from the positive side (`dir="left"`), when failing should switch over to a computation from the negative side (`dir="right"`). The distinction is important for the computation of the inverse of the derivative `derCOPinv2` because the solution finder needs a sign reversal to work.

Author(s)

W.H. Asquith

References


See Also

`derCOPinv2, derCOP`
Examples

derCOP2(cop=W, 0.4, 0.6); derCOP2(cop=P, 0.4, 0.6); derCOP2(cop=M, 0.4, 0.6)

lft <- derCOP2(cop=P, 0.4, 0.6, derdir="left")
rgt <- derCOP2(cop=P, 0.4, 0.6, derdir="right")
cnt <- derCOP2(cop=P, 0.4, 0.6, derdir="center")
cat(c(lft, rgt, cnt, \"\n\")
# stopifnot(all.equal(lft, rgt), all.equal(lft, cnt))

# Let us contrive a singularity though this NOT A COPULA in the function "afunc".
"afunc" <- function(u,v, ...) return(ifelse(u <= 0.5, sqrt(u^2+v^2), P(u,v, ...)))
lft <- derCOP2(cop=afunc, 0.67, 0.5, derdir="left")
rgt <- derCOP2(cop=afunc, 0.67, 0.5, derdir="right")
cnt <- derCOP2(cop=afunc, 0.67, 0.5, derdir="center")
cat(c(lft, rgt, cnt, \"\n\")) # For this example, all are correct (see derCOP examples)

---

derCOPinv

Numerical Derivative Inverse of a Copula for V with respect to U

Description

Compute the inverse of a numerical partial derivative for \( V \) with respect to \( U \) of a copula, which is a conditional quantile function for nonexceedance probability \( t \), or

\[
t = c_u(v) = C_{2|1}^{(-1)}(v|u) = \frac{\delta C(u,v)}{\delta u},
\]

and solving for \( v \). Nelsen (2006, pp. 13, 40–41) shows that this inverse is quite important for random variable generation using the conditional distribution method. This function is not vectorized and will not be so.

Usage

derCOPinv(cop=NULL, u, t, trace=FALSE,
        delu=.Machine$double.eps^0.50, para=NULL, ...)

Arguments

cop 
A copula function;

u 
A single nonexceedance probability \( u \) in the \( X \) direction;

t 
A single nonexceedance probability level \( t \);

trace 
A logical controlling a message on whether the signs on the uniroot are the same—this is helpful in exploring the numerical derivative limits of a given implementation of a copula.

delu 
The \( \Delta u \) interval for the derivative;

para 
Vector of parameters or other data structures, if needed, to pass to \( \text{cop} \); and

... 
Additional arguments to pass to the copula.
**Note**

*AN EDUCATIONAL OPPORTUNITY*—The Farlie-Gumbel-Morgenstern copula \( \text{FGM}(u, v) \) (\( \text{FGMcop} \)) (Joe, 2014, p. 213) is

\[
\text{FGM}(u, v; \Theta) = uv[1 + \Theta(1 - u)(1 - v)],
\]

where \(-1 \leq \Theta \leq 1\) has analytical solutions to the conditional cumulative distribution function (CDF) \( C_{2|1}(v|u) \) as

\[
C_{2|1}(v|u) = v[1 + \Theta(1 - v)(1 - 2u)],
\]

and the inverse of the conditional CDF as

\[
C_{2|1}^{-1}(v|u) = \frac{[1 + \Theta(1 - 2u)] - \sqrt{[1 + \Theta(1 - 2u)]^2 - 4t(1 - 2u)}}{2\Theta(1 - 2u)}.
\]

These three functions for the copula can be defined in \( \text{R} \) by

```
"FGMcop" <- function(u,v, para=NULL, ...) u*v*(1 + para*(1-u)*(1-v) 
"joeFGMder" <- function(u,v, para=NULL, ...) v*(1 + para*(1-v)*(1-2*u))
"joeFGMderinv" <- function(u,t, para=NULL, ...) {
  K <- (1-2*u)
  ((1 + para*K) - sqrt((1 + para*K)^2 - 4*t*K))/(2*para*K)
}
```

The \( C_{2|1}^{-1}(v|u) \) is critical for simulation by the conditional simulation method. Although exclusively for simulation, \text{copBasic} uses inversion of the numerical derivative, the \( \text{FGM} \) copula has three representations of supposedly the same analytical algorithm for simulation in the literature (Durante, 2007; Johnson, 1987; Nelsen, 2006). An opportunity for comparison is thus available.

The three analytical algorithms for nonexceedance probability \( t \) given \( u \) by mathematics and code, following Durante (2007, p. 245), are

\[
A = \Theta(1 - 2u) - 1,
B = \sqrt{A^2 - 4t(A + 1)}, \quad \text{and} \quad v = 2t/(B - A),
\]

and in \( \text{R} \), this “Durante algorithm” is

```
"durFGMderinv" <- function(u,t, para=NULL, ...) { # Durante (2007, p. 245)
  A <- para*(1-2*u) - 1; B <- sqrt(A^2 - 4*t*(A+1)); return(2*t/(B - A))
}
```

and, letting \( K = (2u - 1) \), following Johnson (1987, p. 185)

\[
A = K\Theta - 1
B = \sqrt{1 - 2K\Theta + (K\Theta)^2 + 4tK\Theta}
\]

\[
v = 2t/(B - A)
\]

and in \( \text{R} \), this “Johnson algorithm” is
"jonFGMderinv" <- function(u,t, para=NULL, ...) { # Johnson (1987, p. 185)
  K <- (2*u - 1)
  A <- K*para - 1; B <- sqrt(1 - 2*K*para + (K*para)^2 + 4*t*K*para)
  2*t/(B - A)
}

and finally following Nelsen (2006, p. 87)

\[
A = 1 + \theta(1 - 2u),
\]
\[
B = \sqrt{A^2 - 4t(A - 1)}, \quad \text{and}
\]
\[
v = 2t/(B + A),
\]

and in R, this “Nelsen algorithm” is

"nelFGMderinv" <- function(u,t, para=NULL, ...) { # Nelsen (2006, p. 87)
  A <- 1 + para*(1-2*u); B <- sqrt(A^2 - 4*t*(A-1)); return(2*t/(B + A))
}

With appropriate code now available, two comparisons can be made in the following sections.

**CONDITIONAL DISTRIBUTION FUNCTION**—A comparison of the analytical \(\text{FGM}(u,v)\) derivative shows that Joe’s equation is congruent with the numerical derivative of \(\text{copBasic}\):

\[
\text{joeFGMder}(0.8, 0.44, \text{para=0.78}) \quad \# \quad 0.3246848 \quad \text{(Joe, 2014)}
\]
\[
\text{derCOP}(\quad 0.8, 0.44, \text{para=0.78}, \text{cop=FGMcop}) \# \quad 0.3246848 \quad \text{(copBasic )}
\]

and the result will be used in the computations that follow.

A comparison for \(t = 0.3246848\) of the analytical inverse and the numerical optimization of the numerical derivative of \(\text{copBasic}\) is

\[
\text{joeFGMderinv}(0.8, 0.3246848, \text{para=0.78}) \quad \# \quad 0.5327603
\]
\[
\text{derCOPinv}(\quad 0.8, 0.3246848, \text{para=0.78}, \text{cop=FGMcop}) \# \quad 0.4399934 \quad \rightarrow \quad 0.44
\]

where obviously, the two results are not in agreement—so something is amiss. Because many examples in this documentation clearly demonstrate numerical reliability, a tentative conclusion is that Joe’s listed equation must be in error. Let us check this hypothesis against the three other sources:

\[
\text{durFGMderinv}(0.8, 0.3246848, \text{para=0.78}) \quad \# \quad 0.2074546 \quad \text{(Durante, 2007)}
\]
\[
\text{jonFGMderinv}(0.8, 0.3246848, \text{para=0.78}) \# \quad 0.44 \quad \text{(Johnson, 1987)}
\]
\[
\text{nelFGMderinv}(0.8, 0.3246848, \text{para=0.78}) \# \quad 0.44 \quad \text{(Nelsen, 2006)}
\]

The result from Durante (2007) is different from both Joe (2014) and from \(\text{copBasic}\). However, the Johnson (1987) and Nelsen (2006) versions are equivalent and congruent to \(\text{copBasic}\) with the distinctly different numerical methods of derCOPinv. These incongruent results demonstrate that care is needed when navigating the copula literature and the usefulness of the \(\text{copBasic}\)-style implementation of copula theory. In words, these computations show that the \(t \approx 32\text{nd}\) percentile of the \(\text{FGM}\) copula given that the 80th percentile in \(U\) is about the 44th percentile of \(V\).
Author(s)

W.H. Asquith

References


See Also

derCOP

Examples

u <- runif(1); t <- runif(1)
derCOPinv(u,t, cop=W)  # perfect negative dependence
derCOPinv(u,t, cop=P)  # independence
derCOPinv(u,t, cop=M)  # perfect positive dependence
derCOPinv(u,t, cop=PSP) # a parameterless copula example

## Not run:
# Simulate 500 values from product (independent) copula
plot(NA,NA, type="n", xlim=c(0,1), ylim=c(0,1), xlab="U", ylab="V")
for(i in 1:500) {
  u <- runif(1); t <- runif(1)
  points(u, derCOPinv(cop=P, u, t), cex=0.5, pch=16) # black dots
}
# Now simulate 500 from the Nelsen 4.2.12 copula.
for(i in 1:500) {
  u <- runif(1); t <- runif(1)
  points(u,derCOPinv(cop=N4212cop, para=9.3, u, t), cex=2, pch=16, col=2) # red dots
} #
## End(Not run)

## Not run:
# Zhang and Singh (2019) exam. 3.23, p. 105
# show the application of the derivative inversion C2|1
# for u=0.6036 and t=0.6036 ---> v = 0.4719
derCOPinv( cop=CLcop, 0.6036, 0.6036, para=0.5) # 0.4719 for C2|1
derCOPinv2(cop=CLcop, 0.6036, 0.4028, para=0.5) # 0.4719 for C1|2
# and C2|1 and C1|2 are equal because the copula has permutation symmetry
isCOP.permsym(cop=CLcop, para=0.5) # TRUE
## End(Not run)
derCOPinv2

Numerical Derivative Inverse of a Copula for U with respect to V

Description
Compute the inverse of a numerical partial derivative for $U$ with respect to $V$ of a copula, which is a conditional quantile function for nonexceedance probability $t$, or

$$ t = c_v(u) = C_{1|2}^{(-1)}(u|v) = \frac{\delta C(u,v)}{\delta v}, $$

and solving for $u$. Nelsen (2006, pp. 13, 40–41) shows that this inverse is quite important for random variable generation using the conditional distribution method. This function is not vectorized and will not be so.

Usage
derCOPinv2(cop=NULL, v, t, trace=FALSE, delv=.Machine$double.eps^0.50, para=NULL, ...)

Arguments
cop A copula function;
v A single nonexceedance probability $v$ in the $Y$ direction;
t A single nonexceedance probability level $t$;
trace A logical controlling a message on whether the signs on the uniroot are the same—this is helpful in exploring the numerical derivative limits of a given implementation of a copula.
delv The $\Delta v$ interval for the derivative;
para Vector of parameters or other data structure, if needed, to pass to cop; and
... Additional arguments to pass to the copula.

Value
Value(s) for the derivative inverse are returned.

Author(s)
W.H. Asquith

References

See Also
derCOP2
Examples

\begin{verbatim}
u <- runif(1); t <- runif(1)
derCOPinv2(u, t, cop=W)  # perfect negative dependence
derCOPinv2(u, t, cop=P)  # independence
derCOPinv2(u, t, cop=M)  # perfect positive dependence
derCOPinv2(u, t, cop=PSP) # a parameterless copula example

## Not run:
# Simulate 500 values from product (independent) copula
plot(NA, NA, type="n", xlim=c(0,1), ylim=c(0,1), xlab="U", ylab="V")
for(i in 1:500) {
  v <- runif(1); t <- runif(1)
  points(derCOPinv2(cop=P, v, t), v, cex=0.5, pch=16) # black dots
}
# Simulate 500 of a Frechet Family copula and note crossing singularities.
for(i in 1:500) {
  v <- runif(1); t <- runif(1)
  u <- derCOPinv2(v, t, cop=FRECHETcop, para=list(alpha=0.7, beta=0.169))
  points(u, v, cex=2, pch=16, col=2) # red dots
}
## End(Not run)
\end{verbatim}

\begin{description}
\item[diagCOP] The Diagonals of a Copula
\end{description}

Description

Compute the primary diagonal or alternatively the secondary diagonal (Nelsen, 2006, pp. 12 and 16) of copula \( C(u,v) \). The primary diagonal is defined as

\[
\delta_C(t) = C(t,t),
\]

and the secondary diagonal is defined as

\[
\delta_C^*(t) = C(t,1-t).
\]

Plotting is provided by this function because the diagonals are such important visual attributes of a copula. This function computes whole diagonals. If individual values are desired, then users are asked to use function calls along the diagonal such as \( \text{COP}(0.25, 0.25, \text{cop}=\text{P}) \) for the primary diagonal and \( \text{COP}(0.25, 1-0.25, \text{cop}=\text{P}) \) for the secondary diagonal, where for both examples the independence copula \( (uv = \Pi; P) \) was chosen for purposes of clarification.

The \( \delta_C(t) \) is related to order statistics of the multivariate sample (here bivariate) (Durante and Sempi, 2015, p. 68). The probability for the maxima is \( \text{Pr}[\max(u,v) \leq t] = C(t,t) = \delta_C(t) \) and the probability for the minima is \( \text{Pr}[\min(u,v) \leq t] = 2t - \delta_C(t) \).

Usage

\begin{verbatim}
diagCOP(cop=NULL, para=NULL, secondary=FALSE, ploton=TRUE, lines=TRUE, delt=0.005, ...)
\end{verbatim}
diagCOP

Arguments

- **cop**: A copula function;
- **para**: Vector of parameters, if needed, to pass to the copula;
- **secondary**: A logical to toggle the secondary diagonal;
- **ploton**: A logical to toggle on the plot;
- **lines**: Draw the lines of diagonal to the current device;
- **delt**: The increment of the diagonal curve to plot, defaults to 0.5-percent intervals, which should be small enough to resolve fine curvature for many copulas in practice; and
- **...**: Additional arguments to pass to the `plot()` and `lines()` functions in R.

Value

An R list of the $t$ values, $\delta_C(t, t)$ (primary) or $\delta_C(t, 1-t)$ (secondary diagonal), along with a tag as to which diagonal is returned.

Author(s)

W.H. Asquith

References


See Also

diagCOPatf, COP, sectionCOP

Examples

```r
## Not run:
# The primary diagonal of the W, P, M, and PSP copulas on the same plot
D <- diagCOP(cop=W, lwd=2)
D <- diagCOP(cop=P, lty=2, ploton=FALSE)
D <- diagCOP(cop=M, col=2, ploton=FALSE)
D <- diagCOP(cop=PSP, col=3, ploton=FALSE)
mtext("PRIMARY DIAGONAL OF SIMPLE COPULAS") # four primary diagonals
## End(Not run)

## Not run:
# The secondary diagonal of the W, P, M, and PSP copulas on the same plot
D <- diagCOP(cop=W, lwd=2, secondary=TRUE)
D <- diagCOP(cop=P, lty=2, secondary=TRUE, ploton=FALSE)
D <- diagCOP(cop=M, col=2, secondary=TRUE, ploton=FALSE)
D <- diagCOP(cop=PSP, col=3, secondary=TRUE, ploton=FALSE)
mtext("SECONDARY DIAGONAL OF SIMPLE COPULAS") # four secondary diagonals
## End(Not run)
```
**diagCOPatf**

**Numerical Rooting the Diagonal of a Copula**

**Description**

Compute a numerical root along the primary diagonal (Nelsen, 2006, pp. 12 and 16) of copula $C(u, v)$ having joint probability $F$. The diagonals treat the nonexceedance probabilities $u$ and $v$ as equals ($u = v = t$). The primary diagonal is defined for a joint nonexceedance probability $t$ as

$$F = C(t, t) \rightarrow t = \delta_C^{(-1)}(f),$$

where the function solves for $t$. Examples using the concept behind diagCOPatf are available under duCOP and jointCOP, thus the diagCOPatf function can be also called by either jointCOP and joint.curvesCOP. Internally, the function uses limits of the root finder that are not equal to the anticipated interval $[0, 1]$, but equal to “small” (see description for argument interval). The function does trap for $f = 0$ by returning zero and $f = 1$ by returning unity.

**Usage**

```r
diagCOPatf(f, cop=NULL, para=NULL, verbose=FALSE, interval=NULL, tol=.
```

```r
Machine$double.eps/10, ...)
diagCOPinv(f, cop=NULL, para=NULL, verbose=FALSE, interval=NULL, tol=.
```

```r
Machine$double.eps/10, ...)
```

**Arguments**

- **f**: Joint probability values as a nonexceedance probability $F$ for which to compute the root $t$;
- **cop**: A copula function;
- **para**: Vector of parameters, if needed, to pass to the copula;
- **verbose**: If TRUE then the whole output of the numerical root is returned using only the first value provided by argument $f$;
- **interval**: An optional interval for the root search. The default is interval=c(lo, 1-lo) for lo=.Machine$double.eps because of difficulties for an interval on $[0, 1]$; and
- **tol**: The tolerance to pass to uniroot. The default here is much smaller than the default of the uniroot() function in R because of possibility that diagCOPatf would be used at extremely large nonexceedance probabilities; and
- **...**: Additional arguments to pass.

**Value**

An R list of the root by the uniroot() function in R is returned if verbose is TRUE, otherwise the roots (diagonal inverses) for $t$ are returned, and if an individual inverse operation fails, then a NA is returned instead.
Author(s)
W.H. Asquith

References

See Also
diagCOP, jointCOP, joint.curvesCOP

Examples
diagCOPatf(0.67, cop=PSP) # 0.8023879
diagCOPatf(0.99, cop=M) # 0.99 (now see the example below)
## Not run:
# Several functions from the lmomco package are needed.
# Suppose we have two phenomena with these log10 L-moments:
lmrA <- lmomco::vec2lmom(c(3.97, 0.485, -0.1178, 0.06857))
lmrB <- lmomco::vec2lmom(c(3.77, 0.475, -0.1377, 0.08280))
# Suppose we think that the Gumbel-Hougaard copula is appropriate with a Tau=0.45
Tau <- 0.45 # Kendall Tau between A and B.
# Suppose that the F=0.99 for either A and B provides a common risk level when they
# are considered in isolation. But what if A and B are rivers that join and joint
# FF=0.99 at their union is of interest?
FF <- 0.99
parA <- lmomco::lmom2par(lmrA, type="kap")
parB <- lmomco::lmom2par(lmrB, type="kap")
EventA <- lmomco::qlmomco(FF, parA)
EventB <- lmomco::qlmomco(FF, parB)
ApB <- 10^(EventA) + 10^(EventB) # Purely an additive conceptualization
# The FF=0.99 event is assumed to occur simultaneously on both streams, which is
# equivalent to saying that the correlation between the two is absolute 1-to-1.

# Now consider including the association as measured by Kendall Tau:
Fjoint <- diagCOPatf(FF, cop=GHcop, para=GHcop(tau=Tau)$para)
EventAj <- lmomco::qlmomco(Fjoint, parA)
EventBj <- lmomco::qlmomco(Fjoint, parB)
AcB <- 10^(EventAj) + 10^(EventBj) # Joint probability 0.99 at the union

# Now consider the association if the rivers are INDEPENDENT:
Fjoint <- diagCOPatf(FF, cop=GHcop, para=GHcop(tau=0)$para)
EventAj <- lmomco::qlmomco(Fjoint, parA)
EventBj <- lmomco::qlmomco(Fjoint, parB)
AiB <- 10^(EventAj) + 10^(EventBj) # Joint probability 0.99 at the union

# ApB = 312,000 # The perfectly simultaneous addition makes too little.
# AcB = 323,000 # The copula preserves at least the known association.
# AiB = 330,000 # The independence conceptualization makes too much.
## End(Not run)
duCOP

The Dual of a Copula Function

duCOP

The Dual of a Copula Function

Description

Compute the dual of a copula (function) from a copula (Nelsen, 2006, pp. 33–34), which is defined as

\[ \Pr[U \leq u \text{ or } V \leq v] = \tilde{C}(u, v) = u + v - C(u, v), \]

where \( \tilde{C}(u, v) \) is the dual of a copula and \( u \) and \( v \) are nonexceedance probabilities. The dual of a copula is the expression for the probability that either \( U \leq u \) or \( V \leq v \), which is unlike the co-copula (function) (see coCOP) that provides \( \Pr[U > u \text{ or } V > v] \). The dual of a copula is a function and not in itself a copula. The dual of the survival copula (surCOP) is the co-copula (function) (coCOP). Some rules of copulas mean that

\[ \hat{C}(u', v') + \tilde{C}(u, v) = 1, \]

where \( \hat{C}(u', v') \) is the survival copula in terms of exceedance probabilities \( u' \) and \( v' \) or in copBasic code that the functions surCOP + duCOP equal unity.

The function duCOP gives “protection” against simultaneous (concurrent or dual) risk by failure if and only if failure is caused (defined) by both hazard sources \( U \) and \( V \) being by themselves responsible for failure. Expressing this in terms of an annual probability of occurrence (\( q \)), one has

\[ q = 1 - \Pr[U \leq v \text{ or } V \leq v] = 1 - \tilde{C}(u, v) \text{ or} \]

in R code \( q <- 1 - \text{duCOP}(u, v) \). So, as a mnemonic: A dual of a copula is the probability of nonexceedance if the hazard sources must dual (concur, link, pair, twin, twain) between each other to cause failure. An informative graphic is shown within copBasic-package.

Usage

duCOP(u, v, cop=NULL, para=NULL, ...)

Arguments

u Nonexceedance probability \( u \) in the \( X \) direction;

v Nonexceedance probability \( v \) in the \( Y \) direction;

cop A copula function;

para Vector of parameters or other data structure, if needed, to pass to the copula; and

Additional arguments to pass (such as parameters, if needed, for the copula in the form of a list.

Value

Value(s) for the dual of a copula are returned.
Note

There can be confusion in the interpretation and implementation of the or condition of joint probability provided by \( C(u, v) \). Two types of or’s seemingly exist depending on one’s concept of the meaning of “or.” To start, there is the “either or both” conceptualization (joint or) that encompasses either “event” (say a loss) of importance for random variables \( U \) and \( V \) as well as the joint and conditions where both variables simultaneously are generating an event of importance.

Let us continue by performing a massive simulation for the \( \text{PSP}(u, v) \) copula (PSP) and set an either event standard on the margins as 10 percent for an arbitrary starting point. The PSP has positive association with lower tail dependency, and the example here considers the left tail as the risk tail.

```r
Event <- 0.1; nn <- 100000; set.seed(9238)
UV <- simCOP(n=nn, cop=PSP, graphics=FALSE) # 1E5 realizations
Next, let us step through counting and then make theoretical comparisons using copula theory. The joint and condition as nonexceedances is

\[
\text{ANDs} = \frac{\text{length}(\text{UV}\$U[\text{UV}\$U <= \text{Event} & \text{UV}\$V <= \text{Event}])}{\text{nn}}
\]

\[
\text{ANDt} = \text{duCOP(\text{Event, Event, cop=PSP})}
\]

message(  "Joint AND by simulation = ", round(ANDs, digits=5),  "\n Joint AND by theory = ", round(ANDt, digits=5))  
# ANDs = 0.05348 and ANDt = 0.05263 (numerical congruence)

where it is obvious that the simulations and theory estimate about the same joint and condition. Now, the joint or condition as nonexceedances is

\[
\text{ORs} = \frac{\text{length}(\text{UV}\$U[\text{UV}\$U <= \text{Event} | \text{UV}\$V <= \text{Event}])}{\text{nn}}
\]

\[
\text{ORT} = \text{duCOP(\text{Event, Event, cop=PSP})}
\]

message(  "Joint OR by simulation = ", round(ORs, digits=5),  "\n Joint OR by theory = ", round(ORT, digits=5))  
# ORs = 0.14779 and ORt = 0.14737 (numerical congruence)

where it is obvious that the simulations and theory estimate about the same joint or condition. Finally, the joint mutually exclusive or condition as nonexceedances is

\[
\text{eORs} = \frac{\text{length}((\text{UV}\$U[\text{UV}\$U <= \text{Event} | \text{UV}\$V <= \text{Event}) &  
(\text{UV}\$U <= \text{Event} & \text{UV}\$V <= \text{Event})))}{\text{nn}}
\]

\[
\text{eORt} = \text{ORT} - \text{ANDt} # theoretical computation
\]

message(  "Joint exclusive OR by simulation = ", round(eORs, digits=5),  "\n Joint exclusive OR by theory = ", round(eORt, digits=5))  
# eORs = 0.09431 and eORt = 0.09474 (numerical congruence)

where it is obvious that the simulations and theory estimate about the same joint mutually exclusive or condition, and where it is shown that the prior two theoretical joint probabilities can be subtracted from each to yield the mutually exclusive or condition.

Let us then play out a scenario in which it is judged that of the events causing damage that the simultaneous occurrence is worse but that engineering against about 5 percent of events not occurring at the same time represents the most funding available. Using numerical methods, it is possible to combine \( C \) and \( \tilde{C} \) and assume equal marginal risk in \( U \) and \( V \) as the following list shows:
"designf" <- function(t) {  
  duCOP(t, t, cop=PSP) - COP(t, t, cop=PSP) - 5/100  # 5 percent
}

dThres <- uniroot/designf, c(.Machine$double.eps,0.5))$root

where the uniroot function performs the optimization and the .Machine$double.eps value is used because the PSP is NaN for zero probability. (It is unity for unity marginal probabilities.) The design threshold on the margins then is dThres ≈ 0.05135. In other words, the designThres is the marginal probability that results in about 5 percent of events not occurring at the same time. Then considering the simulated sample and counting the nonexceedances by code one achieves:

Damage <- length( UV$U[ UV$U <= dThres | UV$V <= dThres ])
SimDamage <- length( UV$U[ UV$U <= dThres & UV$V <= dThres ])
NonSimDamage <- length( (UV$U <= dThres & UV$V <= dThres) & (UV$U <= dThres & UV$V <= dThres))

message(  
  "Damaging Events (sim.) = ", Damage,
  "Simultaneous damaging events (sim.) = ", SimDamage,
  "Nonsimultaneous damaging events (sim.) = ", NonSimDamage
)

but also the theoretical expectations are readily computed using copula theory:

tDamage <- as.integer(duCOP(dThres, dThres, cop=PSP) * nn)
tSimDamage <- as.integer( COP(dThres, dThres, cop=PSP) * nn)
tNonSimDamage <- tDamage - tSimDamage

message(  
  "Damaging Events (theory) = ", tDamage,
  "Simultaneous damaging events (theory) = ", tSimDamage,
  "Nonsimultaneous damaging events (theory) = ", tNonSimDamage
)

The counts from the former listing are 7,670; 2,669; and 5,001, whereas the respective counts from the later listing are 7,635; 2,635; and 5,000. Numerical congruency in the counts thus exists.

Author(s)

W.H. Asquith

References


See Also

COP, coCOP, surCOP, jointCOP, joint.curvesCOP

Examples

u <- runif(1); t <- runif(1)
duCOP(cop=W,u,t)  # joint or probability for perfect negative dependence
duCOP(cop=P,u,t)  # joint or probability for perfect independence
duCOP(cop=M,u,t)  # joint or probability for some positive dependence
duCOP(cop=PSP,u,t)  # joint or probability for some positive dependence
The bivariate empirical copula (Nelsen, 2006, p. 219) for a bivariate sample of length \( n \) is defined for random variables \( X \) and \( Y \) as

\[
C_n\left(\frac{i}{n}, \frac{j}{n}\right) = \frac{\text{number of pairs } (x, y) \text{ with } x \leq x(i) \text{ and } y \leq y(j)}{n},
\]

where \( x(i) \) and \( y(j) \), \( 1 \leq i, j \leq n \) or expressed as

\[
C_n\left(\frac{i}{n}, \frac{j}{n}\right) = \frac{1}{n} \sum_{i=1}^{n} 1\left(\frac{R_i}{n} \leq u_i, \frac{S_i}{n} \leq v_i\right),
\]

where \( R_i \) and \( S_i \) are ranks of the data for \( U \) and \( V \), and \( 1(\cdot) \) is an indicator function that scores 1 if condition is true otherwise scoring zero. Using more generic notation, the empirical copula can be defined by

\[
C_n(u,v) = \frac{1}{n} \sum_{i=1}^{n} 1(u_{obs}^i \leq u_i, v_{obs}^i \leq v_i),
\]

where \( u_{obs}^i \) and \( v_{obs}^i \) are thus some type of nonparametric nonexceedance probabilities based on counts of the underlying data expressed in probabilities.

**Hazen Empirical Copula**—The “Hazen form” of the empirical copula is

\[
C_n^H(u,v) = \frac{1}{n} \sum_{i=1}^{n} 1\left(\frac{R_i - 0.5}{n} \leq u_i, \frac{S_i - 0.5}{n} \leq v_i\right),
\]

which can be triggered by \( \text{ctype} = \text{"hazen"} \). This form is named for this package because of direct similarity of the Hazen plotting position to the above definition. Joe (2014, pp. 247–248) uses the Hazen form. Joe continues by saying “[the] adjustment of the uniform score \( [(R - 0.5)/n] \) could be done in an alternative form, but there is [asymptotic] equivalence[, and that] \( C_n^H \) puts mass of \( n^{-1} \) at the tuples \( [(r_{i1} - 0.5)/n, \ldots, (r_{id} - 0.5)/n] \) for \( i = 1, \ldots, n \).” A footnote by Joe (2014) says that “the conversion \( [R/(n + 1)] \) is commonly used for the empirical copula.” This later form is the
“Weibull form” described next. Joe’s preference for the Hazen form is so that the sum of squared normal scores is closer to unity for large \( n \) than such a sum would be attained using the Weibull form.

**Weibull Empirical Copula**—The “Weibull form” of the empirical copula is

\[
C_{n}^{W}(u, v) = \frac{1}{n} \sum_{i=1}^{n} 1 \left( \frac{R_{i}}{n + 1} \leq u_{i}, \frac{S_{i}}{n + 1} \leq v_{i} \right),
\]

which can be triggered by `ctype="weibull"`. This form is named for this package because of direct similarity of the Weibull plotting position to the definition, and this form is the default (see argument description).

**Bernstein Empirical Copula**—The empirical copula can be extended nonparametrically as the Bernstein empirical copula (Hernández-Maldonado, Díaz-Viera, and Erdely, 2012) and is formulated as

\[
C_{n}^{B}(u, v; \eta) = \sum_{i=1}^{n} \sum_{j=1}^{n} C_{n} \left( \frac{i}{n}, \frac{j}{n} \right) \times \eta(i, j; u, v),
\]

where the individual Bernstein weights \( \eta(i, j) \) for the \( k \)th paired value of the \( u \) and \( v \) vectors are

\[
\eta(i, j; u, v) = \left( \begin{array}{c} n \\ i \end{array} \right) u^{i} (1 - u)^{n-i} \times \left( \begin{array}{c} n \\ j \end{array} \right) v^{j} (1 - v)^{n-j}.
\]

The Bernstein extension, albeit conceptually pure in its shuffling by binomial coefficients and left- and right-tail weightings, is quite CPU intensive as inspection of the equations above indicates a nest of four `for()` loops in \( \text{R} \). (The native \( \text{R} \) code of \texttt{copBasic} uses the `sapply()` function in \( \text{R} \) liberally for substantial but not a blazing fast speed increase.) The Bernstein extension results in a smoother surface of the empirical copula and can be triggered by `ctype="bernstein"`.

**Checkerboard Empirical Copula**—A simple smoothing to the empirical copula is the checkerboard empirical copula (Segers et al., 2017) that has been adapted from the \texttt{copula} package. It is numerically intensive like the Bernstein and possibly of limited usefulness for large sample sizes. The checkerboard extension can be triggered by `ctype="checkerboard"` and is formulated as

\[
C_{n}^{\#}(U) = \frac{1}{n + o} \sum_{i=1}^{n} \prod_{j=1}^{d} \min[\max[nU_{i} - R_{i,j}^{(n)} + 1, 0], 1],
\]

where \( U \) is a \( d = 2 \) column matrix of \( u \) and \( v \), \( \texttt{R} \) is a rank function, and \( o \) is an offset term on \([0, 1]\). The empirical copula frequency can be defined (Nelson, 2006, p. 219) as

\[
c_{n}(u, v) = C_{n} \left( \frac{i}{n}, \frac{j}{n} \right) - C_{n} \left( \frac{i-1}{n}, \frac{j}{n} \right) - C_{n} \left( \frac{i}{n}, \frac{j-1}{n} \right) + C_{n} \left( \frac{i-1}{n}, \frac{j-1}{n} \right).
\]

**Usage**

\[
\text{EMPIRcop}(u, v, \text{para=NULL, ctype=c("weibull", "hazen", "1/n", "bernstein", "checkerboard"), bernprogress=FALSE, checkerboard.offset=0, ...})
\]
Arguments

- **u** Nonexceedance probability \( u \) in the \( X \) direction;
- **v** Nonexceedance probability \( v \) in the \( Y \) direction;
- **para** A vector (single element) of parameters—the \( U \)-statistics of the data (see Examples). Alternatively, para can be a list holding a para as would be done if it were a vector, but arguments bernstein and bernprogress can be optionally included—this feature is provided so that the Bernstein refinement can be controlled within the context of other functions calling EMPIRcop such as by `level.curvesCOP`;
- **ctype** An alternative means for triggering the definition of \( C_n, C_n^H \) (default), \( C_n^W, C_n^B \), or \( C_n^\#: \). This argument of the same name is also used by `blomCOP`;
- **bernprogress** The Bernstein copula extension is CPU intensive(!), so a splash counter is pushed to the console via the `message()` function in \( R \) so as to not discourage the user;
- **checkerboard.offset** A scaling of the ratio \( \text{sum(...)}/(n+\text{offset}) \) for the checkerboard empirical copula; and
- **...** Additional arguments to pass.

Value

Value(s) for the copula are returned.

Note

Not all theoretical measures of copula dependence (both measures of association and measures of asymmetry), which use numerical integration by the `integrate()` function in \( R \), can be used for all empirical copulas because of “divergent” integral errors; however, examples using *Hoeffding Phi* (\( \Phi_C; \) `hoeffCOP`) and shown under Examples. Other measures of copula dependence include `blomCOP`, `footCOP`, `ginicOP`, `rhoCOP`, `tauCOP`, `wolfCOP`, `joeskewCOP`, and `uvlmoms`. Each of these measures fortunately has a built-in sample estimator.

It is important to distinguish between a sample estimator and the estimation of the measure using the empirical copula itself via the `EMPIRcop` function. The sample estimators (triggered by the `as.sample` arguments for the measures) are reasonably fast and numerically preferred over using the empirical copula. Further, the generally slow numerical integrations for the theoretical definitions of these copula measures might have difficulties. Limited testing, however, suggests prevalence of numerical integration not erroring using the Bernstein extension of the empirical copula, which must be a by-product of the enhanced and sufficient smoothness for the \( R \) default numerical integration to succeed. Many of the measures have brute option for a brute-force numerical integration on a regular grid across the empirical copula—these are slow but should not trigger errors. As a general rule, users should still use the sample estimators instead.

Author(s)

W.H. Asquith
References


See Also
diagCOP, level.curvesCOP, simCOP

Examples

```r
## Not run:
set.seed(62)
EMPIRcop(0.321, 0.78, para = simCOP(n=90, cop=N4212cop, para=2.32, graphics=FALSE)) # [1] 0.3222222
N4212cop(0.321, 0.78, para=2.32) # [1] 0.3201281
## End(Not run)

## Not run:
set.seed(62) # See note below about another seed to try.
psp <- simCOP(n=34, cop=PSP, ploton=FALSE, points=FALSE) * 150
# Pretend psp is real data, the * 150 is to clearly get into an arbitrary unit system.

# The sort=FALSE is critical in the following two calls. Although the Weibull
# plotting positions are chosen, internally EMPIRcop uses ranks, but the model
# here is to imagine one having a sample in native units of the random variables
# and then casting them into probabilities for other purposes.
fakeU <- lmomco::pp(psp[,1], sort=FALSE) # Weibull plotting position i/(n+1)
fakeV <- lmomco::pp(psp[,2], sort=FALSE) # Weibull plotting position i/(n+1)
uv <- data.frame(U=fakeU, V=fakeV); # our U-statistics

# The next four values should be very close if n above were say 1000, but the
# ctype="bernstein" should not be used if n >> 34 because of inherently long runtime.
PSP(0.4, 0.6) # 0.3157895 (compare to listed values below)

# Two seeds are shown so that the user can see that depending on the distribution
# of the values given by para that different coincidences of which method is
# equivalent to another exist.
# For set.seed(62) --- "hazen" == "weibull" by coincidence
# "hazen" --> 0.3529412
# "weibull" --> 0.3529412
# "1/n" --> 0.3235294
# "bernstein" --> 0.3228916
# For set.seed(85) --- "1/n" == "hazen" by coincidence
```
EMPIRcop

# "hazen" --> 0.3529412
# "weibull" --> 0.3823529
# "1/n" --> 0.3529412
# "bernstein" --> 0.3440387

# For set.seed(62) --- not all measures of association can be used for all
# empirical copulas because of 'divergent' integral errors, but this is an example
# for Hoeffding Phi. These computations are CPU intensive, esp. Bernstein.
hoefCOP(as.sample=TRUE, para=uv) # (sample estimator is fast) # 0.4987755
hoefCOP(cop=EMPIRcop, para=uv, ctype="hazen") # 0.5035348
hoefCOP(cop=EMPIRcop, para=uv, ctype="weibull") # 0.4977145
hoefCOP(cop=EMPIRcop, para=uv, ctype="1/n") # 0.4003646
hoefCOP(cop=EMPIRcop, para=uv, ctype="bernstein") # 0.4563724
hoefCOP(cop=EMPIRcop, para=uv, ctype="checkerboard") # 0.4952427
## End(Not run)

# All other example suites shown below are dependent on the pseudo-data in the
# variable uv. It is suggested to not run with a sample size much larger than the
# above n=34 if the Bernstein comparison is intended (wanted) simply because of
# lengthy(!) run times, but the n=34 does provide a solid demonstration how the
# level curves for bernstein weights are quite smooth.
## Not run:
# Now let us construct as many as three sets of level curves to the sample
# resided in the uv sample from above using the PSP copula.
level.curvesCOP(cop=PSP); # TRUE, parametric, fast, BLACK CURVES

# Empirical copulas can consume lots of CPU.
# RED CURVES, if n is too small, uniroot() errors might be triggered and likely
# will be using the sample size of 34 shown above.
level.curvesCOP(cop=EMPIRcop, para=uv, delu=0.03, col=2, ploton=FALSE)

# GREEN CURVES (large CPU commitment)
# Bernstein progress is uninformative because level.curvesCOP() has taken over control.
# bpara <- list(ctype="bernstein", bernprogress=FALSE)
level.curvesCOP(cop=EMPIRcop, para=bpara, delu=0.03, col=3, ploton=FALSE)
# The delu is increased for faster execution but more important,
# notice the greater smoothness of the Bernstein refinement.
## End(Not run)

## Not run:
# Experimental from R Graphics by Murrell (2005, p.112)
"trans3d" <- # blackslashes seem needed for the package
function(x,y,z, pmat) { # for user manual building but bad syntax
  tmat <- cbind(x,y,z,1) %*% pmat # because remember the percent sign is a
  return(tmat[,1:2] / tmat[,4]) # a comment character in LaTeX.
}

the.grid <- EMPIRgrid(para=uv, ctype="checkerboard")
the.diag <- diagCOP(cop=EMPIRcop, para=uv, ploton=FALSE, lines=FALSE)

the.persp <- persp(the.grid$empcop, theta=-25, phi=20,
  xlab="U VARIABLE", ylab="V VARIABLE", zlab="COPULA C(u,v)")
EMPIRcopdf

Data Frame Representation of the Bivariate Empirical Copula

Description

Generate an R data.frame representation of the bivariate empirical copula (Salvadori et al., 2007, p. 140) using the coordinates as preserved in the raw data in the parameter object of the bivariate empirical copula.

Usage

EMPIRcopdf(para=NULL, ...)

Arguments

para A vector (single element) of parameters—the U-statistics of the data (see example) to pass to EMPIRcop; and

... Additional arguments to pass to EMPIRcop.

Value

An R data.frame of \( u, v \), and \( C_n(u,v) \) values of the bivariate empirical copula is returned.

Author(s)

W.H. Asquith

References


See Also

EMPIRcop
EMPIRgrid

Examples

```r
## Not run:
psp <- simCOP(n=39, cop=PSP, ploton=FALSE, points=FALSE) * 150
# Pretend psp is real data, the * 150 is to clearly get into an arbitrary unit system.

# The sort=FALSE is critical in the following two calls to pp() from lmomco.
fakeU <- lmomco::pp(psp[,1], sort=FALSE) # Weibull plotting position i/(n+1)
fakeV <- lmomco::pp(psp[,2], sort=FALSE) # Weibull plotting position i/(n+1)
uv <- data.frame(U=fakeU, V=fakeV) # our U-statistics

empcop <- EMPIRcopdf(para=uv)
plot(empcop$u, empcop$v, cex=1.75*empcop$empcop, pch=16,
    xlab="U, NONEXCEEDANCE PROBABILITY", ylab="V, NONEXCEEDANCE PROBABILITY")
# Dot size increases with joint probability (height of the copulatic surface).
points(empcop$u, empcop$v, col=2) # red circles
## End(Not run)
```

EMPIRgrid

Grid of the Bivariate Empirical Copula

Description

Generate a gridded representation of the bivariate empirical copula (see EMPIRcop, Salvadori et al., 2007, p. 140). This function has the primary intention of supporting 3-D renderings or 2-D images of the copulatic surface, but many empirical copula functions in copBasic rely on the grid of the empirical copula—unlike the functions that support parametric copulas.

Usage

```r
EMPIRgrid(para=NULL, deluv=0.05, verbose=FALSE, ...)
```

Arguments

- `para`: A vector (single element) of parameters—the U-statistics of the data (see example);
- `deluv`: A delta value of the both the `u` and `v` axes (grid edges) for empirical copula estimation by the EMPIRcop function;
- `verbose`: A logical controlling whether the progress during grid building is to be shown; and
- `...`: Additional arguments to pass to EMPIRcop.

Value

An R list of the gridded values of `u`, `v`, and $C_n(u,v)$ values of the bivariate empirical copula is returned. (Well only $C_n(u,v)$ is in the form of a grid as an R matrix.) The deluv used to generated the grid also is returned.
Note
The extensive suite of examples is included here because the various ways that algorithms involving empirical copulas can be tested. The figures also provide excellent tools for education on copulas.

Author(s)
W.H. Asquith

References

See Also
EMPIRcop, EMPIRcopdf

Examples
## Not run:
# EXAMPLE 1:
psp <- simCOP(n=490, cop=PSP, ploton=FALSE, points=FALSE) * 150
# Pretend psp is real data, the * 150 is to clearly get into an arbitrary unit system.

# The sort=FALSE is critical in the following two calls to pp() from lmomco.
fakeU <- lmomco::pp(psp[,1], sort=FALSE) # Weibull plotting position i/(n+1)
fakeV <- lmomco::pp(psp[,2], sort=FALSE) # Weibull plotting position i/(n+1)
uv <- data.frame(U=fakeU, V=fakeV) # our U-statistics

# The follow function is used to make 3-D --> 2-D transformation
# From R Graphics by Murrell (2005, p.112)
"trans3d" <- function(x,y,z, pmat) {  # blackslashes seem needed for the package
  tmat <- cbind(x,y,z,1) %*% pmat # for user manual building but bad syntax
  return(tmat[,1:2] / tmat[,4])  # because remember the percent sign is a comment character in LaTeX.
}

the.grid <- EMPIRgrid(para=uv)
cop.diag <- diagCOP(cop=EMPIRcop, para=uv, ploton=FALSE, lines=FALSE)
empcop <- EMPIRcopdf(para=uv) # data.frame of all points

# EXAMPLE 1: PLOT NUMBER 1
the.persp <- persp(the.grid$empcop, theta=-25, phi=20,
  xlab="U VARIABLE", ylab="V VARIABLE", zlab="COPULA C(u,v)"
)

# EXAMPLE 1: PLOT NUMBER 2 (see change in interaction with variable 'the.grid')
the.persp <- persp(x=the.grid$u, y=the.grid$v, z=the.grid$empcop, theta=-25, phi=20,
  xlab="U VARIABLE", ylab="V VARIABLE", zlab="COPULA C(u,v)"
)

the.diag <- trans3d(cop.diag$t, cop.diag$t, cop.diag$diagcop, the.persp)
lines(the.diag, lwd=4, col=3, lty=2)
EMPIRgrid

points(trans3d(empcop$u, empcop$v, empcop$empcop, the.persp),
      col=rgb(0,1-sqrt(empcop$empcop),1,sqrt(empcop$empcop)), pch=16)
# the sqrt() is needed to darken or enhance the colors

S <- sectionCOP(cop=PSP, 0.2, ploton=FALSE, lines=FALSE)
thonlines <- trans3d(rep(0.2, length(S$t)), S$t, S$seccop, the.persp)
lines(thelines, lw=2, col=6)
S <- sectionCOP(cop=PSP, 0.2, ploton=FALSE, lines=FALSE, dercop=TRUE)
thonlines <- trans3d(rep(0.2, length(S$t)), S$t, S$seccop, the.persp)
lines(thelines, lw=2, col=6, lty=2)

S <- sectionCOP(cop=PSP, 0.85, ploton=FALSE, lines=FALSE, wrtV=TRUE)
thonlines <- trans3d(S$t, rep(0.85, length(S$t)), S$seccop, the.persp)
lines(thelines, lw=2, col=2)
S <- sectionCOP(cop=PSP, 0.85, ploton=FALSE, lines=FALSE, dercop=TRUE)
thonlines <- trans3d(S$t, rep(0.85, length(S$t)), S$seccop, the.persp)
lines(thelines, lw=2, col=2, lty=2)

empder <- EMPIRgridder(empgrid=the.grid)
thonlines <- trans3d(rep(0.2, length(the.grid$v)), the.grid$v, empder[3,], the.persp)
lines(thelines, lw=4, col=6) #
## End(Not run)

## Not run:
# EXAMPLE 2:
# An asymmetric example to demonstrate that the grid is populated with the
# correct orientation---meaning U is the horizontal and V is the vertical.
"MOcop" <- function(u,v, para=NULL) { # Marshall-Olkin copula
  alpha <- para[1]; beta <- para[2]; return(min(v*u^(1-alpha), u*v^(1-beta)))
}
# EXAMPLE 2: PLOT NUMBER 1 # See the asymmetry
uv <- simCOP(1000, cop=MOcop, para=c(0.4,0.9)) # The parameters cause asymmetry.
mtext("Simulation from a defined Marshall-Olkin Copula")
the.grid <- EMPIRgrid(para=uv, deluv=0.025)
# EXAMPLE 2: PLOT NUMBER 2
# The second plot by image() will show a "hook" of sorts along the singularity.
image(the.grid$empcop, col=terrain.colors(40)) # Second plot is made
mtext("Image of gridded Empirical Copula")
# EXAMPLE 2: PLOT NUMBER 3
empcop <- EMPIRcopdf(para=uv) # data.frame of all points
# The third plot is the 3-D version overlain with the data points.
the.persp <- persp(x=the.grid$u, y=the.grid$v, z=the.grid$empcop, theta=240, phi=40,
                   xlab="U VARIABLE", ylab="V VARIABLE", zlab="COPULA C(u,v)"
) points(trans3d(empcop$u, empcop$v, empcop$empcop, the.persp),
       col=rgb(0,1-sqrt(empcop$empcop),1,sqrt(empcop$empcop)), pch=16)
# EXAMPLE 2: PLOT NUMBER 4
# The fourth plot shows a simulation and the quasi-emergence of the singularity
# that of course the empirical perspective "knows" nothing about. Do not use
# the Kumaraswamy smoothing because in this case the singularity because
# too smoothed out relative to the raw empirical, but of course the sample size
# is large enough to see such things. (Try kumaraswamy=TRUE)
empsim <- EMPIRsim(n=1000, empgrid = the.grid, kumaraswamy=FALSE)
mtext("Simulations from the Empirical Copula")
## End(Not run)

## Not run:
# EXAMPLE 3:
psp <- simCOP(n=4900, cop=PSP, ploton=FALSE, points=FALSE) * 150
# Pretend psp is real data, the * 150 is to clearly get into an arbitrary unit system.
# The sort=FALSE is critical in the following two calls to pp() from lmomco.
fakeU <- lmomco::pp(psp[,1], sort=FALSE) # Weibull plotting position i/(n+1)
fakeV <- lmomco::pp(psp[,2], sort=FALSE) # Weibull plotting position i/(n+1)

uv <- data.frame(U=fakeU, V=fakeV) # our U-statistics

# EXAMPLE 3: # PLOT NUMBER 1
deluv <- 0.0125 # going to cause long run time with large n
# The small deluv is used to explore solution quality at U=0 and 1.
the.grid <- EMPIRgrid(para=uv, deluv=deluv)
the.persp <- persp(x=the.grid$u, y=the.grid$v, z=the.grid$empcop, theta=-25, phi=20,
                   xlab="U VARIABLE", ylab="V VARIABLE", zlab="COPULA C(u,v)")

S <- sectionCOP(cop=PSP, 1, ploton=FALSE, lines=FALSE)
the.lines <- trans3d(rep(1, length(S$t)), S$t, S$seccop, the.persp)
lines(the.lines, lwd=2, col=2)

S <- sectionCOP(cop=PSP, 0, ploton=FALSE, lines=FALSE)
the.lines <- trans3d(rep(0, length(S$t)), S$t, S$seccop, the.persp)
lines(the.lines, lwd=2, col=2)

S <- sectionCOP(cop=PSP, 1, ploton=FALSE, lines=FALSE, dercop=TRUE)
the.lines <- trans3d(rep(1, length(S$t)), S$t, S$seccop, the.persp)
lines(the.lines, lwd=2, col=2, lty=2)

S <- sectionCOP(cop=PSP, 2*deluv, ploton=FALSE, lines=FALSE, dercop=TRUE)
the.lines <- trans3d(rep(2*deluv, length(S$t)), S$t, S$seccop, the.persp)
lines(the.lines, lwd=2, col=2, lty=2)

empder <- EMPIRgridder(empgrid=the.grid)
the.lines <- trans3d(rep(2*deluv,length(the.grid$v)),the.grid$v,empder[3,,],the.persp)
lines(the.lines, lwd=4, col=5, lty=2)

pdf("conditional_distributions.pdf")
ix <- 1:length(attributes(empder)$rownames)
for(i in ix) {
  u <- as.numeric(attributes(empder)$rownames[i])
  S <- sectionCOP(cop=PSP, u, ploton=FALSE, lines=FALSE, dercop=TRUE)
  # The red line is the true.
  plot(S$t, S$seccop, lwd=2, col=2, lty=2, type="l", xlim=c(0,1), ylim=c(0,1),
       xlab="V, NONEXCEEDANCE PROBABILITY", ylab="V, VALUE")
  lines(the.grid$v, empder[i,,], lwd=4, col=5, lty=2) # empirical
  mtext(paste("Conditioned on U="u," nonexceedance probability")

}
EMPIRgridder

Derivatives of the Grid of the Bivariate Empirical Copula for V with respect to U

Description

Generate derivatives of V with respect to U of a gridded representation of the bivariate empirical copula (see EMPIRcop). This function is the empirical analog to derCOP.

Usage

EMPIRgridder(empgrid=NULL, ...)

Arguments

empgrid The grid from EMPIRgrid; and
...

Additional arguments to pass.

Value

The gridded values of the derivatives of the bivariate empirical copula.

Author(s)

W.H. Asquith

See Also

EMPIRcop, EMPIRcopdf, EMPIRgrid, EMPIRgridder2

Examples

## Not run:
para <- list(alpha=0.15, beta=0.65, cop1=PLACKETTcop, cop2=PLACKETTcop, para1=0.005, para2=1000)
uv <- simCOP(n=1000, cop=composite2COP, para=para)
fakeU <- lmomco::pp(uv[,1], sort=FALSE)
fakeV <- lmomco::pp(uv[,2], sort=FALSE)
uv <- data.frame(U=fakeU, V=fakeV)

"trans3d" <- # blackslashes seem needed for the package
function(x,y,z, pmat) { # for user manual building but bad syntax
tmat <- cbind(x,y,z,1) %*% pmat # because remember the percent sign is a
return(tmat[,1:2] / tmat[,4]) # a comment character in LaTeX.
}
the.grid <- EMPIRgrid(para=uv, deluv=0.1)
the.diag <- diagCOP(cop=EMPIRcop, para=uv, ploton=FALSE, lines=FALSE)
empcop <- EMPIRcopdf(para=uv)  # data.frame of all points

the.persp <- persp(the.grid$empcop, theta=-25, phi=20,
xlab="U VARIABLE", ylab="V VARIABLE", zlab="COPULA C(u,v)"
points(trans3d(empcop$u, empcop$v, empcop$empcop, the.persp),
col=rgb(0,1-sqrt(empcop$empcop),1,sqrt(empcop$empcop)), pch=16, cex=0.75)

# Now extract the copula sections
some.lines <- trans3d(rep(0.2, length(the.grid$v)),
                        the.grid$v, the.grid$empcop[,3], the.persp)
lines(some.lines, lwd=2, col=2)
some.lines <- trans3d(the.grid$u, rep(0.6, length(the.grid$u)),
                        the.grid$empcop[,7], the.persp)
lines(some.lines, lwd=2, col=3)
some.lines <- trans3d(rep(0.7, length(the.grid$v)),
                        the.grid$v, the.grid$empcop[,8], the.persp)
lines(some.lines, lwd=2, col=6)

# Now compute some derivatives or conditional cumulative
# distribution functions
empder <- EMPIRgridder(empgrid=the.grid)
some.lines <- trans3d(rep(0.2, length(the.grid$v)), the.grid$v, empder[3,], the.persp)
lines(some.lines, lwd=4, col=2)

empder <- EMPIRgridder2(empgrid=the.grid)
some.lines <- trans3d(the.grid$u, rep(0.6, length(the.grid$u)), empder[,7], the.persp)
lines(some.lines, lwd=4, col=3)

empder <- EMPIRgridder(empgrid=the.grid)
some.lines <- trans3d(rep(0.7, length(the.grid$v)), the.grid$v, empder[8,], the.persp)
lines(some.lines, lwd=4, col=6)

# Demonstrate conditional quantile function extraction for
# the 70th percentile of U and see how it plots on top of
# the thick purple line
empinv <- EMPIRgridderinv(empgrid=the.grid)
some.lines <- trans3d(rep(0.7, length(the.grid$v)), empinv[8,],
                      attributes(empinv)$colnames, the.persp)
lines(some.lines, lwd=4, col=5, lty=2)

## End(Not run)
EMPIRgridderinv

Description

Generate derivatives of \( U \) with respect to \( V \) of a gridded representation of the bivariate empirical copula (see EMPIRcop). This function is the empirical analog to \texttt{derCOP2}.

Usage

\texttt{EMPIRgridder2(empgrid=NULL, \ldots)}

Arguments

\begin{itemize}
  \item \texttt{empgrid} The grid from \texttt{EMPIRgrid}; and
  \item \texttt{\ldots} Additional arguments to pass.
\end{itemize}

Value

The gridded values of the derivatives of the bivariate empirical copula.

Author(s)

W.H. Asquith

See Also

\texttt{EMPIRcop, EMPIRcopdf, EMPIRgrid, EMPIRgridder}

Examples

\begin{verbatim}
# See examples under EMPIRgridder
\end{verbatim}

---

EMPIRgridderinv \hspace{1cm} Derivative Inverses of the Grid of the Bivariate Empirical Copula for \( V \) with respect to \( U \)

Description

Generate a gridded representation of the inverse of the derivatives of the bivariate empirical copula of \( V \) with respect to \( U \). This function is the empirical analog to \texttt{derCOPinv}.

Usage

\texttt{EMPIRgridderinv(empgrid=NULL, kumaraswamy=FALSE, dergrid=NULL, \ldots)}
EMPIRgridderinv

Arguments

empgrid    The grid from EMPIRgrid;
kumaraswamy A logical to trigger Kumaraswamy smoothing of the conditional quantile function;
dergrid    The results of EMPIRgridders and if left NULL then that function is called internally. There is some fragility at times in the quality of the numerical derivative and the author has provided this argument so that the derivative can be computed externally and then fed to this inversion function; and
... Additional arguments to pass.

Value

The gridded values of the inverse of the derivative of $V$ with respect $U$.

Author(s)

W.H. Asquith

See Also

EMPIRcop, EMPIRcopdf, EMPIRgrid, EMPIRgridders2

Examples

## Not run:
uv <- simCOP(n=10000, cop=PSP, ploton=FALSE, points=FALSE)
fakeU <- lmomco::pp(uv[,1], sort=FALSE)
fakeV <- lmomco::pp(uv[,2], sort=FALSE)
uv <- data.frame(U=fakeU, V=fakeV)

uv.grid <- EMPIRgrid(para=uv, deluv=.1) # CPU hungry
uv.inv1 <- EMPIRgridderinv(empgrid=uv.grid)
uv.inv2 <- EMPIRgridderinv2(empgrid=uv.grid)
plot(uv, pch=16, col=rgb(0,0,0,.1), xlim=c(0,1), ylim=c(0,1),
     xlab="U, NONEXCEEDANCE PROBABILITY", ylab="V, NONEXCEEDANCE PROBABILITY")
lines(qua.regressCOP(f=0.5, cop=PSP), col=2)
lines(qua.regressCOP(f=0.2, cop=PSP), col=2)
lines(qua.regressCOP(f=0.7, cop=PSP), col=2)
lines(qua.regressCOP(f=0.1, cop=PSP), col=2)
lines(qua.regressCOP(f=0.9, cop=PSP), col=2)

med.wrtu <- EMPIRqua.regress(f=0.5, empinv=uv.inv1)
lines(med.wrtu, col=2, lwd=4)
qua.wrtu <- EMPIRqua.regress(f=0.2, empinv=uv.inv1)
lines(qua.wrtu, col=2, lwd=2, lty=2)
qua.wrtu <- EMPIRqua.regress(f=0.7, empinv=uv.inv1)
lines(qua.wrtu, col=2, lwd=2, lty=2)
qua.wrtu <- EMPIRqua.regress(f=0.1, empinv=uv.inv1)
lines(qua.wrtu, col=2, lwd=2, lty=4)
qua.wrtu <- EMPIRqua.regress(f=0.9, empinv=uv.inv1)
EMPIRgridderinv

```r
lines(qua.wrtu, col=2, lwd=2, lty=4)
lines(qua.regressCOP(f=0.5, cop=PSP), col=4)
lines(qua.regressCOP(f=0.2, cop=PSP), col=4)
lines(qua.regressCOP(f=0.7, cop=PSP), col=4)
lines(qua.regressCOP(f=0.1, cop=PSP), col=4)
lines(qua.regressCOP(f=0.9, cop=PSP), col=4)

med.wrtv <- EMPIRqua.regress2(f=0.5, empinv=uv.inv2)
lines(med.wrtv, col=4, lwd=4)
qua.wrtv <- EMPIRqua.regress2(f=0.2, empinv=uv.inv2)
lines(qua.wrtv, col=4, lwd=2, lty=2)
qua.wrtv <- EMPIRqua.regress2(f=0.7, empinv=uv.inv2)
lines(qua.wrtv, col=4, lwd=2, lty=2)
qua.wrtv <- EMPIRqua.regress2(f=0.1, empinv=uv.inv2)
lines(qua.wrtv, col=4, lwd=2, lty=4)
qua.wrtv <- EMPIRqua.regress2(f=0.9, empinv=uv.inv2)
lines(qua.wrtv, col=4, lwd=2, lty=4)

## Not run:
# Now try a much more complex shape
para <- list(alpha=0.15, beta=0.65, cop1=PLACKETTcop, cop2=PLACKETTcop,
             para1=0.005, para2=1000)
uv <- simCOP(n=30000, cop=composite2COP, para=para)
fakeU <- lmomco::pp(uv[,1], sort=FALSE)
fakeV <- lmomco::pp(uv[,2], sort=FALSE)

uv <- data.frame(U=fakeU, V=fakeV)

uv.grid <- EMPIRgrid(para=uv, deluv=0.05) # CPU hungry
uv.inv1 <- EMPIRgridderinv(empgrid=uv.grid)

plot(uv, pch=16, col=rgb(0,0,0,0.1), xlim=c(0,1), ylim=c(0,1),
     xlab="U, NONEXCEEDANCE PROBABILITY", ylab="V, NONEXCEEDANCE PROBABILITY")
lines(qua.regressCOP(f=0.5, cop=composite2COP, para=para), col=2)
lines(qua.regressCOP(f=0.2, cop=composite2COP, para=para), col=2)
lines(qua.regressCOP(f=0.7, cop=composite2COP, para=para), col=2)
lines(qua.regressCOP(f=0.1, cop=composite2COP, para=para), col=2)
lines(qua.regressCOP(f=0.9, cop=composite2COP, para=para), col=2)

med.wrtu <- EMPIRqua.regress(f=0.5, empinv=uv.inv1)
lines(med.wrtu, col=2, lwd=4)
qua.wrtu <- EMPIRqua.regress(f=0.2, empinv=uv.inv1)
lines(qua.wrtu, col=2, lwd=2, lty=2)
qua.wrtu <- EMPIRqua.regress(f=0.7, empinv=uv.inv1)
lines(qua.wrtu, col=2, lwd=2, lty=2)
qua.wrtu <- EMPIRqua.regress(f=0.1, empinv=uv.inv1)
lines(qua.wrtu, col=2, lwd=2, lty=4)
qua.wrtu <- EMPIRqua.regress(f=0.9, empinv=uv.inv1)
lines(qua.wrtu, col=2, lwd=2, lty=4)

lines(qua.regressCOP2(f=0.5, cop=composite2COP, para=para), col=4)
lines(qua.regressCOP2(f=0.2, cop=composite2COP, para=para), col=4)
```

EMPIRgridderinv2

**Derivative Inverses of the Grid of the Bivariate Empirical Copula for U with respect to V**

### Description

Generate a gridded representation of the inverse of the derivatives of the bivariate empirical copula of U with respect to V. This function is the empirical analog to `derCOPinv2`.

### Usage

```r
EMPIRgridderinv2(empgrid=NULL, kumaraswamy=FALSE, dergrid=NULL, ...)
```

### Arguments

- **empgrid**: The grid from `EMPIRgrid`;
- **kumaraswamy**: A logical to trigger Kumaraswamy smoothing of the conditional quantile function;
- **dergrid**: The results of `EMPIRgridder2` and if left NULL then that function is called internally. There is some fragility at times in the quality of the numerical derivative and the author has provided this argument so that the derivative can be computed externally and then fed to this inversion function; and
- **...**: Additional arguments to pass.

### Value

The gridded values of the inverse of the derivative of U with respect to V.

### Author(s)

W.H. Asquith
EMPIRmed.regress

See Also

EMPIRcop, EMPIRcopdf, EMPIRgrid, EMPIRgridder2, EMPIRgridderinv, EMPIRgridderinv2

Examples

# See examples under EMPIRgridderinv

EMPIRmed.regress

Median Regression of the Grid of the Bivariate Empirical Copula for V with respect to U

Description

Perform median regression from the gridded inversion of the bivariate empirical copula of V with respect to U.

Usage

EMPIRmed.regress(...)

Arguments

... Arguments to pass to EMPIRqua.regress.

Value

The gridded values of the median regression of V with respect to U.

Author(s)

W.H. Asquith

See Also

EMPIRgridderinv, EMPIRqua.regress, EMPIRmed.regress2, EMPIRmed.regress2

Examples

# See examples under EMPIRqua.regress
EMPIRmed.regress2  
**Median Regression of the Grid of the Bivariate Empirical Copula for**  
*U* with respect to *V*

**Description**
Perform *median regression* from the gridded inversion of the *bivariate empirical copula* of *U* with respect to *V*.

**Usage**
```r
EMPIRmed.regress2(...)  
```

**Arguments**
...  
Arguments to pass to `EMPIRqua.regress2`.

**Value**
The gridded values of the median regression of *U* with respect to *V*.

**Author(s)**
W.H. Asquith

**See Also**
- `EMPIRgridderv2`  
- `EMPIRqua.regress`  
- `EMPIRmed.regress`  
- `EMPIRmed.regress2`

**Examples**
```r
# See examples under EMPIRqua.regress
```

EMPIRqua.regress  
**Quantile Regression of the Grid of the Bivariate Empirical Copula for**  
*V* with respect to *U*

**Description**
Perform *quantile regression* from the gridded inversion of the *bivariate empirical copula* of *V* with respect to *U*.

**Usage**
```r
EMPIRqua.regress(f=0.5, u=seq(0.01,0.99, by=0.01), empinv=NULL,  
lowess=FALSE, f.lowess=1/5, ...)  
```
Arguments

- **f**: The nonexceedance probability $F$ to perform regression at and defaults to median regression $F = 1/2$;
- **u**: A vector of $u$ nonexceedance probabilities;
- **empinv**: The grid from `EMPIRgridderinv`;
- **lowess**: Perform lowess smooth on the quantile regression using the smooth factor of **f.lowess**;
- **f.lowess**: Smooth factor of almost the same argument name fed to the `lowess()` function in R; and
- **...**: Additional arguments to pass.

Value

The gridded values of the quantile regression of $V$ with respect to $U$.

Author(s)

W.H. Asquith

See Also

`EMPIRgridderinv`, `EMPIRqua.regress2`, `EMPIRmed.regress`, `EMPIRmed.regress2`

Examples

```R
## Not run: # EXAMPLE 1
theta <- 25
uv <- simCOP(n=1000, cop=PLACKETTcop, para=theta, ploton=FALSE, points=FALSE)
fakeU <- lmomco::pp(uv[,1], sort=FALSE)
fakeV <- lmomco::pp(uv[,2], sort=FALSE)
uv <- data.frame(U=fakeU, V=fakeV)

uv.grid <- EMPIRgrid(para=uv, deluv=0.05) # CPU hungry
uv.inv1 <- EMPIRgridderinv(empgrid=uv.grid)
plot(uv, pch=16, col=rgb(0,0,0,.1), xlim=c(0,1), ylim=c(0,1),
     xlab="U, NONEXCEEDANCE PROBABILITY", ylab="V, NONEXCEEDANCE PROBABILITY")
lines(qua.regressCOP(f=0.5, cop=PLACKETTcop, para=theta), lwd=2)
lines(qua.regressCOP(f=0.2, cop=PLACKETTcop, para=theta), lwd=2)
lines(qua.regressCOP(f=0.7, cop=PLACKETTcop, para=theta), lwd=2)
lines(qua.regressCOP(f=0.1, cop=PLACKETTcop, para=theta), lwd=2)
lines(qua.regressCOP(f=0.9, cop=PLACKETTcop, para=theta), lwd=2)

med.wrtu <- EMPIRqua.regress(f=0.5, empinv=uv.inv1, lowess=TRUE, f.lowess=0.1)
lines(med.wrtu, col=2, lwd=4)
qua.wrtu <- EMPIRqua.regress(f=0.2, empinv=uv.inv1, lowess=TRUE, f.lowess=0.1)
lines(qua.wrtu, col=2, lwd=2, lty=2)
qua.wrtu <- EMPIRqua.regress(f=0.7, empinv=uv.inv1, lowess=TRUE, f.lowess=0.1)
lines(qua.wrtu, col=2, lwd=2, lty=2)
qua.wrtu <- EMPIRqua.regress(f=0.1, empinv=uv.inv1, lowess=TRUE, f.lowess=0.1)
```
EMPIRqua.regress

```r
lines(qua.wrtu, col=2, lwd=2, lty=4)
qua.wrtu <- EMPIRqua.regress(f=0.9, empinv=uv.inv1, lowess=TRUE, f.lowess=0.1)
lines(qua.wrtu, col=2, lwd=2, lty=4)

library(quantreg) # Quantile Regression by quantreg
U <- seq(0.01, 0.99, by=0.01)
rqlm <- rq(V~U, data=uv, tau=0.1)
rqlm <- rq(V~U, data=uv, tau=0.2)
rq.2 <- rqlm$coefficients[1] + rqlm$coefficients[2]*U
rqlm <- rq(V~U, data=uv, tau=0.5)
rq.5 <- rqlm$coefficients[1] + rqlm$coefficients[2]*U
rqlm <- rq(V~U, data=uv, tau=0.7)
rq.7 <- rqlm$coefficients[1] + rqlm$coefficients[2]*U
rqlm <- rq(V~U, data=uv, tau=0.9)
rq.9 <- rqlm$coefficients[1] + rqlm$coefficients[2]*U

lines(U, rq.1, col=4, lwd=2, lty=4)
lines(U, rq.2, col=4, lwd=2, lty=2)
lines(U, rq.5, col=4, lwd=4)
lines(U, rq.7, col=4, lwd=2, lty=2)
lines(U, rq.9, col=4, lwd=2, lty=4)
# End(Not run)

## Not run: # EXAMPLE 2
# Start again with the PSP copula
uv <- simCOP(n=10000, cop=PSP, ploton=FALSE, points=FALSE)
fakeU <- lmomco::pp(uv[,1], sort=FALSE)
fakeV <- lmomco::pp(uv[,2], sort=FALSE)
uv <- data.frame(U=fakeU, V=fakeV)

uv.grid <- EMPIRgrid(para=uv, deluv=0.05) # CPU hungry
uv.inv1 <- EMPIRgridderinv(empgrid=uv.grid)
plot(uv, pch=16, col=rgb(0,0,0,0.1), xlim=c(0,1), ylim=c(0,1),
     xlab="U, NONEXCEEDANCE PROBABILITY", ylab="V, NONEXCEEDANCE PROBABILITY")
lines(qua.regressCOP(f=0.5, cop=PSP), lwd=2)
lines(qua.regressCOP(f=0.2, cop=PSP), lwd=2)
lines(qua.regressCOP(f=0.7, cop=PSP), lwd=2)
lines(qua.regressCOP(f=0.1, cop=PSP), lwd=2)
lines(qua.regressCOP(f=0.9, cop=PSP), lwd=2)

med.wrtu <- EMPIRqua.regress(f=0.5, empinv=uv.inv1, lowess=TRUE, f.lowess=0.1)
lines(med.wrtu, col=2, lwd=4)
qua.wrtu <- EMPIRqua.regress(f=0.2, empinv=uv.inv1, lowess=TRUE, f.lowess=0.1)
lines(qua.wrtu, col=2, lwd=2, lty=2)
qua.wrtu <- EMPIRqua.regress(f=0.7, empinv=uv.inv1, lowess=TRUE, f.lowess=0.1)
lines(qua.wrtu, col=2, lwd=2, lty=2)
qua.wrtu <- EMPIRqua.regress(f=0.1, empinv=uv.inv1, lowess=TRUE, f.lowess=0.1)
lines(qua.wrtu, col=2, lwd=2, lty=4)
qua.wrtu <- EMPIRqua.regress(f=0.9, empinv=uv.inv1, lowess=TRUE, f.lowess=0.1)
lines(qua.wrtu, col=2, lwd=2, lty=4)

library(quantreg) # Quantile Regression by quantreg
```
```r
U <- seq(0.01, 0.99, by=0.01)
rlm <- rq(V~U, data=uv, tau=0.1)
rlm <- rq(V~U, data=uv, tau=0.2)
rlm.2 <- rlm$coefficients[1] + rlm$coefficients[2]*U
rlm <- rq(V~U, data=uv, tau=0.5)
rlm.5 <- rlm$coefficients[1] + rlm$coefficients[2]*U
rlm <- rq(V~U, data=uv, tau=0.7)
rlm.7 <- rlm$coefficients[1] + rlm$coefficients[2]*U
rlm <- rq(V~U, data=uv, tau=0.9)
rlm.9 <- rlm$coefficients[1] + rlm$coefficients[2]*U
lines(U, rlm.1, col=4, lwd=2, lty=4)
lines(U, rlm.2, col=4, lwd=2, lty=2)
lines(U, rlm.5, col=4, lwd=4)
lines(U, rlm.7, col=4, lwd=2, lty=2)
lines(U, rlm.9, col=4, lwd=2, lty=4)

## End(Not run)

## Not run: # EXAMPLE 3
uv <- simCOP(n=10000, cop=PSP, ploton=FALSE, points=FALSE)
fakeU <- lmomco::pp(uv[,1], sort=FALSE)
fakeV <- lmomco::pp(uv[,2], sort=FALSE)
uv <- data.frame(U=fakeU, V=fakeV)

uv.grid <- EMPIRgrid(para=uv, deluv=0.1) # CPU hungry
uv.inv1 <- EMPIRgridderinv(empgrid=uv.grid)

plot(uv, pch=16, col=rgb(0,0,0,.1), xlim=c(0,1), ylim=c(0,1),
xlab="U, NONEXCEEDANCE PROBABILITY", ylab="V, NONEXCEEDANCE PROBABILITY")
lines(qua.regressCOP(f=0.5, cop=PSP), col=2)
lines(qua.regressCOP(f=0.2, cop=PSP), col=2)
lines(qua.regressCOP(f=0.7, cop=PSP), col=2)
lines(qua.regressCOP(f=0.1, cop=PSP), col=2)
lines(qua.regressCOP(f=0.9, cop=PSP), col=2)

med.wrtu <- EMPIRqua.regress(f=0.5, empinv=uv.inv1)
lines(med.wrtu, col=2, lwd=4)
quawrtu <- EMPIRqua.regress(f=0.2, empinv=uv.inv1)
lines(quawrtu, col=2, lwd=2, lty=2)
quawrtu <- EMPIRqua.regress(f=0.7, empinv=uv.inv1)
lines(quawrtu, col=2, lwd=2, lty=2)
quawrtu <- EMPIRqua.regress(f=0.1, empinv=uv.inv1)
lines(quawrtu, col=2, lwd=2, lty=4)
quawrtu <- EMPIRqua.regress(f=0.9, empinv=uv.inv1)
lines(quawrtu, col=2, lwd=2, lty=4)

lines(qua.regressCOP2(f=0.5, cop=PSP), col=4)
lines(qua.regressCOP2(f=0.2, cop=PSP), col=4)
lines(qua.regressCOP2(f=0.7, cop=PSP), col=4)
lines(qua.regressCOP2(f=0.1, cop=PSP), col=4)
lines(qua.regressCOP2(f=0.9, cop=PSP), col=4)
```

---

**EMPIRqua.regress**

103
EMPIRqua.regress

med.wrtv <- EMPIRqua.regress2(f=0.5, empinv=uv.inv2)
lines(med.wrtv, col=4, lwd=4)
qua.wrtv <- EMPIRqua.regress2(f=0.2, empinv=uv.inv2)
lines(qua.wrtv, col=4, lwd=2, lty=2)
qua.wrtv <- EMPIRqua.regress2(f=0.7, empinv=uv.inv2)
lines(qua.wrtv, col=4, lwd=2, lty=2)
qua.wrtv <- EMPIRqua.regress2(f=0.1, empinv=uv.inv2)
lines(qua.wrtv, col=4, lwd=2, lty=4)
qua.wrtv <- EMPIRqua.regress2(f=0.9, empinv=uv.inv2)
lines(qua.wrtv, col=4, lwd=2, lty=4)

## Not run:  # EXAMPLE 4
# Now try a much more complex shape
# lowess smoothing on quantile regression is possible,
# see next example
para <- list(alpha=0.15, beta=0.65,
        cop1=PLACKETTcop, cop2=PLACKETTcop, para1=0.005, para2=1000)
uv <- simCOP(n=20000, cop=composite2COP, para=para)
fakeU <- lmomco::pp(uv[,1], sort=FALSE)
fakeV <- lmomco::pp(uv[,2], sort=FALSE)
uv <- data.frame(U=fakeU, V=fakeV)

uv.grid <- EMPIRgrid(para=uv, deluv=0.025) # CPU hungry
uv.inv1 <- EMPIRgridderinv(empgrid=uv.grid)
uv.inv2 <- EMPIRgridderinv2(empgrid=uv.grid)
plot(uv, pch=16, col=rgb(0,0,0,0.1), xlim=c(0,1), ylim=c(0,1),
    xlab="U, NONEXCEEDANCE PROBABILITY", ylab="V, NONEXCEEDANCE PROBABILTIY")
lines(qua.regressCOP(f=0.5, cop=composite2COP, para=para), col=2)
lines(qua.regressCOP(f=0.2, cop=composite2COP, para=para), col=2)
lines(qua.regressCOP(f=0.7, cop=composite2COP, para=para), col=2)
lines(qua.regressCOP(f=0.1, cop=composite2COP, para=para), col=2)
lines(qua.regressCOP(f=0.9, cop=composite2COP, para=para), col=2)

med.wrtu <- EMPIRqua.regress(f=0.5, empinv=uv.inv1)
lines(med.wrtu, col=2, lwd=4)
qua.wrtu <- EMPIRqua.regress(f=0.2, empinv=uv.inv1)
lines(qua.wrtu, col=2, lwd=2, lty=2)
qua.wrtu <- EMPIRqua.regress(f=0.7, empinv=uv.inv1)
lines(qua.wrtu, col=2, lwd=2, lty=2)
qua.wrtu <- EMPIRqua.regress(f=0.1, empinv=uv.inv1)
lines(qua.wrtu, col=2, lwd=2, lty=4)
qua.wrtu <- EMPIRqua.regress(f=0.9, empinv=uv.inv1)
lines(qua.wrtu, col=2, lwd=2, lty=4)

lines(qua.regressCOP2(f=0.5, cop=composite2COP, para=para), col=4)
lines(qua.regressCOP2(f=0.2, cop=composite2COP, para=para), col=4)
lines(qua.regressCOP2(f=0.7, cop=composite2COP, para=para), col=4)
lines(qua.regressCOP2(f=0.1, cop=composite2COP, para=para), col=4)
lines(qua.regressCOP2(f=0.9, cop=composite2COP, para=para), col=4)
med.wrtu <- EMPIRqua.regress2(f=0.5, empinv=uv.inv2)
lines(med.wrtu, col=4, lwd=4)
EMPIRqua.regress

qua.wrtv <- EMPIRqua.regress2(f=0.2, empinv=uv.inv2)
lines(qua.wrtv, col=4, lwd=2, lty=2)
qua.wrtv <- EMPIRqua.regress2(f=0.7, empinv=uv.inv2)
lines(qua.wrtv, col=4, lwd=2, lty=2)
qua.wrtv <- EMPIRqua.regress2(f=0.1, empinv=uv.inv2)
lines(qua.wrtv, col=4, lwd=2, lty=4)
qua.wrtv <- EMPIRqua.regress2(f=0.9, empinv=uv.inv2)
lines(qua.wrtv, col=4, lwd=2, lty=4)

## End(Not run)

## Not run: # EXAMPLE 5
plot(uv, pch=16, col=rgb(0,0,0,.1), xlim=c(0,1), ylim=c(0,1),
    xlab="U, NONEXCEEDANCE PROBABILITY", ylab="V, NONEXCEEDANCE PROBABILITY")
lines(qua.regressCOP(f=0.5, cop=composite2COP, para=para), col=2)
lines(qua.regressCOP(f=0.2, cop=composite2COP, para=para), col=2)
lines(qua.regressCOP(f=0.7, cop=composite2COP, para=para), col=2)
lines(qua.regressCOP(f=0.1, cop=composite2COP, para=para), col=2)
lines(qua.regressCOP(f=0.9, cop=composite2COP, para=para), col=2)
med.wrtu <- EMPIRqua.regress(f=0.5, empinv=uv.inv1, lowess=TRUE)
lines(med.wrtu, col=2, lwd=4)
qua.wrtu <- EMPIRqua.regress(f=0.2, empinv=uv.inv1, lowess=TRUE)
lines(qua.wrtu, col=2, lwd=2, lty=2)
qua.wrtu <- EMPIRqua.regress(f=0.7, empinv=uv.inv1, lowess=TRUE)
lines(qua.wrtu, col=2, lwd=2, lty=2)
qua.wrtu <- EMPIRqua.regress(f=0.1, empinv=uv.inv1, lowess=TRUE)
lines(qua.wrtu, col=2, lwd=2, lty=4)
qua.wrtu <- EMPIRqua.regress(f=0.9, empinv=uv.inv1, lowess=TRUE)
lines(qua.wrtu, col=2, lwd=2, lty=4)
lines(qua.regressCOP2(f=0.5, cop=composite2COP, para=para), col=4)
lines(qua.regressCOP2(f=0.2, cop=composite2COP, para=para), col=4)
lines(qua.regressCOP2(f=0.7, cop=composite2COP, para=para), col=4)
lines(qua.regressCOP2(f=0.1, cop=composite2COP, para=para), col=4)
lines(qua.regressCOP2(f=0.9, cop=composite2COP, para=para), col=4)
med.wrtv <- EMPIRqua.regress2(f=0.5, empinv=uv.inv2, lowess=TRUE)
lines(med.wrtv, col=4, lwd=4)
qua.wrtv <- EMPIRqua.regress2(f=0.2, empinv=uv.inv2, lowess=TRUE)
lines(qua.wrtv, col=4, lwd=2, lty=2)
qua.wrtv <- EMPIRqua.regress2(f=0.7, empinv=uv.inv2, lowess=TRUE)
lines(qua.wrtv, col=4, lwd=2, lty=2)
qua.wrtv <- EMPIRqua.regress2(f=0.1, empinv=uv.inv2, lowess=TRUE)
lines(qua.wrtv, col=4, lwd=2, lty=4)
qua.wrtv <- EMPIRqua.regress2(f=0.9, empinv=uv.inv2, lowess=TRUE)
lines(qua.wrtv, col=4, lwd=2, lty=4)

## End(Not run)

## Not run: # EXAMPLE 6 (changing the smoothing on the lowess)
plot(uv, pch=16, col=rgb(0,0,0,.1), xlim=c(0,1), ylim=c(0,1),
    xlab="U, NONEXCEEDANCE PROBABILITY", ylab="V, NONEXCEEDANCE PROBABILITY")
lines(qua.regressCOP(f=0.5, cop=composite2COP, para=para), col=2)
lines(qua.regressCOP(f=0.2, cop=composite2COP, para=para), col=2)
EMPIRqua.regress2

Quantile Regression of the Grid of the Bivariate Empirical Copula for U with respect to V

Description

Generate quantile regression from the gridded inversion of the bivariate empirical copula of U with respect to V.

Usage

EMPIRqua.regress2(f=0.5, v=seq(0.01,0.99, by=0.01), empinv=NULL, lowess=FALSE, f.lowess=1/5, ...)

EMPIRqua.regress2(f=0.7, cop=composite2COP, para=para), col=2)
lines(qua.regressCOP(f=0.1, cop=composite2COP, para=para), col=2)
lines(qua.regressCOP(f=0.9, cop=composite2COP, para=para), col=2)

med.wrtu <- EMPIRqua.regress(f=0.5, empinv=uv.inv1, lowess=TRUE, f.lowess=0.1)
lines(med.wrtu, col=2, lwd=4)
qua.wrtu <- EMPIRqua.regress(f=0.2, empinv=uv.inv1, lowess=TRUE, f.lowess=0.1)
lines(qua.wrtu, col=2, lwd=2, lty=2)
qua.wrtu <- EMPIRqua.regress(f=0.7, empinv=uv.inv1, lowess=TRUE, f.lowess=0.1)
lines(qua.wrtu, col=2, lwd=2, lty=2)
qua.wrtu <- EMPIRqua.regress(f=0.1, empinv=uv.inv1, lowess=TRUE, f.lowess=0.1)
lines(qua.wrtu, col=2, lwd=2, lty=4)
qua.wrtu <- EMPIRqua.regress(f=0.9, empinv=uv.inv1, lowess=TRUE, f.lowess=0.1)
lines(qua.wrtu, col=2, lwd=2, lty=4)

lines(qua.regressCOP2(f=0.5, cop=composite2COP, para=para), col=4)
lines(qua.regressCOP2(f=0.2, cop=composite2COP, para=para), col=4)
lines(qua.regressCOP2(f=0.7, cop=composite2COP, para=para), col=4)
lines(qua.regressCOP2(f=0.1, cop=composite2COP, para=para), col=4)
lines(qua.regressCOP2(f=0.9, cop=composite2COP, para=para), col=4)

med.wrtv <- EMPIRqua.regress2(f=0.5, empinv=uv.inv2, lowess=TRUE, f.lowess=0.1)
lines(med.wrtv, col=4, lwd=4)
qua.wrtv <- EMPIRqua.regress2(f=0.2, empinv=uv.inv2, lowess=TRUE, f.lowess=0.1)
lines(qua.wrtv, col=4, lwd=2, lty=2)
qua.wrtv <- EMPIRqua.regress2(f=0.7, empinv=uv.inv2, lowess=TRUE, f.lowess=0.1)
lines(qua.wrtv, col=4, lwd=2, lty=2)
qua.wrtv <- EMPIRqua.regress2(f=0.1, empinv=uv.inv2, lowess=TRUE, f.lowess=0.1)
lines(qua.wrtv, col=4, lwd=2, lty=4)
qua.wrtv <- EMPIRqua.regress2(f=0.9, empinv=uv.inv2, lowess=TRUE, f.lowess=0.1)
lines(qua.wrtv, col=4, lwd=2, lty=4)#
## End(Not run)
Arguments

- **f**: The nonexceedance probability $F$ to perform regression at and defaults to median regression $F = 1/2$;
- **v**: A vector of $v$ nonexceedance probabilities;
- **empinv**: The grid from `EMPIRgridderinv`;
- **lowess**: Perform lowess smooth on the quantile regression using the smooth factor of $f=f.lowess$;
- **f.lowess**: Smooth factor of almost the same argument name fed to the lowess() function in R;
- **...**: Additional arguments to pass.

Value

The gridded values of the quantile regression of $U$ with respect to $V$.

Author(s)

W.H. Asquith

See Also

`EMPIRgridderinv2,EMPIRqua.regress,EMPIRmed.regress,EMPIRmed.regress2`

Examples

```r
# See examples under EMPIRqua.regress
```

---

**EMPIRsim**  
*Simulate a Bivariate Empirical Copula*

Description

*EXPERIMENTAL*—Perform a simulation on a *bivariate empirical copula* to produce the random variates $U$ and $V$ and return an R data.frame of them. The method is more broadly known as *conditional simulation method*. This function is an empirical parallel to `simCOP` that is used for parametric copulas. If circumstances require conditional simulation of $V|U$, then function `EMPIRsimg`, which produces a vector of $V$ from a fixed $u$, should be used.

For the usual situation in which an individual $u$ during the simulation loops is not a value aligned on the grid, then the bounding conditional quantile functions are solved for each of the $n$ simulations and the following interpolation is made by

$$v = \frac{v_1/w_1 + v_2/w_2}{1/w_1 + 1/w_2},$$

which states that the weighted mean is computed. The values $v_1$ and $v_2$ are ordinates of the conditional quantile function for the respective grid lines to the left and right of the $u$ value. The values $w_1 = u - u_{grid}^{left}$ and $w_2 = u_{grid}^{right} - u$. 
Usage

`EMPIRsim(n=100, empgrid=NULL, kumaraswamy=FALSE, na.rm=TRUE, keep=FALSE, graphics=TRUE, ploton=TRUE, points=TRUE, snv=FALSE, infsnv.rm=TRUE, trapinfsnv=.Machine$double.eps, ...)`

Arguments

- **n**: A sample size, default is 100;
- **empgrid**: Gridded empirical copula from `EMPIRgrid`;
- **kumaraswamy**: A logical to trigger Kumaraswamy distribution smoothing of the conditional quantile function that is passed to `EMPIRgridderinv`. The Kumaraswamy distribution is a distribution having support \([0,1]\) with an explicit quantile function and takes the place of a Beta distribution (see `lmomco` function `quakur()` for more details);
- **na.rm**: A logical to toggle the removal of NA entries on the returned data.frame;
- **keep**: Keep the \(t\) uniform random variable for the simulation as the last column in the returned data.frame;
- **graphics**: A logical that will disable graphics by setting ploton and points to FALSE and overriding whatever their settings were;
- **ploton**: A logical to toggle on the plot;
- **points**: A logical to actually draw the simulations by the `points()` function in R;
- **snv**: A logical to convert the \(\{u,v\}\) to standard normal scores (variates) both for the optional graphics and the returned data.frame. Joe (2014) advocates extensively for use of normal scores, which is in contrast to Nelsen (2006) who does not;
- **infsnv.rm**: A logical that will quietly strip out any occurrences of \(u = \{0,1\}\) or \(v = \{0,1\}\) from the simulations because these are infinity in magnitude when converted to standard normal variates is to occur. Thus, this logical only impacts logic flow when snv is TRUE. The `infsnv.rm` is mutually exclusive from `trapinfsnv`;
- **trapinfsnv**: If TRUE and presumably small, the numerical value of this argument \(\eta\) is used to replace \(u = \{0,1\}\) and \(v = \{0,1\}\) with \(u(0)=v(0)=\eta\) or \(u(1)=v(1)=1-\eta\) as appropriate when conversion to standard normal variates is to occur. The setting of `trapinfsnv` only is used if snv is TRUE and `infsnv.rm` is FALSE; and
- **...**: Additional arguments to pass to the `points()` function in R.

Value

An R data.frame of the simulated values is returned.

Author(s)

W.H. Asquith
EMPIRsimv

See Also
EMPIRgrid, EMPIRgridderinv, EMPIRsimv

Examples

# See other examples under EMPIRsimv

## Not run:

pdf("EMPIRsim_experiment.pdf")
nsim <- 5000
para <- list(alpha=0.15, beta=0.65,
     cop1=PLACKETTcop, cop2=PLACKETTcop, para1=0.005, para2=1000)
set.seed(1)
uv <- simCOP(n=nsim, cop=composite2COP, para=para, snv=TRUE,
     pch=16, col=rgb(0,0,0,.2))
mtext("A highly complex simulated bivariate relation")
# set.seed() # try not resetting the seed
uv.grid <- EMPIRgrid(para=uv, deluv=0.025)

uv2 <- EMPIRsim(n=nsim, empgrid=uv.grid, kumaraswamy=FALSE, snv=TRUE,
    col=rgb(1,0,0,0.1), pch=16)
mtext("Resimulation without Kumaraswamy smoothing")

uv3 <- EMPIRsim(n=nsim, empgrid=uv.grid, kumaraswamy=TRUE, snv=TRUE,
    col=rgb(1,0,0,0.1),pch=16)
mtext("Resimulation but using the Kumaraswamy Distribution for smoothing")
dev.off()#
## End(Not run)

EMPIRsimv

Simulate a Bivariate Empirical Copula For a Fixed Value of U

Description

*EXPERIMENTAL*—Perform a simulation on a *bivariate empirical copula* to extract the random variates \( V \) from a given and fixed value for \( u = \) constant. The purpose of this function is to return a simple vector of the \( V \) simulations. This behavior is similar to \( \text{simCOP} \) but differs from the general 2-D simulation implemented in the other functions: \( \text{EMPIRsim} \) and \( \text{simCOP} \)—these two functions generate R data.frames of simulated random variates \( U \) and \( V \) and optional graphics as well.

For the usual situation in which \( u \) is not a value aligned on the grid, then the bounding conditional quantile functions are solved for each of the \( n \) simulations and the following interpolation is made by

\[
    v = \frac{v_1/w_1 + v_2/w_2}{1/w_1 + 1/w_2},
\]

which states that that the weighted mean is computed. The values \( v_1 \) and \( v_2 \) are ordinates of the conditional quantile function for the respective grid lines to the left and right of the \( u \) value. The values \( w_1 = u - u_{\text{grid}}^{\text{left}} \) and \( w_2 = u_{\text{grid}}^{\text{right}} - u \).
Usage

EMPIRsimv(u, n=1, empgrid=NULL, kumaraswamy=FALSE, ...)

Arguments

u
The fixed probability \( u \) on which to perform conditional simulation for a sample of size \( n \);

n
A sample size, default is 1;

empgrid
Gridded empirical copula from EMPIRgrid;

kumaraswamy
A logical to trigger Kumaraswamy distribution smoothing of the conditional quantile function that is passed to EMPIRgridderinv. The Kumaraswamy distribution is a distribution having support \([0, 1]\) with an explicit quantile function and takes the place of a Beta distribution (see lmomco function quakur() for more details); and

... Additional arguments to pass.

Value

A vector of simulated \( V \) values is returned.

Author(s)

W.H. Asquith

See Also

EMPIRgrid, EMPIRsim

Examples

## Not run:

nsim <- 3000
para <- list(alpha=0.15, beta=0.65,
      cop1=PLACKETTcop, cop2=PLACKETTcop, para1=.005, para2=1000)
set.seed(10)
uv <- simCOP(n=nsim, cop=composite2COP, para=para, pch=16, col=rgb(0,0,0,0.2))
uv.grid <- EMPIRgrid(para=uv, deluv=.1)
set.seed(1)
V1 <- EMPIRsimv(u=0.6, n=nsim, empgrid=uv.grid)
set.seed(1)
V2 <- EMPIRsimv(u=0.6, n=nsim, empgrid=uv.grid, kumaraswamy=TRUE)
plot(V1,V2)
abline(0,1)

invgrid1 <- EMPIRgridderinv(empgrid=uv.grid)
invgrid2 <- EMPIRgridderinv(empgrid=uv.grid, kumaraswamy=TRUE)
att <- attributes(invgrid2); kur <- att$kumaraswamy
# Now draw random variates from the Kumaraswamy distribution using
# rlmomco() and vec2par() provided by the lmomco package.
FGMcop

The Generalized Farlie–Gumbel–Morgenstern Copula

Description

The generalized Farlie–Gumbel–Morgenstern copula (Bekrizade et al., 2012) is

\[ C_{\Theta, \alpha, n}(u, v) = FGM(u, v) = uv[1 + \Theta(1 - u^\alpha)(1 - v^\alpha)]^n, \]

where \( \Theta \in [-\min\{1, 1/(\alpha^2)\}, 1/(\alpha)] \), \( \alpha > 0 \), and \( n \in 0, 1, 2, \ldots \). The copula \( \Theta = 0 \) or \( \alpha = 0 \) or \( n = 0 \) becomes the independence copula \((\Pi(u, v); P)\). When \( \alpha = n = 1 \), then the well-known, single-parameter Farlie–Gumbel–Morgenstern copula results, and Spearman Rho \((\rho_{COP})\) is \( \rho_C = \Theta/3 \) but in general

\[ \rho_C = 12 \sum_{r=1}^{n} \binom{n}{r} \Theta^r \left[ \frac{\Gamma(r+1)\Gamma(2/\alpha)}{\alpha \Gamma(r+1+2/\alpha)} \right]^2. \]
The support of $\rho_C(\cdots; \Theta, 1, 1)$ is $[-1/3, +1/3]$ but extends via $\alpha$ and $n$ to $\approx [-0.50, +0.43]$, which shows that the generalization of the copula increases the range of dependency. The generalized version is implemented by FGMcop.

The *iterated Farlie–Gumbel–Morgenstern copula* (Chine and Benatia, 2017) for the $r$th iteration is

$$C_\beta(u, v) = FGM_i(u, v) = uv + \sum_{j=1}^{r} \beta_j \cdot (uv)^{[j/2]} \cdot (u'v')^{[(j+1)/2]} ,$$

where $u' = 1 - u$ and $v' = 1 - v$ for $|\beta_j| \leq 1$ that has $r$ dimensions $\beta = (\beta_1, \cdots, \beta_j, \cdots, \beta_r)$ and $[t]$ is the integer part of $t$. The copula $\beta = 0$ becomes the *independence copula* $(\Pi(u, v); P)$. The support of $\rho_C(\cdots; \beta)$ is approximately $[-0.43, +0.43]$. The iterated version is implemented by FGMicop. Internally, the $r$ is determined from the length of the $\beta$ in the para argument.

Usage

FGMcop( u, v, para=c(NA, 1,1), ...)  
FGMicop(u, v, para=NULL, ...)

Arguments

- **u** Nonexceedance probability $u$ in the $X$ direction;
- **v** Nonexceedance probability $v$ in the $Y$ direction;
- **para** A vector of parameters. For the generalized version, the $\Theta$, $\alpha$, and $n$ of the copula where the default argument shows the need to include the $\Theta$. However, if a fourth parameter is present, it is treated as a logical to reverse the copula $(u + v - 1 + FGM(1 - u, 1 - v; \Theta, \alpha, n))$. Also if a single parameter is given, then the $\alpha = n = 1$ are automatically set to produce the single-parameter Farlie–Gumbel–Morgenstern copula. For the iterated version, the $\beta$ vector of $r$ iterations;
- ... Additional arguments to pass.

Value

Value(s) for the copula are returned.

Author(s)

W.H. Asquith

References


See Also

P, mleCOP
Examples

```r
## Not run:
# Bekrizade et al. (2012, table 1) report for a=2 and n=3 that range in
# theta = [-0.1667, 0.1667] and range in rho = [-0.1806641, 0.4036458]. However,
# we see that they have seemingly made an error in listing the lower bounds of theta:
rhoCOP(FGMcop, para=c( 1/6, 2, 3)) # 0.4036458
rhoCOP(FGMcop, para=c( -1/6, 2, 3)) # Following error results
# In cop(u, v, para = para, ...): parameter Theta < -0.0833333333333333
rhoCOP(FGMcop, para=c(-1/12, 2, 3)) # -0.1806641
## End(Not run)
```

```r
## Not run:
# Support of FGMrcop(): first for r=1 iterations and then for large r.
sapply(c(-1, 1), function(t) rhoCOP(cop=FGMrcop, para=rep(t, 1)) )
# [1] 0.3333333 0.3333333
sapply(c(-1, 1), function(t) rhoCOP(cop=FGMrcop, para=rep(t,50)) )
# [1] -0.4341385 0.4341385
## End(Not run)
```

```r
## Not run:
# Maximum likelihood estimation near theta upper bounds for a=3 and n=2.
set.seed(832)
UV <- simCOP(300, cop=FGMcop, para=c(+0.16, 3, 2))
# Define a transform function for parameter domain, though mleCOP does
# provide some robustness anyway---not forcing n into the positive
# domain via as.integer(exp(p[3])) seems to not always be needed.
FGMpfunc <- function(p) {
      d <- p[1]; a <- exp(p[2]); n <- as.integer(exp(p[3]))
      lwr <- -min(c(1,1/(n*a^2))); upr <- 1/(n*a)
      d <- ifelse(d <= lwr, lwr, ifelse(d >= upr, upr, d))
      return( c(d, a, n) )
    }
para <- c(0.16, 3, 2); init <- c(0, 1, 1)
ML <- mleCOP(UV$U, UV$V, cop=FGMcop, init.para=init, parafn=FGMpfunc)
print(ML$para) # [1] 0.1596361 3.1321228 2.0000000
# So, we have recovered reasonable estimates of the three parameters
# given through MLE estimation.
densityCOPplot(cop=FGMcop, para= para, contour.col=2)
densityCOPplot(cop=FGMcop, para=ML$para, ploton=FALSE) #
## End(Not run)
```

---

`footCOP`  
The Spearman Footrule of a Copula

Description

Compute the measure of association known as the *Spearman Footrule* $\psi_C$ (Nelsen et al., 2001, p. 281), which is defined as
ψ_C = \frac{3}{2} Q(C, M) - \frac{1}{2},

where C(u, v) is the copula, M(u, v) is the Fréchet–Hoeffding upper bound (M), and Q(a, b) is a concordance function (concordCOP) (Nelsen, 2006, p. 158). The ψ_C in terms of a single integration pass on the copula is

ψ_C = 6 \int_0^1 C(u, u) \, du - 2.

Note that Nelsen et al. (2001) use φ_C but that symbol is taken in copBasic for the Hoeffding Phi (hoefCOP), and Spearman Footrule does not seem to appear in Nelsen (2006).

Usage

footCOP(cop=NULL, para=NULL, by.concordance=FALSE, as.sample=FALSE, ...)

Arguments

- cop A copula function;
- para Vector of parameters or other data structure, if needed, to pass to the copula;
- by.concordance Instead of using the single integral to compute ψ_C, use the concordance function method implemented through concordCOP; and
- as.sample A logical controlling whether an optional R data frame in para is used to compute the ψ (see Note); and
- ... Additional arguments to pass, which are dispatched to the copula function cop and possibly concordCOP, such as brute or delta used by that function.

Value

The value for ψ_C is returned.

Note

Conceptually, the sample Spearman Footrule is a standardized sum of the absolute difference in the ranks (Genest et al., 2010). The sample ψ is

\hat{ψ} = 1 - \frac{\sum_{i=1}^n |R_i - S_i|}{n^2 - 1},

where R_i and S_i are the respective ranks of X and Y and n is sample size. The sampling variance of \hat{ψ} under assumption of independence between X and Y is

\text{var}(\hat{ψ}) = \frac{2n^2 + 7}{5(n + 1)(n - 1)^2}.

Genest et al. (2010) present additional equations for estimation of the distribution \hat{ψ} variance for conditions of dependence based on copulas.
Author(s)

W.H. Asquith

References


See Also

blomCOP, giniCOP, hoefCOP, rhoCOP, tauCOP, wolfCOP

Examples

```r
footCOP(cop=PSP) # 0.3177662
# footCOP(cop=PSP, by.concordance=TRUE) # 0.3178025

## Not run:
n <- 2000; UV <- simCOP(n=n, cop=GHcop, para=2.3, graphics=FALSE)
footCOP(para=UV, as.sample=TRUE) # 0.5594364 (sample version)
footCOP(cop=GHcop, para=2.3) # 0.5513380 (copula integration)
footCOP(cop=GHcop, para=2.3, by.concordance=TRUE) # 0.5513562 (concordance function)
# where the later issued warnings on the integration
## End(Not run)
```

The Fréchet Family Copula

Description

The Fréchet Family copula (Durante, 2007, pp. 256–259) is

\[ C_{\alpha,\beta}(u,v) = FF(u,v) = \alpha M(u,v) + (1 - \alpha - \beta) \Pi(u,v) + \beta W(u,v), \]

where \( \alpha, \beta \geq 0 \) and \( \alpha + \beta < 1 \). The Fréchet Family copulas are convex combinations of the fundamental copulas \( W \) (Fréchet–Hoeffding lower-bound copula; \( W \)), \( \Pi \) (independence; \( P \)), and \( M \) (Fréchet–Hoeffding upper-bound copula; \( M \)). The copula is comprehensive because both \( W \) and \( M \) can be obtained. The parameters are readily estimated using Spearman Rho (\( \rho_C \); rhoCOP) and Kendall Tau (\( \tau_C \); tauCOP) by

\[ \tau_C = \frac{(\alpha - \beta)(\alpha + \beta + 2)}{3} \]

and \( \rho_C = \alpha - \beta \).
The Fréchet Family copula virtually always has a visible singular component unless $\alpha, \beta = 0$. The copula has respective lower- and upper-tail dependency parameters of $\lambda^L = \alpha$ and $\lambda^U = \alpha$ (taildepCOP). Durante (2007, p. 257) reports that the Fréchet Family copula can approximate any bivariate copula in a “unique way” and the error bound can be estimated.

Usage

FRECHETcop(u, v, para=NULL, rho=NULL, tau=NULL, par2rhotau=FALSE, ...)

Arguments

u Nonexceedance probability $u$ in the $X$ direction;
v Nonexceedance probability $v$ in the $Y$ direction;
para A vector (two element) of parameters $\alpha$ and $\beta$;
rho Spearman Rho from which to estimate the parameters;
tau Kendall Tau from which to estimate the parameters;
par2rhotau A logical that if TRUE will return an R list of the $\rho_C$ and $\tau_C$ for the parameters; and
... Additional arguments to pass.

Details

The function will check the consistency of the parameters whether given by argument or computed from $\rho_C$ and $\tau_C$. The term “Family” is used with this particular copula in copBasic so as to draw distinction to the Fréchet lower- and upper-bound copulas as the two limiting copulas are called.

For no other reason than that it can be easily done and makes a nice picture, loop through a nest of $\rho$ and $\tau$ for the Fréchet Family copula and plot the domain of the resulting parameters:

```r
ops <- options(warn=-1) # warning supression because "loops" are dumb
taus <- rhos <- seq(-1,1, by=0.01)
plot(NA, NA, type="n", xlim=c(0,1), ylim=c(0,1),
     xlab="Frechet Copula Parameter Alpha",
     ylab="Frechet Copula Parameter Beta")
for(tau in taus) {
    for(rho in rhos) {
        fcop <- FRECHETcop(rho=rho, tau=tau)
        if(! is.na(fcop$para[1])) points(fcop$para[1], fcop$para[2])
    }
}
options(ops)
```

Value

Value(s) for the copula are returned using the $\alpha$ and $\beta$ as set by argument para; however, if para=NULL and rho and tau are set and compatible with the copula, then $\{\rho_C, \tau_C\} \rightarrow \{\alpha, \beta\}$, parameter estimation made, and an R list is returned.
FRECHETcop

Note

A convex combination (\texttt{convex2COP}) of \(\Pi\) and \(M\), which is a modification of the Fréchet Family, is the \textit{Linear Spearman} copula:

\[
C_\alpha(u,v) = (1 - \alpha)\Pi(u,v) + \alpha M(u,v),
\]

for \(0 \leq \alpha \leq 1\), and the parameter is equal to \(\rho_C\). When the convex combination is used for construction, the complement of the parameter is equal to \(\rho_C\) (e.g. \(1 - \alpha = \rho_C\); \texttt{rhoCOP}), which can be validated by

\[
\texttt{rhoCOP(cop=convex2COP, para=list(alpha=1-0.48, cop1=P, cop2=M))} \# 0.4799948
\]

Author(s)

W.H. Asquith

References


See Also

\(M, P, W\)

Examples

```r
## Not run:
ppara <- c(0.25, 0.50)
fcop <- FRECHETcop(para=ppara, par2rhotau=TRUE)
RHO <- fcop$rho; TAU <- fcop$tau
level.curvesCOP(cop=FRECHETcop, para=ppara) \# Durante (2007, Fig. C.27(b))

UV <- simCOP(n=50, cop=FRECHETcop, para=ppara, ploton=FALSE, points=FALSE)
tau <- cor(UV$U, UV$V, method="kendall") \# sample Kendall Tau
rho <- cor(UV$U, UV$V, method="spearman") \# sample Spearman Rho
spara <- FRECHETcop(rho=rho, tau=tau) \# a fitted Frechet Family copula
spara <- spara$para
if(is.na(spara[1])) { \# now a fittable combination is not guaranteed
  warning("sample rho and tau do not provide valid parameters, ",
           "try another simulation")
} else { \# now if fit, draw some red-colored level curves for comparison
  level.curvesCOP(cop=FRECHETcop, para=spara, ploton=FALSE, col=2)
}
## End(Not run)
```
Description

The \textit{g-EV copula} (Joe, 2014, p. 105) is a limiting form of the \textit{Gaussian copula}:

\[ C_\rho(u, v) = \text{gEV}(u, v; \rho) = \exp(-A(x, y; \rho)), \]

where \( x = -\log(u) \), \( y = -\log(v) \), and

\[ A(x, y; \rho) = y, \]

for \( 0 \leq x/(x+y) \leq \rho^2/(1+\rho^2) \),

\[ A(x, y; \rho) = (x + y - 2\rho\sqrt{xy})/(1-\rho^2), \]

for \( \rho^2/(1+\rho^2) \leq x/(x+y) \leq 1/(1+\rho^2) \),

\[ A(x, y; \rho) = x, \]

for \( 1/(1+\rho^2) \leq x/(x+y) \leq 1 \) and where \( \rho \in [0, 1] \). A somewhat curious observation is that this copula has relative hard boundaries into the upper-left and lower-right corners when compared to the other copulas supported by the \texttt{copBasic} package. In other words, the hull defined by the copula has a near hard (not fuzzy) curvilinear boundaries that adjust with the parameter \( \rho \).

Usage

\texttt{gEVcop(u, v, para=NULL, ...)}

Arguments

\begin{itemize}
\item \texttt{u} Nonexceedance probability \( u \) in the \textit{X} direction;
\item \texttt{v} Nonexceedance probability \( v \) in the \textit{Y} direction;
\item \texttt{para} The parameter \( \rho \); and
\item \texttt{...} Additional arguments to pass.
\end{itemize}

Value

Value(s) for the copula are returned.

Author(s)

W.H. Asquith

References

### Examples

```r
## Not run:
UV <- simCOP(200, cop=gEVcop, para=0.8) #
## End(Not run)

## Not run:
# Joe (2014, p. 105) has brief detail indicating rho = [0,1] and though it seems
# Rho would be a Pearson correlation, this does not seem to be the case. The Rho
# seems to start with that of the Gaussian and then through the extreme-value
# transform, it just assumes the role of a parameter called Rho.
rho <- 0.8
UV <- simCOP(2000, cop=gEVcop, para=rho)
P <- cor(UV[,1], UV[,2], method="pearson")
if(abs(P - rho) < 0.001) {
  print("Yes same")
} else { print("nope not") } #
## End(Not run)

## Not run:
gEVparameter <- seq(0.02, 1, by=0.02)
SpearmanRho <- sapply(gEVparameter, function(k) rhoCOP(cop=gEVcop, para=k))
plot(gEVparameter, SpearmanRho)
for(rho in gEVparameter) UV <- simCOP(n=1000, cop=gEVcop, para=rho) #
## End(Not run)
```

---

**GHcop**

**The Gumbel–Hougaard Extreme Value Copula**

### Description

**SYMMETRIC GUMBEL-HOUGAARD**—The Gumbel–Hougaard copula (Nelsen, 2006, pp. 118 and 164) is

\[
C_\Theta(u, v) = \text{GH}(u, v) = \exp\left\{-\left[-\log u\right]^\Theta + \left[-\log v\right]^\Theta\right\}^{1/\Theta},
\]

where \( \Theta \in [1, \infty) \). The copula here is a bivariate extreme value copula (BEV). The parameter \( \Theta \) is readily estimated using a Kendall Tau (say a sample version \( \hat\tau \)) where the \( \tau \) of the copula (\( \tau_C \)) is defined as

\[
\tau_C = \frac{\Theta - 1}{\Theta} \rightarrow \Theta = \frac{1}{1 - \tau}.
\]

The copula is readily extended into \( d \) dimensions by

\[
C_\Theta(u, v) = \exp\left\{-\left[-\log u_1\right]^\Theta + \cdots + \left[-\log u_d\right]^\Theta\right\}^{1/\Theta}.
\]

However, such an implementation is not available in the **copBasic** package.
Every Gumbel–Hougaard copula is a multivariate extreme value (MEV) copula, and hence useful in analysis of extreme value distributions. The Gumbel–Hougaard copula is the only Archimedean MEV (Salvadori et al., 2007, p. 192). The Gumbel–Hougaard copula has respective lower- and upper-tail dependency parameters of $\lambda^L = 0$ and $\lambda^U = 2 - 2^{1/\theta}$, respectively. Nelsen (2006, p. 96) shows that $C^U_u(u^{1/r}, v^{1/r}) = C_d(u, v)$ so that every Gumbel–Hougaard copula has a property known as max-stable. A dependence measure uniquely defined for BEV copulas is shown under rhobevCOP.

A comparison through simulation between Gumbel–Hougaard implementations by the R packages acopula, copBasic, copula, and Gumbel is shown in the Examples section. At least three divergent techniques for random variate generation are used amongst those packages. The simulations also use copBasic-style random variate generation (conditional simulation) using an analytical-numerical hybrid solution to conditional inverse described in the Note section.

TWO-PARAMETER GUMBEL–HOUGAARD—A permutation symmetric (isCOP.permsym) but almost certainly radial asymmetric (isCOP.radsym) version of the copula is readily constructed (Brahimi et al., 2015) into a two-parameter version:

$$C(u, v; \beta_1, \beta_2) = \left[\left((u^{-\beta_2} - 1) + (v^{-\beta_2} - 1)^{\beta_1}\right)^{1/\beta_1} + 1\right]^{-1/\beta_2},$$

where $\beta_1 \geq 1$ and $\beta_2 > 0$. Both parameters controls the general level of association, whereas parameter $\beta_2$ can be thought of as controlling left-tail dependency (taildepCOP, $\lambda^{U/L}_{\beta_1, \beta_2}$; e.g. $\lambda^U_{1.5; 0.2} = 0.413$ for all $\beta_2$ but $\lambda^L_{1.5; 0.2} = 0.811$ and $\lambda^L_{1.5; 2.2} = 0.099$. Brahimi et al. (2015) report a Spearman Rho (rhoCOP) for a GHB(1.5, 0.2)(u, v) is 0.5, which is readily confirmed in copBasic by the function call rhoCOP(cop=GHCop, para=c(1.5, 0.2)). The two-parameter GHB is triggered if the length of the para argument is exactly 2.

ASYMMETRIC GUMBEL–HOUGAARD—An asymmetric version of the copula is readily constructed (Joe, 2014, p. 185–186) into a three-parameter version with Marshall–Olkin copulas on the boundaries:

$$C(u, v; \Theta, \pi_2, \pi_3) = \exp[-A(-\log u, -\log v; \Theta, \pi_2, \pi_3)],$$

where $\Theta \geq 1$ as before, $0 \leq \pi_2, \pi_3 \leq 1$, and

$$A(x, y; \Theta, \pi_2, \pi_3) = \left[\left((\pi_2 x)^{\Theta} + (\pi_3 y)^{\Theta}\right)^{1/\Theta} + (1 - \pi_2)x + (1 - \pi_3)y\right].$$

The asymmetric GHB is triggered if the length of the para argument is exactly 3. The GHCop function provides no mechanism for estimation of the parameters for the asymmetric version. Reviewing simulations, the bounds on the $\pi$ parameters in Joe (2014, p. 185) “[0 $\leq \pi_2 < \pi_3 \leq 1]” might be incorrect—by Joe back referencing to Joe (2014, eq. 4.35, p. 183) the $\pi$-limits as stated for copBasic are shown. An algorithm for parameter estimation for the asymmetric GHB using two different measures of bivariate skewness as well as an arbitrary measure of association is shown in section Details in joeskewCOP.

Usage

GHCop(u, v, para=NULL, tau=NULL, tau.big=0.985, cor=NULL, ...)

GHcop
Arguments

- **u** Nonexceedance probability $u$ in the $X$ direction;
- **v** Nonexceedance probability $v$ in the $Y$ direction;
- **para** A vector (single element or triplet) of parameters—the $\Theta$ parameter of the copula;
- **tau** Kendall Tau $\tau$ from which to estimate the parameter $\Theta$;
- **tau.big** The largest value for $\tau_C$ prior to switching to the $M$ copula applicable to the symmetric version of this copula;
- **cor** A **copBasic** syntax for “the correlation coefficient” suitable for the copula—a synonym for tau; and
- Additional arguments to pass.

Details

Numerical experiments seem to indicate for $\tau_C > 0.985$ that failures in the numerical partial derivatives in **derCOP** and **derCOP2** result—a $\tau_C$ this large is indeed large. As $\Theta \rightarrow \infty$ the Gumbel–Hougaard copula becomes the Fréchet–Hoeffding upper-bound copula $M$ (see §). A $\tau_C \approx 0.985$ yields $\Theta \approx 66 + 2/3$, then for $\Theta > 1/(1 - \tau_C)$ flips over to the $M$ copula with a warning issued.

Value

Value(s) for the copula are returned using the $\Theta$ as set by argument **para**. Alternative returned values are possible: (1) If **para=NULL** and **tau** is set, then $\tau_C \rightarrow \Theta$ and an **R** list is returned. (2) If **para=NULL** and **tau=NULL**, then an attempt to estimate $\Theta$ from the $u$ and $v$ is made by $\text{cor}(u, v) \rightarrow \tau_C \rightarrow \Theta$ by either trigger using $\text{cor}(u, v, \text{method}="kendall")$ in **R**, and an **R** list is returned. The possibly returned list has the following elements:

- **para** The computed $\Theta$ from the given bivariate data in **para**; and
- **tau** The sample estimate of $\tau$.

Note

**SYMMETRIC GUMBEL–HOUGAARD**—A function for the derivative of the copula (Joe, 2014, p. 172) given $u$ is

```
"GHcop.derCOP" <- function(u, v, para=NULL, ...) {
  x <- -log(u); y <- -log(v)
  A <- exp(-(x^para + y^para)^(1/para)) * (1 + (y/x)^para)^(1/para - 1)
  return(A/u)
}
```

that can be tested by the following

```
Theta <- 1/(1-.15) # a Kendall Tau of 0.15
GHcop.derCOP( 0.5, 0.75, para=Theta) # 0.7787597
derCOP(cop=GHcop, 0.5, 0.75, para=Theta) # 0.7787597
# The next two nearly return same value but conversion to GRVs
```
where the last numerical approximation shows that tighter tolerance is needed. A function for the inverse of the derivative (Joe, 2014, p. 172) given $u$ by an analytical-numerical hybrid is

```r
"GHcop.derCOPinv" <- function(u,t, para=NULL, verbose=FALSE, 
  toI=.Machine$double.eps, ...) {
  if(length(u) > 1) warning("only the first value of u will be used")
  if(length(t) > 1) warning("only the first value of t will be used")
  if(is.null(para)) { warning("para can not be NULL"); return(NA) }
  u <- u[1]; t <- t[1]; rt <- NULL
  x <- -log(u); A <- (x + (para - 1)*log(x) - log(t))
  hz <- function(z) { z + (para - 1)*log(z) - A }
  zmax <- x; i <- 0; hofz.lo <- hz(zmax)
  if(sign(hofz.lo) != -1) warning("sign for h(z) is not negative!")
  while(1) {
    i <- i + 1
    if(i > 100) {
      warning("maximum iterations looking for zmax reached"); break
    } # increment zmax by 1/2 log cycle, sign(hofz.lo) should be negative!
    if(sign(hz(zmax <- zmax + 1/2)) != sign(hofz.lo)) break
  }
  try(rt <- uniroot(hz, c(x, zmax), tol=toI, ...), silent=FALSE)
  if(verbose) print(rt)
  if(is.null(rt)) {
    warning("NULL on the inversion of the GH copula derivative")
    return(NA)
  }
  zo <- rt$root
  y <- (zo^para - x^para)^(1/para)
  names(y) <- NULL
  return(exp(-y))
}
```

that can be tested by the following, which also shows how to increase the tolerance on the numerical implementation

```r
u <- 0.999; p <- 0.999
GHcop.derCOPinv( u, p, para=1.56) # 0.999977
derCOPinv(cop=GHcop, u, p, para=1.56) # 1 (unity), needs tighter tolerance
derCOPinv(cop=GHcop, u, p, para=1.56, tol=.Machine$double.eps/10) # 0.999977
```

**ASYMMETRIC GUMBEL–HOUGAARD**—Set $\tau_C = 0.35$ then for a symmetric and then reflection
on the 1:1 line of the asymmetric Gumbel–Hougaard copula and compute the primary parameter $\Theta$, and lastly, compute three bivariate $\nu_C$ skewnesses (\texttt{nuskewCOP}):

\begin{verbatim}
Theta1 <- uniroot(function(t) {
  0.35 - tauCOP(cop=GHcop, para=c(t))
}, c(1,10))$root
Theta2 <- uniroot(function(t) {
  # asymmetric
  0.35 - tauCOP(cop=GHcop, para=c(t, 0.6, 0.9))
}, c(1,30))$root
Theta3 <- uniroot(function(t) {
  # asymmetric reflection on 1:1
  0.35 - tauCOP(cop=GHcop, para=c(t, 0.9, 0.6))
}, c(1,30))$root
# Theta1 = 1.538462 and Theta2 = Theta3 = 2.132856
# Three "skews" based on a combination of U, V, and C(u,v) [nuskew()]
nuskewCOP(cop=GHcop, 1.538462) # zero bivariate skewness
nuskewCOP(cop=GHcop, c(2.132856, 0.6, 0.9)) # 0.008245653
nuskewCOP(cop=GHcop, c(2.132856, 0.9, 0.6)) # -0.008245653
So, we see, holding $\tau_C$ constant, that the GH has a $\nu_{GH}(1.538) = 0$ but the asymmetric case $\nu_{GH}(2.133,0.6,0.9) = 0.0082$ and $\nu_{GH}(2.133,0.9,0.6) = -0.0082$ where the change in sign represents reflection about the 1:1 line. Finally, compute \textit{L-coskew} by large simulation and the adjective “bow” representing the direction of bowing or curvature of the principle copula density.

# Because the Tau's are all similar, there is nothing to learn from the # L-correlation, let us inspect the L-coskew instead:
coT3.1<-lmomco::lcomoms2(simCOP(n=8000, cop=GHcop, para=c(Theta1 )))$T3
coT3.2<-lmomco::lcomoms2(simCOP(n=8000, cop=GHcop, para=c(Theta2,.6,.9)))$T3
coT3.3<-lmomco::lcomoms2(simCOP(n=8000, cop=GHcop, para=c(Theta3,.9,.6)))$T3
# The simulations for Theta1 have no curvature about the diagonal.
# The simulations for Theta2 have curvature towards the upper left.
# The simulations for Theta3 have curvature towards the lower right.
message("# L-coskews: ",round(coT3.1[1,2],digits=4),"(symmetric) ",
  round(coT3.2[1,2],digits=4),"(symm.--bow UL) ",
  round(coT3.3[1,2],digits=4),"(symm.--bow UL)")
message("# L-coskews: ",round(coT3.1[2,1],digits=4),"(symmetric) ",
  round(coT3.2[2,1],digits=4),"(symm.--bow LR) ",
  round(coT3.3[2,1],digits=4),"(symm.--bow LR)")
# L-coskews: 0.0533(symmetric) 0.1055(asy.-bow UL) 0.0021(asy.-bow LR)
# L-coskews: 0.0679(symmetric) 0.0112(asy.-bow LR) 0.1154(asy.-bow LR)

Thus, the \textit{L-comoments} (Asquith, 2011) using their sample values measure something fundamental about the bivariate association between the three copulas chosen. The L-coskews for the symmetrical case are about equal and are

$$\tau_{3[12]}^{GH(\Theta_1)} \approx \tau_{3[21]}^{GH(\Theta_1)} \rightarrow (0.0533 + 0.0679)/2 = 0.0606,$$

whereas the L-coskew for the curvature to the upper left are

$$\tau_{3[1]}^{GH(\Theta_2,0.6,0.9)} = 0.1055$$

and $\tau_{3[12]}^{GH(\Theta_2,0.6,0.9)} = 0.0112$, whereas the L-coskew for the curvature to the lower right are

$$\tau_{3[12]}^{GH(\Theta_3,0.9,0.6)} = 0.0021$$

and $\tau_{3[12]}^{GH(\Theta_3,0.9,0.6)} = 0.1154$.

Thus, the $\pi_2$ and $\pi_3$ parameters as chosen add about $(0.1154 + 0.0112)/2 - 0.0606 \rightarrow 0.1105 - 0.0606 = 0.05$ L-coskew units to the bivariate distribution.
Author(s)

W.H. Asquith

References


See Also

M, GLcop, HRcop, tEVcop, rhobevCOP

Examples

Theta <- 2.2 # Let us see if numerical and analytical tail deps are the same.
del.lamU <- abs( taildepCOP(cop=GHcop, para=Theta)$lambdaU - (2-2^(1/Theta)) )
as.logical(del.lamU < 1E-6) # TRUE
## Not run:
# The simulations match Joe (2014, p. 72) for Gumbel-Hougaard
n <- 600; nsim <- 1000; set.seed(946) # see for reproducibility
SM <- sapply(1:nsim, function(i) { rs <- semicorCOP(cop=GHcop, para=1.35, n=n)
c(rs$botleft.semicor, rs$topright.semicor) })
RhoM <- round(mean(SM[,1]), digits=3)
RhoP <- round(mean(SM[,2]), digits=3)
SE.RhoM <- round(sd(SM[,1]), digits=3)
SE.RhoP <- round(sd(SM[,2]), digits=3)
SE.RhoMP <- round(sd(SM[,2] - SM[,1]), digits=3)
# Semi-correlations (sRho) and standard errors (SEs)
message("# sRho[-]=", RhoM, " (SE[-]=", SE.RhoM, ") Joe(p.72)=0.132 (SE[-]=0.08)"
message("# sRho[+]=", RhoP, " (SE[+]=", SE.RhoP, ") Joe(p.72)=0.415 (SE[+]=0.07)"
message("# SE(sRho[-] - sRho[+])="", SE.RhoMP, " Joe(p.72) SE=0.10"
# sRho[-]=0.134 (SE[-]=0.076) Joe(p.72)=0.132 (SE[-]=0.08)
# sRho[+]=0.407 (SE[+]=0.074) Joe(p.72)=0.415 (SE[+]=0.07)
# SE(sRho[-] - sRho[+])=0.107 Joe(p.72) SE=0.10
## Not run:

## Not run:
file <- "Lcomom_study_of_GHcopPLACKETTcop.txt"
x <- data.frame(tau=NA, trho=NA, srho=NA, PLtheta=NA, PLT2=NA, PLT3=NA, PLT4=NA,
\begin{verbatim}

n <- 250  # Make a large number for very long CPU run but seems stable

for(tau in seq(0,0.98, by=.005)) {
  thetag <- GHcop(u=NULL, v=NULL, tau=tau)$para
  rho <- rhoCOP(cop=GHcop, para=thetag)
  GH <- simCOP(n=n, cop=GHcop, para=thetag, points=FALSE, ploton=FALSE)
  srho <- cor(GH$U, GH$V, method="spearman")
  thetap <- PLACKETTpar(rho=rho)
  PL <- simCOP(n=n, cop=PLACKETTcop, para=thetap, points=FALSE, ploton=FALSE)
  x <- data.frame(tau=tau, rho=rho, srho=srho,
                   GHtheta=thetag, PLtheta=thetap,
                   GHT2=mean(c(GHl$T2[1,2], GHl$T2[2,1])),
                   GHT3=mean(c(GHl$T3[1,2], GHl$T3[2,1])),
                   GHT4=mean(c(GHl$T4[1,2], GHl$T4[2,1])),
                   PLT2=mean(c(PLl$T2[1,2], PLl$T2[2,1])),
                   PLT3=mean(c(PLl$T3[1,2], PLl$T3[2,1])),
                   PLT4=mean(c(PLl$T4[1,2], PLl$T4[2,1])))

  write.table(x, file=file, row.names=FALSE, col.names=FALSE, append=TRUE)
}

D <- read.table(file, header=TRUE); D <- D[complete.cases(D),]

plot(D$tau, D$GHT3, ylim=c(-0.08,0.08), type="n",
     xlab="KENDALL TAU", ylab="L-COSKEW OR NEGATED L-COKURTOSIS")
points(D$tau, D$GHT3, col=2); points(D$tau, D$PLT3, col=1)
points(D$tau, -D$GHT4, col=4, pch=2); points(D$tau, -D$PLT4, col=1, pch=2)

LM3 <- lm(D$GHT3~I(D$tau^1)+I(D$tau^2)+I(D$tau^4)-1)
LM4 <- lm(D$GHT4~I(D$tau^1)+I(D$tau^2)+I(D$tau^4)-1)
LM3c <- LM3$coe; LM4c <- LM4$coe

# Let us compare the conditional simulation method of copBasic by numerics and by the
# above analytical solution for the Gumbel-Hougaard copula to two methods implemented
# by package gumbel, a presumed Archimedean technique by package acopula, and an
# Archimedean technique by package copula. Setting seeds by each "method" below does
# not appear diagnostic because of the differences in which the simulations are made.
nsim <- 10000; kn <- "kendall"  # The theoretical KENDALL TAU is (1.5-1)/1.5 = 1/3

A <- cor(copBasic::simCOP(nsim, cop=GHcop, para=1.5, graphics=FALSE), method=kn)[1,2]
U <- runif(nsim)  # GHcop.derCOPinv() comes from earlier in this documentation.
V <- sapply(1:nsim, function(i) { GHcop.derCOPinv(U[i], runif(1), para=1.5) })

B <- cor(U, V, method=kn)  # Simulate by conditional simulation using exact analytical solution

C <- cor(gumbel::rgumbel(nsim, 1.5, dim=2, method=1), method=kn)[1,2]

# Simulate by conditional simulation using exact analytical solution

D <- cor(gumbel::rgumbel(nsim, 1.5, dim=2, method=2), method=kn)[1,2]

\end{verbatim}
# Simulate by an Archimedean implementation (presumably)
E <- cor(acopula::rCopula(nsim, pars=1.5), method=kn)[1,2]
# Simulate by an Archimedean implementation
G <- cor(copula::rCopula(nsim, copula::gumbelCopula(1.5)), method=kn)[1,2]
K <- round(c(A, B, C, D, E, G), digits=5); rm(A, B, C, D, E, G, kn); tx <- ", "
message("Kendall Tau: ", K[1], tx, K[2], tx, K[3], tx, K[4], tx, K[5], tx, K[6])
# Kendall Tau: 0.32909, 0.32474, 0.33060, 0.32805, 0.32874, 0.33986 -- run 1
# Kendall Tau: 0.33357, 0.32748, 0.33563, 0.32913, 0.32732, 0.32416 -- run 2
# Kendall Tau: 0.34311, 0.33415, 0.33815, 0.33224, 0.32961, 0.33008 -- run 3
# Kendall Tau: 0.32830, 0.33573, 0.32756, 0.33401, 0.33567, 0.33182 -- nsim=50000!
# All solutions are near 1/3 and it is unknown without further study which of the
# six methods would result in the least bias and (or) sampling variability.
## End(Not run)

## giniCOP

The Gini Gamma of a Copula

### Description

Compute the measure of association known as the *Gini Gamma* $\gamma_C$ (Nelsen, 2006, pp. 180–182), which is defined as

$$\gamma_C = Q(C, M) + Q(C, W),$$

where $C(u, v)$ is the copula, $M(u, v)$ is the $M$ function, and $W(u, v)$ is the $W$ function. The function $Q(a, b)$ (concordCOP) is a concordance function (Nelsen, 2006, p. 158). Nelsen also reports that “Gini Gamma measures a concordance relation of “distance” between $C(u, v)$ and monotone dependence, as represented by the Fréchet–Hoeffding lower bound and Fréchet–Hoeffding upper bound copulas $[M(u, v), M$ and $W(u, v), W$ respectively].”

A simpler method of computation and the default for giniCOP is to compute $\gamma_C$ by

$$\gamma_C = 4 \left[ \int_I C(u, u) \, du + \int_I C(u, 1-u) \, du \right] - 2,$$

or in terms of the primary diagonal $\delta(t)$ and secondary diagonal $\delta^*(t)$ (see diagCOP) by

$$\gamma_C = 4 \left[ \int_I \delta(t) \, dt + \int_I \delta^*(t) \, dt \right] - 2.$$

The simpler method is more readily implemented because single integration is fast. Lastly, Nelsen et al. (2001, p. 281) show that $\gamma_C$ also is computable by

$$\gamma_C = 2 \, Q(C, A),$$

where $A$ is a convex combination (convex2COP, using $\alpha = 1/2$) of the copulas $M$ and $W$ or $A = (M + W)/2$. However, integral convergence errors seem to trigger occasionally, and the first definition by summation $Q(C, M) + Q(C, W)$ thus is used. The convex combination is demonstrated in the Examples section.
Usage

\[
\text{giniCOP}(\text{cop=NULL, para=NULL, by.concordance=FALSE, as.sample=FALSE, ...})
\]

Arguments

- **cop**: A copula function;
- **para**: Vector of parameters or other data structure, if needed, to pass to the copula;
- **by.concordance**: Instead of using the single integrals (Nelsen, 2006, pp. 181–182) to compute \( \hat{\gamma}_C \), use the concordance function method implemented through `concordCOP`;
- **as.sample**: A logical controlling whether an optional `R data.frame` in `para` is used to compute the \( \hat{\gamma}_C \) (see Note); and
- **...**: Additional arguments to pass, which are dispatched to the copula function `cop` and possibly `concordCOP` if `by.concordance=TRUE`, such as `delta` used by that function.

Value

The value for \( \hat{\gamma}_C \) is returned.

Note

Conceptually, the sample Gini Gamma (\( \hat{\gamma} \); Genest et al., 2010) is

\[
\hat{\gamma} = \frac{2}{n^2} \sum_{i=1}^{n} \left| \left( \frac{n+1}{n} - R_i \right) - S_i \right| - \left| R_i - S_i \right| \frac{1}{n^2 - 1},
\]

where \( R_i \) and \( S_i \) are the respective ranks of \( X \) and \( Y \) and \( n \) is sample size. The sampling variance of \( \hat{\gamma} \) under assumption of independence between \( X \) and \( Y \) is

\[
\text{var}(\hat{\gamma})_{n \text{ even}} = \frac{2}{3} \frac{(n^2 + 2)}{(n - 1)n^2} \quad \text{and}
\]

\[
\text{var}(\hat{\gamma})_{n \text{ odd}} = \frac{2}{3} \frac{(n^2 + 3)}{(n - 1)(n^2 - 1)}.
\]

Genest et al. (2010) present additional equations for estimation of the distribution \( \hat{\gamma} \) variance for conditions of dependence based on copulas.

Author(s)

W.H. Asquith

References


See Also

blomCOP, footCOP, hoefCOP, rhoCOP, tauCOP, wolfCOP, joeskewCOP, uvlmoms

Examples

giniCOP(cop=PSP) # = 0.3819757
## Not run:
giniCOP( cop=PSP, by.concordance=TRUE) # Q(C,M) + Q(C,W) = 0.3820045
# use convex combination ---triggers integration warning but returns anyway
cxpara <- list(alpha=1/2, cop1=M, cop2=W) # parameters for convex2COP()
2*tauCOP(cop=PSP, cop2=convex2COP, para2=cxpara) # 2*Q(C,A) = 0.3819807
# where the later issued warnings on the integration
## End(Not run)

## Not run:
n <- 2000; UV <- simCOP(n=n, cop=N4212cop, para=9.3, graphics=FALSE)
giniCOP(para=UV, as.sample=TRUE) # 0.9475900 (sample version)
giniCOP(cop=N4212cop, para=9.3) # 0.9479528 (copula integration)
giniCOP(cop=N4212cop, para=9.3, by.concordance=TRUE) # 0.9480267 (concordance function)
# where the later issued warnings on the integration
## End(Not run)

## Not run:
# The canonical example of theoretical and sample estimators of bivariate
# association for the package: Blomqvist Beta, Spearman Footrule, Gini Gamma,
# Hoeffding Phi, Kendall Tau, Spearman Rho, and Schweizer-Wolff Sigma
# and comparison to L-correlation via lmomco::lcomoms2().
n <- 9000; set.seed(56)
para <- list(cop1=PLACKETTcop, cop2=PLACKETTcop, para1=1.45, para2=21.9,
alpha=0.41, beta=0.08)
D <- simCOP(n=n, cop=composite2COP, para=para, cex=0.5, col=rgb(0,0,0,0.2), pch=16)
blomCOP(cop=composite2COP, para=para) # 0.4037908 (theoretical)
blomCOP(para=D, as.sample=TRUE) # 0.4008899 (sample)
footCOP(cop=composite2COP, para=para) # 0.3721555 (theoretical)
footCOP(para=D, as.sample=TRUE) # 0.3703623 (sample)
giniCOP(cop=composite2COP, para=para) # 0.4334687 (theoretical)
giniCOP(para=D, as.sample=TRUE) # 0.4311698 (sample)
tauCOP(cop=composite2COP, para=para) # 0.3806909 (theoretical)
tauCOP(para=D, as.sample=TRUE) # 0.3788139 (sample)
rhoCOP(cop=composite2COP, para=para) # 0.5257662 (theoretical)
rhoCOP(para=D, as.sample=TRUE) # 0.5242380 (sample)

lmomco::lcomoms2(D)$T2 # 1
# 0.5245154 1
hoefCOP(cop=composite2COP, para=para) # 0.5082776 (theoretical)
subsample <- D[sample(1:n, n/5),] # subsampling for speed
hoefCOP(para=subsample, as.sample=TRUE) # 0.5033842 (re-sample)
#hoefCOP(para=D, as.sample=TRUE) # major CPU hog, n too big
# because the Ds are already "probabilities" just resample as shown above
wolfCOP(cop=composite2COP, para=para) # 0.5257662 (theoretical)
#wolfCOP(para=D, as.sample=TRUE) # major CPU hog, n too big
wolfCOP(para=subsample, as.sample=TRUE) # 0.5338009 (re-sample)
## End(Not run)
The Galambos Extreme Value Copula (with Gamma Power Mixture [Joe/BB4] and Lower Extreme Value Limit)

**Description**

The Galambos copula (Joe, 2014, p. 174) is

\[ C_\Theta(u, v) = GL(u, v) = uv \exp\left[\left(x^{-\Theta} + y^{-\Theta}\right)^{-1/\Theta}\right], \]

where \( \Theta \in [0, \infty) \), \( x = -\log u \), and \( y = -\log v \). As \( \Theta \to 0^+ \), the copula limits to independence \( (\Pi; P) \) and as \( \Theta \to \infty \), the copula limits to perfect association \( (M; M) \). The copula here is a bivariate extreme value copula \( (BEV) \), and parameter estimation for \( \Theta \) requires numerical methods.

There are two other genetically related forms. Joe (2014, p. 197) describes an extension of the Galambos copula as a Galambos gamma power mixture \( (GLPM) \), which is Joe’s BB4 copula, with the following form

\[ C_{\Theta,\delta}(u, v) = GLPM(u, v) = \left( x + y - 1 - \left[(x - 1)^{-\delta} + (y - 1)^{-\delta}\right]^{-1/\delta}\right)^{-1/\Theta}, \]

where \( x = u^{-\Theta}, y = v^{-\Theta} \), and \( \Theta \geq 0, \delta \geq 0 \). (Joe shows \( \delta > 0 \), but zero itself seems to work without numerical problems in practical application.) As \( \delta \to 0^+ \), the “MTJC family” (Mardia–Takahasi–Cook–Johnson) results (implemented internally with \( \Theta \) as the incoming parameter). As \( \Theta \to 0^+ \) the Galambos above results with \( \delta \) as the incoming parameter.

This second copula in turn has a lower extreme value limit form that leads to a min-stable bivariate exponential having Pickand dependence function of

\[ A(x, y; \Theta, \delta) = x + y - \left[x^{-\Theta} + y^{-\Theta} - (x^{\Theta\delta} + y^{\Theta\delta})^{-1/\delta}\right]^{-1/\Theta}, \]

where this third copula is

\[ C_{\Theta,\delta}^{LEV}(u, v) = GLEV(u, v) = \exp[-A(-\log u, -\log v; \Theta, \delta)], \]

for \( \Theta \geq 0, \delta \geq 0 \) and is known as the two-parameter Galambos. (Joe shows \( \delta > 0 \), but \( \delta = 0 \) itself seems to work without numerical problems in practical application.)

**Usage**

```
GLcop( u, v, para=NULL, ...)  
GLEVcop( u, v, para=NULL, ...)  
GLPMcop( u, v, para=NULL, ...) # inserts third parameter automatically  
JOcopBB4( u, v, para=NULL, ...) # inserts third parameter automatically
```

**Arguments**

- **u**  Nonexceedance probability \( u \) in the \( X \) direction;
- **v**  Nonexceedance probability \( v \) in the \( Y \) direction;
To trigger $GL(u,v)$, a vector (single element) of $\Theta$, to trigger $GLEV(u,v)$, a two element vector of $\Theta$ and $\delta$ and alias is GLEVcop, and to trigger $GLPM(u,v)$, a three element vector of $\Theta, \delta$, and any number (the presence of the third entry alone is the triggering mechanism) though aliases GLPM or J0copBB4 will insert the third parameter automatically for convenience; and

Additional arguments to pass.

**Value**

Value(s) for the copula are returned.

**Note**

Joe (2014, p. 198) shows $GLEV(u,v; \Theta, \delta)$ as a two-parameter Galambos, but its use within the text seemingly is not otherwise obvious. However, testing of the implementation here seems to show that this copula is really not broader in form than $GL(u,v; \alpha)$. The $\alpha$ can always(?) be chosen to mimic the $\{\Theta, \delta\}$. This assertion can be tested from a semi-independent direction. First, define an alternative style of one-parameter Galambos:

```r
GL1cop <- function(u,v, para=NULL, ...) {
  GL1pA <- function(x,y,t) { # Pickend dependence func form 1p Galambos
    x + y - (x^-t + y^-t)^(-1/t)
  }
  if(length(u) == 1) { u <- rep(u, length(v)) } else
  if(length(v) == 1) { v <- rep(v, length(u)) }
  exp(-GL1pA(-log(u), -log(v), para[1]))
}
```

Second, redefine the two-parameter Galambos:

```r
GL2cop <- function(u,v, para=NULL, ...) {
  GL2pA <- function(x,y,t,d) { # Pickend dependence func form 2p Galambos
    x + y - (x^-t + y^-t - (x^(t*d) + y^(t*d))^(-1/d))^(-1/t)
  }
  if(length(u) == 1) { u <- rep(u, length(v)) } else
  if(length(v) == 1) { v <- rep(v, length(u)) }
  exp(-GL2pA(-log(u), -log(v), para[1], para[2]))
}
```

Next, we can combine the Pickend dependence functions into an objective function. This objective function will permit the computation of the $\alpha$ given a pair $\{\Theta, \delta\}$.

```r
objfunc <- function(a,t=NA,d=NA, x=0.7, y=0.7) {
  lhs <- (x^-t + y^-t - (x^(t*d) + y^(t*d))^(-1/d))^(-1/t)
  rhs <- (x^-a + y^-a)^(-1/a); return(rhs - lhs) # to be uniroot'ed
}
```

A demonstration can now be made:
t <- 0.6; d <- 4; lohi <- c(0,100)
set.seed(3); UV <- simCOP(3000, cop=GL2cop, para=c(t,d), pch=16,col=3,cex=0.5)
a <- uniroot(objfunc, interval=lohi, t=t, d=d)$root
set.seed(3); UV <- simCOP(3000, cop=GL1cop, para=a, lwd=0.5, ploton=FALSE)

The graphic so produced shows almost perfect overlap in the simulated values. To date, the author has not really found that the two parameters can be chosen such that the one-parameter version can not attain. Extensive numerical experiments using simulated parameter combinations through the use of various copula metrics (tail dependencies, L-comoments, etc) have not found material differences. Has the author of this package missed something?

Author(s)
W.H. Asquith

References

See Also
M, P, GHcop, HRcop, tEVcop

Examples
# Theta = pi for GLcop and recovery through Blomqvist Beta (Joe, 2014, p. 175)
log(2)/(log(log(2)/log(1+blomCOP(cop=GLcop, para=pi))))

# Theta = 2 and delta = 3 for the GLPM form and Blomqvist Beta (Joe, 2014, p. 197)
t <- 2; Btheo <- blomCOP(GLPMcop, para=c(t,3))
Bform <- (2^(t+1) - 1 - taildepCOP(GLPMcop, para=c(t,3))$lambdaU*(2^t -1))^(-1/t)
print(c(Btheo, 4*Bform-1)) # [1] 0.8611903 0.8611900

## Not run:
# See the Note section but check Blomqvist Beta here:
blomCOP(cop=GLcop, para=c(6.043619)) # 0.8552863 (2p version)
blomCOP(cop=GLcop, para=c(5.6, 0.3)) # 0.8552863 (1p version)
## End(Not run)

---

### glueCOP

**Gluing Two Copulas**

**Description**

The glueing copula technique (Erdely, 2017, p. 71), given two bivariate copulas $C_A$ and $C_B$ and a fixed value $0 \leq \gamma \leq 1$ is

$$C_\gamma(u, v) = \gamma \cdot C_A(u/\gamma, v)$$

for $0 \leq u \leq \gamma$ and

$$C_\gamma(u, v) = (1 - \gamma) \cdot C_B((u - \gamma)/(1 - \gamma), v)$$
for $\gamma \leq u \leq 1$ and $\gamma$ represents the gluing point in $u$ (horizontal axis). The logic is simply the rescaling of $C_A$ to $[0, \gamma] \times [0, 1]$ and $C_B$ to $[\gamma, 1] \times [0, 1]$. Copula gluing is potentially useful in circumstances for which regression is non-monotone.

Usage

\[\text{glueCOP}(u, v, \text{para} = \text{NULL}, \ldots)\]

Arguments

- $u$: Nonexceedance probability $u$ in the $X$ direction;
- $v$: Nonexceedance probability $v$ in the $Y$ direction;
- $\text{para}$: A special parameter list (see Note) with a mandatory element of glue parameter $\gamma$; and
- $\ldots$: Additional arguments to pass to the copulas.

Value

Value(s) for the copula are returned.

Note

The following descriptions list in detail the structure and content of the $\text{para}$ argument:

- $\text{glue}$: The $\gamma$ gluing parameter;
- $\text{cop1}$: Function of the first copula $A$;
- $\text{cop2}$: Function of the second copula $B$;
- $\text{para1}$: Vector of parameters $\Theta_A$ for $A$; and
- $\text{para2}$: Vector of parameters $\Theta_B$ for $B$.

Author(s)

W.H. Asquith

References


See Also

COP, breveCOP, composite1COP, composite2COP, composite3COP, convexCOP
Examples

```r
# Not run:
para <- list(cop1=PLACKETTcop, para1=.2, cop2=GLcop, para2=1.2, glue=0.6)
densityCOPplot(cop=glueCOP, para=para)  #
# End(Not run)

# Not run:
# Concerning Nelsen (2006, exam. 3.3, pp. 59-61)
# Concerning Erdely (2017, exam. 5.1, p. 71)
# Concerning Erdely (2017, exam. 5.2, p. 75)
# Nelsen's example is a triangle with vertex at [G, 1].
# Erdley's example permits the construction using glueCOP from M and W.
"coptri" <- function(u,v, para=NA, ...) {
  p <- para[1]; r <- 1 - (1-p)*v
  if(length(u) > 1 | length(v) > 1) stop("only scalars for this function")
  if(0 <= u & u <= p*v & p*v <= p) { return(u)
  } else if( 0 <= p*v & p*v < u & u < r) { return(p*v)
  } else if( p <= r & r <= u & u <= 1 ) { return(u+v-1)
  } else { stop("should not be here in logic") }
}
"UsersCop" <- function(u,v, ...) { asCOP(u,v, f=coptri, ...) }
# Demonstrate Nelsen's triangular copula  (black dots)
UV <- simCOP(cop=UsersCop, para=0.35, cex=0.5, pch=16)
# Add Erdley's gluing of M() and W() copula (red circles)
para <- list(cop1=M, cop2=W, para1=NA, para2=NA, glue=0.35)
UV <- simCOP(cop=glueCOP, para=para, col=2, ploton=FALSE)
# We see in the plot that the triangular copulas are the same.

# For G = 0.5, Erdley shows Spearman Rho = 2*G-1 = 0, but
# Schweizer-Wolff = G^2 + (G-1)^2 = 0.5, let us check these:
para <- list(cop1=M, cop2=W, para1=NA, para2=NA, glue=0.5)
rhoCOP( cop=glueCOP, para=para) # -2.181726e-17
wolfCOP(cop=glueCOP, para=para) # 0.4999953
# So, rhoCOP() indicates independence, but wolfCOP() indicates
# dependence at the minimum value possible for a triangular copula.
# End(Not run)
```

gridCOP

**Compute a Copula on a Grid**

Description

Compute a grid of copula values. This function has the primary intention of supporting 3D renderings or 2D images of the *copulatic surface*. Users should be aware of the convention of the placement of the plotting origin and the various plotting mechanisms available to them in R. By convention copulatic surfaces start in lower left corner for \( u = v = 0 \), but matrix conventions (or at least how some functions plot matrices) start with the origin in the upper left.
Usage

gridCOP(cop=NULL, para=NULL, delta=0.05, transpose=TRUE, ...)

Arguments

cop A copula function;
para Vector of parameters or other data structure, if needed, to pass to the copula;
delta The $\Delta u = \Delta v$ of the grid edges;
transpose A logical to transpose the returned grid. This is needed if functions such as
image() in R are to be used for visualization (see last example in Examples
with composite2COP); and
...
Additional arguments to pass.

Value

The values for $C(u, v)$ are returned as a grid as an R matrix.

Author(s)

W.H. Asquith

See Also

EMPIRcopdf

Examples

## Not run:
the.grid <- gridCOP(cop=PSP)
the.grid[1,1] <- 0 # replace the NaN
image(the.grid) # ramps to the upper right
## End(Not run)

## Not run:
# See this composite copula also used in densityCOPplot() documentation.
para <- list(alpha=0.15, beta=0.90, kappa=0.06, gamma=0.96,
cop1=GHcop, cop2=PLACKETTcop, para1=5.5, para2=0.07)
GR <- gridCOP(cop=composite2COP, para=para, delta=0.005)
image(GR, col=terrain.colors(20)) # asymmetric, high curvature in top half
## End(Not run)
The Hoeffding Phi of a Copula or $L_p$ Distances (Independence, Radial Asymmetry, or Reflection Symmetry Forms)

Description

Compute the measure of association known as the Hoeffding Phi $\Phi_C$ of a copula from independence $(uv = \Pi; P)$ according to Cherubini et al. (2004, p. 164) by

$$\Phi_C = 3\sqrt{10} \int_\mathbb{T}^2 (C(u,v) - uv)^2 \, du \, dv,$$

and Nelsen (2006, p. 210) shows this as (and absolute value notation by Nelsen helps in generalization)

$$\Phi_C = \left(90 \int_\mathbb{T}^2 |C(u,v) - uv|^2 \, du \, dv\right)^{1/2},$$

for which $\Phi_C^2$ (the square of the quantity) is known as the dependence index. Gaíßer et al. (2010, eq. 1) have $\Phi_C^2$ as the Hoeffding Phi-Square, and their definition, when square-rooted, matches Nelsen’s listing.

A generalization (Nelsen, 2006) to $L_p$ distances from independence $(uv = \Pi; P)$ through the $L_p$COP function is

$$L_p \equiv \Phi_C(p) = \left(k(p) \int_\mathbb{T}^2 |C(u,v) - uv|^p \, du \, dv\right)^{1/p},$$

for a $p : 1 \leq p \leq \infty$ and where $k(p)$ is a normalization constant such that $\Phi_C(p) = 1$ when the copula $C$ is $M$ (see M) or $W$ (see W). The $k(p)$ (bivariate definition only) for other powers is given (Nelsen, 2006, exer. 5.44, p. 213) in terms of the complete gamma function $\Gamma(t)$ by

$$k(p) = \frac{\Gamma(2p + 3)}{2[\Gamma(p + 1)]^2},$$

which is implemented by the hoefCOP function. It is important to realize that the $L_p$ distances are all symmetric nonparametric measures of dependence (Nelsen, 2006, p. 210). These are symmetric because distance from independence is used as evident by “uv” in the above definitions.

Reflection/Radial and Permutation Asymmetry—Asymmetric forms similar to the above distances exist. Joe (2014, p. 65) shows two measures of bivariate reflection asymmetry or radial asymmetry (term favored in copBasic) as the distance between $C(u,v)$ and the survival copula $\hat{C}(u,v)$ ($\text{surCOP}$) measured by

$$L_\infty^{(\text{radsym})} = \sup_{0 \leq u,v \leq 1} |C(u,v) - \hat{C}(u,v)|,$$

or its $L_p^{(\text{radsym})}$ counterpart

$$L_p^{(\text{radsym})} = \left[\int_\mathbb{T}^2 |C(u,v) - \hat{C}(u,v)|^p \, du \, dv\right]^{1/p} \text{ with } p \geq 1,$$
where \( \hat{C}(u, v) = u + v - 1 + C(1 - u, 1 - v) \) and again \( p : 1 \leq p \leq \infty \). Joe (2014) does not seem to discuss and normalization constants for these two radial asymmetry distances.

Joe (2014, p. 66) offers analogous measures of bivariate permutation asymmetry \( (C(u, v) \neq C(v, u)) \) defined as

\[
L_{\infty}^{(\text{permsym})} = \sup_{0 \leq u, v \leq 1} |C(u, v) - \hat{C}(v, u)|,
\]

or its \( L_p^{(\text{permsym})} \) counterpart

\[
L_p^{(\text{permsym})} = \left( \int_{\mathbb{R}^2} |C(u, v) - \hat{C}(v, u)|^p \, du \, dv \right)^{1/p} \quad \text{with } p \geq 1,
\]

where \( p : 1 \leq p \leq \infty \). Again, Joe (2014) does not seem to discuss and normalization constants for these two permutation symmetry distances. Joe (2014, p. 65) states that the “simplest one-parameter bivariate copula families [and] most of the commonly used two-parameter bivariate copula families are permutation symmetric.”

The asymmetrical \( L_\infty \) and \( L_p \) measures identified by Joe (2014, p. 66) are nonnegative with an upper bounds that depends on \( p \). The bound dependence on \( p \) is caused by the lack of normalization constant \( k(p) \). In an earlier paragraph, Joe (2014) indicates an upper bounds of \( 1/3 \) for both (likely?) concerning \( L_\infty^{(\text{radsym})} \) and \( L_\infty^{(\text{permsym})} \). The numerical integrations for \( L_p^{(\text{radsym})} \) and \( L_p^{(\text{permsym})} \) can readily return zeros and often inspection of the formula for the \( C(u, v) \) itself would be sufficient to judge whether symmetry exists and hence the distances are uniquely zero.

Joe (2014, p. 66) completes the asymmetry discussion with three definitions of skewness of combinations of random variables \( U \) and \( V \): Two definitions are in \( \text{uvlmoms} \) (for \( U + V - 1 \) and \( U - V \)) and two are for \( V - U \) (\( \text{nuskewCOP} \)) and \( U + V - 1 \) (\( \text{nustarCOP} \)).

Usage

```r
hoefCOP( cop=NULL, para=NULL, p=2, as.sample=FALSE,
          sample.as.prob=TRUE, brute=FALSE, delta=0.002, ...)
```

```r
LpCOP( cop=NULL, para=NULL, p=2, brute=FALSE, delta=0.002, ...)
```

```r
LpCOPradsym( cop=NULL, para=NULL, p=2, brute=FALSE, delta=0.002, ...)
```

```r
LpCOPpermsym(cop=NULL, para=NULL, p=2, brute=FALSE, delta=0.002, ...)
```

Arguments

- **cop**
  A copula function;

- **para**
  Vector of parameters or other data structure, if needed, to pass to the copula;

- **p**
  The value for \( p \) as described above with a default to 2 to match the discussion of Nelsen (2006) and the Hoeffding Phi of Cherubini et al. (2004). Do not confuse \( p \) with \( d \) described in Note;

- **as.sample**
  A logical controlling whether an optional \( \text{R} \) data.frame in para is used to compute the \( \hat{C} \) (see Note). If set to -1, then the message concerning CPU effort will be supressed;
When as.sample triggered, what are the units incoming in para? If they are probabilities, the default is applicable. If they are not, then the columns are re-ranked and divided simply by \(1/n\)—more sophisticated empirical copula probabilities are not used (EMPIRcop).

Should brute force be used instead of two nested integrate() functions in R to perform the double integration;

The \(du\) and \(dv\) for the brute force (brute=TRUE) integration; and

Additional arguments to pass.

**Value**

The value for \(\Phi_C(p)\) is returned.

**Note**

Concerning the distance from independence, when \(p = 1\), then the Spearman Rho (rhoCOP) of a copula is computed where is it seen in that documentation that the \(k_1(1) = 12\). The respective values of \(k(p)\) for select integers \(p\) are

\[ p \mapsto [1, 2, 3, 4, 5] \equiv k(p) \mapsto \{12, 90, 560, 3150, 16600\}, \]

and these values are hardwired into hoefCOP and LpCOP. The integers for \(k_p\) ensures that the equality in the second line of the examples is TRUE, but the \(p\) can be a noninteger as well. Nelsen (2006, p. 211) reports that when \(p = \infty\) that \(L_\infty\) is

\[ L_\infty \equiv \Phi_C(\infty) = \Lambda_C = 4 \sup_{u,v \in \mathbb{I}} |C(u,v) - uv|. \]

A sample \(\hat{\Phi}_C\) (square root of the Hoeffding Phi-Square) based on nonparametric estimation generalized for \(d\) dimensions (\(d = 2\) for bivariate) is presented by Gaißer et al. (2010, eq. 10) for estimated probabilities \(\hat{U}_{ij}\) for the \(i\)th dimension and \(j\)th row (observation) for sample of size \(n\). Those authors suggest that the \(\hat{U}_{ij}\) be estimated from the empirical copula:

\[ \hat{\Phi}_C = \sqrt{h(d)[A + B]}, \]

where

\[ A = \left(\frac{1}{n}\right)^2 n \sum_{j=1}^{d} \sum_{k=1}^{d} \prod_{i=1}^{d} [1 - \max(\hat{U}_{ij}, \hat{U}_{ik})], \]

\[ B = \left(\frac{1}{3}\right)^d - \left(\frac{2}{n}\right) \left(\frac{1}{2}\right)^d \sum_{j=1}^{d} \prod_{i=1}^{d} [1 - \hat{U}_{ij}^2], \]

and the normalization constant is a function of dimension and is

\[ h(d)^{-1} = \frac{2}{(d+1)(d+2)} - \left(\frac{1}{2}\right)^d \frac{d! \prod_{i=0}^{d} i + (1/2)}{\prod_{i=0}^{d} i + (1/2)} + \left(\frac{1}{3}\right)^d. \]

set.seed(1); UV <- simCOP(n=1000, cop=PSP) hoefCOP(cop=PSP) # 0.4547656 (theo.) hoefCOP(para=UV, as.sample=TRUE) # 0.4892757 set.seed(1); UV <- simCOP(n=1000, cop=PSP, snv=TRUE) # std normal variates hoefCOP(para=UV, as.sample=TRUE, sample.as.prob=FALSE) # 0.4270324
Author(s)
W.H. Asquith

References

See Also
blomCOP, footCOP, giniCOP, rhoCOP, tauCOP, wolfCOP, joeskewCOP, uvlmoms

Examples
## Not run:
# Example (ii) Gaisser et al. (2010, p. 2574)
Theta <- 0.66 # Phi^2 = Theta^2 ---> Phi == Theta as shown
hoefCOP(cop=convex2COP, para=c(alpha=Theta, cop1=M, cop2=P)) # 0.6599886
rhoCOP(cop=PSP) == hoefCOP(cop=PSP, p=1) # TRUE
LpCOP(cop=PLACKETTcop, para=1.6, p=2.6) # 0.1445137 (Fractional p)
## End(Not run)

## Not run:
set.seed(938) # Phi(1.6; Plackett) = 0.1184489; L_1 = 0.1168737
UV <- simCOP(cop=PLACKETTcop, para=1.6, n=2000, ploton=FALSE, points=FALSE)
hoefCOP(cop=PLACKETTcop, para=1.6, p=200) # Large p near internal limits
L_1 <- 4*max(abs(PLACKETTcop(UV$U, UV$V, para=1.6) - UV$U*UV$V)) # p is infiny
# and finite n and arguably a sample-like statistic here, now on intuition try
# a more sample-like means
U <- runif(10000); V <- runif(10000)
L_2 <- 4*max(abs(EMPIRcop(U, V, para=UV) - U*V)) # 0.1410254 (not close enough)
## End(Not run)

## Not run:
para <- list(alpha=0.15, beta=0.90, kappa=0.06, gamma=0.96,
cop1=GHcop, cop2=PLACKETTcop, para1=5.5, para2=0.07)
LpCOPradsym( cop=composite2COP, para=para) # 0.02071164
LpCOPpermsym(cop=composite2COP, para=para) # 0.01540297
## End(Not run)

## Not run:
"MOcop.formula" <- function(u,v, para=para, ...) {
  alpha <- para[1]; beta <- para[2]; return(min(v*u^(1-alpha), u*v^(1-beta)))
}
"MOcop" <- function(u,v, ...) { asCOP(u,v, f=MOcop.formula, ...) }
The Hüsler–Reiss Extreme Value Copula

Description

The Hüsler–Reiss copula (Joe, 2014, p. 176) is

\[ C_{\Theta}(u, v) = HR(u, v) = \exp[-x\Phi(X) - y\Phi(Y)], \]

where \( \Theta \geq 0, x = -\log(u), y = -\log(v), \Phi(.) \) is the cumulative distribution function of the standard normal distribution, \( X \) and \( Y \) are defined as:

\[ X = \frac{1}{\Theta} + \frac{\Theta}{2}\log\left[\frac{x}{y}\right] \quad \text{and} \quad Y = \frac{1}{\Theta} + \frac{\Theta}{2}\log\left(\frac{y}{x}\right). \]

As \( \Theta \to 0^+ \), the copula limits to independence (\( \Pi_1; \Pi \)). The copula here is a bivariate extreme value copula (BEV), and the parameter \( \Theta \) requires numerical methods.

Usage

HRcop(u, v, para=NULL, ...)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>u</td>
<td>Nonexceedance probability ( u ) in the ( X ) direction;</td>
</tr>
<tr>
<td>v</td>
<td>Nonexceedance probability ( v ) in the ( Y ) direction;</td>
</tr>
<tr>
<td>para</td>
<td>A vector (single element) of parameters—the ( \Theta ) parameter of the copula; and</td>
</tr>
<tr>
<td>...</td>
<td>Additional arguments to pass.</td>
</tr>
</tbody>
</table>

Value

Value(s) for the copula are returned.

Author(s)

W.H. Asquith

References


See Also

P, GHcop, GLcop, tEVcop
### Examples

# Parameter Theta = pi recovery through the Blomqvist Beta (Joe, 2014, p. 176)
qnorm(1 - log(1+blomCOP(cop=HRcop, para=pi))/(2*log(2)))^(-1)

---

**isCOP.LTD**

*Is a Copula Left-Tail Decreasing*

---

**Description**

Numerically set a logical whether a copula is *left-tail decreasing* (LTD) as described by Nelsen (2006, pp. 192–193) and Salvadori *et al.* (2007, p. 222). A copula $C(u, v)$ is left-tail decreasing for LTD($V|U$) if and only if for any $v \in [0, 1]$ that the following holds

$$
\frac{\delta C(u, v)}{\delta u} \leq \frac{C(u, v)}{u}
$$

for almost all $u \in [0, 1]$. Similarly, a copula $C(u, v)$ is left-tail decreasing for LTD($U|V$) if and only if for any $u \in [0, 1]$ that the following holds

$$
\frac{\delta C(u, v)}{\delta v} \leq \frac{C(u, v)}{v}
$$

for almost all $v \in [0, 1]$ where the later definition is controlled by the wrtV=TRUE argument.

The LTD concept is associated with the concept of *tail monotonicity* (Nelsen, 2006, p. 191). Specifically, but reference to Nelsen (2006) definitions and geometric interpretations is recommended, LTD($V|U$) (or LTD($V|U$)) means that the probability $P[Y \leq y|X \leq x]$ (or $P[X \leq x|Y \leq y]$) is a nonincreasing function of $x$ (or $y$) for all $y$ (or $x$).

A positive LTD of either LTD($V|U$) or LTD($U|V$) implies positively quadrant dependency (PQD, *isCOP.PQD*) but the condition of PQD does not imply LTD. Finally, the accuracy of the numerical assessment of the returned logical by *isCOP.LTD* is dependent on the the “smallness” of the delta argument passed into the function.

**Usage**

```r
isCOP.LTD(cop=NULL, para=NULL, wrtV=FALSE, delta=0.005, ...)
```

**Arguments**

- **cop**  
  A copula function;

- **para**  
  Vector of parameters, if needed, to pass to the copula;

- **wrtV**  
  A logical to toggle between with respect to $v$ or $u$ (default);

- **delta**  
  The increment of $\{u, v\} \mapsto [0 + \Delta \delta, 1 - \Delta \delta, \Delta \delta]$ set by wrtV; and

- **...**  
  Additional arguments to pass to the copula or derivative of a copula function.

**Value**

A logical TRUE or FALSE is returned.
isCOP.permsym

Description

Numerically set a logical whether a copula is symmetric (Nelsen, 2006, p. 38), or has exchangable variables, or is permutation symmetric (Joe, 2014, p. 66). A copula \( C(u, v) \) is permutation symmetric if and only if for any \( \{u, v\} \in [0, 1] \) the following holds

\[
C(u, v) = C(v, u).
\]

The computation is (can be) CPU intensive.

Usage

isCOP.permsym(cop=NULL, para=NULL, delta=0.005, tol=1e-4, ...)
Arguments

- **cop**: A copula function;
- **para**: Vector of parameters, if needed, to pass to the copula;
- **delta**: The increment of \( \{u, v\} \mapsto [0 + \Delta \delta, 1 - \Delta \delta, \Delta \delta] \);
- **tol**: A tolerance on the check for symmetry, default 1 part in 10,000, which is the test for the \( \equiv 0 \) (zero equivalence, see source code); and
- ... Additional arguments to pass to the copula.

Value

A logical TRUE or FALSE is returned.

Author(s)

W.H. Asquith

References


See Also

- isCOP.radsym

Examples

```r
## Not run:
isCOP.perm sym (cop=GHcop, para=1.3) # TRUE
## End(Not run)
```

---

**Description**

Numerically determine the global property of the *positively quadrant dependency* (PQD) characteristic of a copula as described by Nelsen (2006, p. 188). The random variables \( X \) and \( Y \) are PQD if for all \( (x, y) \) in \( \mathcal{R}^2 \) when \( H(x, y) \geq F(x)G(x) \) for all \( (x, y) \) in \( \mathcal{R}^2 \) and thus by the copula \( C(u, v) \geq uv \) for all \( (u, v) \) in \( \mathcal{I}^2 \). Alternatively, this means that \( C(u, v) \geq \Pi \), and thus it can be said that it is globally “greater” than independence \( (uv = \Pi) \).

Nelsen (2006) shows that a copula is PQD when

\[
0 \leq \beta_C, 0 \leq \gamma_C, \text{ and } 0 \leq \rho_C \leq 3\tau_C,
\]

where \( \beta_C, \gamma_C, \rho_C \), and \( \tau_C \) are various copula measures of association or concordance that are respectively described in **blomCOP, giniCOP, rhoCOP**, and **tauCOP**. The concept of negatively quadrant
dependency (NQD) is the reverse: \( C(u,v) \leq \Pi \) for all \((u,v)\) in \(I^2\); so NQD is globally “smaller” than independence.

Conceptually, PQD is related to the probability that two random variables are simultaneously small (or simultaneously large) is at least as great as it would be if they were independent. The graph of a PQD copula lies on or above the copulatic surface of the independence copula \( \Pi \), and conversely a NQD copula lies on or below \( \Pi \).

Albeit a “global” property of a copula, there can be “local” variations in the PQD/NQD state. Points in \(I^2\) where \( C(u,v) - \Pi \geq 0 \) are locally PQD, whereas points in \(I^2\) where \( C(u,v) - \Pi \leq 0 \) and locally NQD. Lastly, readers are directed to the last examples in \texttt{wolfCOP} because as those examples involve the copulatic difference from independence \( C(u,v) - \Pi = C(u,v) - \Pi \) with 3-D renderings.

Usage

\[
\text{isCOP.PQD}(\text{cop=NULL, para=NULL, uv=NULL, empirical=FALSE, verbose=TRUE, \ldots})
\]

Arguments

- \texttt{cop} A copula function;
- \texttt{para} Vector of parameters or other data structure, if needed, to pass to the copula;
- \texttt{uv} An optional \texttt{R} \texttt{data.frame} of \texttt{U} and \texttt{V} nonexceedance probabilities \texttt{u} and \texttt{v} for the random variables \texttt{X} and \texttt{Y}. This argument triggers different value return behavior (see \texttt{Value});
- \texttt{empirical} A logical that will use sample versions for \textit{Gini Gamma}, \textit{Spearman Rho}, and \textit{Kendall Tau}. This feature is \textit{only} applicable if the copula is empirical and therefore the \texttt{para} argument is the \texttt{data.frame} of \texttt{u} and \texttt{v}, which will be passed along to sample version functions instead of copula (see \texttt{Note});
- \texttt{verbose} A logical that will report the four concordance measures; and
- \texttt{...} Additional arguments to pass, which are then passed to subordinate functions.

Value

If \(uv = \text{NULL}\) then a logical for the global property of PQD is returned but if argument \(uv\) is a \texttt{data.frame}, then an \texttt{R} \texttt{list} is returned, and that list holds the global condition in \texttt{global.PQD} and local condition assessments in \texttt{local.PQD} and \texttt{local.NQD}.

Note

The function \texttt{isCOP.PQD} will try brute force computations if subordinate calls to one or more functions fail. The user can use \texttt{...} to set the \texttt{delta} argument for \texttt{giniCOP}, \texttt{rhoCOP}, and \texttt{(or)} \texttt{tauCOP}.

This function is not guaranteed to work using a \textit{bivariate empirical copula} such as the following operation: \texttt{copPQD(cop=EMPIRcop, para=the.data)}. An evidently open problem for \texttt{copBasic} is how to support PQD assessment (either globally or locally) for empirical copulas. The \( \tau_C \) for the bivariate empirical copula example \texttt{brute=TRUE|FALSE} to unity and \( \gamma_C \) and \( \rho_C \) reach maximum number of subdivisions on the numerical integration and thus fail. If an empirical bivariate copula is “Tau’d” to itself, is \( \tau_C \equiv 1 \) guaranteed? The \( \tau_C \) computation relies on numerical partial derivatives.
of the copula, whereas the $\gamma_C$ and $\rho_C$ use the copula itself. It seems in the end that use of sample versions of $\gamma_C$, $\rho_C$, and $\tau_C$ would be appropriate and leave the $\beta_C$ as either copula or direct sample computation (see Examples).

**SPECIAL DEMONSTRATION 1**—Given the following.

```r
para <- list(cop1=PLACKETTcop, cop2=PLACKETTcop, para1=c(14.5), para2=c(1.45),
              alpha=0.51, beta=0.15, kappa=0.45, gamma=0.78)
D <- simCOP(n=500, cop=composite3COP, para=para, cex=0.5, col=1, pch=16)
```

the two different call types to `isCOP.PQD` for an empirical copula are illustrative:

```r
global.only <- isCOP.PQD(cop=EMPIRcop, para=D, empirical=TRUE)
```

and

```r
PQD.list <- isCOP.PQD(cop=EMPIRcop, para=D, empirical=TRUE, uv=D)
points(D, col=PQD.list$local.PQD+2, lwd=2) # red (if present) is local NQD
```

which in the former only returns the global PQD and the later returns an R list with global (global.PQD), local (local.PQD as well as local.NQD), and the four statistics ($\beta_C, \gamma_C, \rho_C, \tau_C$) used to determine global PQD.

**SPECIAL DEMONSTRATION 1**—Lastly, the `ctype=“bernstein”` argument to the empirical copula can be used. Repeated iterations of the following will show that local quadrant dependency can appear slightly different when the `bernstein` argument is present. The simulation sample size is reduced considerably for this second example because of the CPU effort triggered by the Bernstein extension (see `EMPIRcop`) having been turned on.

```r
para <- list(cop1=PLACKETTcop, cop2=PLACKETTcop, para1=14.5, para2=1.45,
              alpha=0.51, beta=0.15, kappa=0.45, gamma=0.78)
D <- simCOP(n=50, cop=composite3COP, para=para, cex=0.5, col=1, pch=16)
PQD.A <- isCOP.PQD(cop=EMPIRcop, para=D, empirical=TRUE, uv=D)
points(D, col=PQD.A$local.PQD+2, lwd=2) # red (if present) is local NQD
```

```r
PQD.B <- isCOP.PQD(cop=EMPIRcop, para=D, empirical=TRUE, uv=D, ctype=“bernstein”)
points(D, col=PQD.B$local.PQD+2, lwd=1, pch=3, cex=1.5)
```

**Author(s)**

W.H. Asquith

**References**


**See Also**

`blomCOP`, `giniCOP`, `rhoCOP`, `tauCOP`, `isCOP.LTD`, `isCOP.RTI`
Examples

```
## Not run:
isCOP.PQD(cop=PSP) # TRUE
## End(Not run)

## Not run:
# Example concerning Empirical Bivariate Copula and sample versions for comparison.
set.seed(10); n <- 1000
para <- list(cop1=PLACKETTcop, cop2=PLACKETTcop, para1=0.145, para2=1.45,
            alpha=0.81, beta=0.8)
D <- simCOP(n=n, cop=composite2COP, para=para, cex=0.5, col=rgb(0,0,0.2), pch=16)
#tauCOP(cop=EMPIRcop, para=D) # ??? but == 1
cor(D$U, D$V, method="kendall") # -0.3224705
blomCOP(cop=EMPIRcop, para=D) # -0.332
giniCOP(cop=EMPIRcop, para=D) # -0.392037
GINI <- sum(abs(rank(D$U)+rank(D$V)-n-1)) - sum(abs(rank(D$U)-rank(D$V)))
print(GINI/as.integer(n^2/2)) # -0.39996
rhoCOP(cop=EMPIRcop, para=D) # ??? but fails
cor(D$U, D$V, method="spearman") # -0.456694
lmomco::lcomoms2(D)$T2 # 1.0000000 -0.4568357
    # -0.4567859 1.0000000
## End(Not run)
```

isCOP.radsym

Is a Copula Radially Symmetric

Description

Numerically set a logical whether a copula is radially symmetric (Nelsen, 2006, p. 37) [reflection symmetric, Joe (2014, p. 64)]. A copula \( C(u,v) \) is radially symmetric if and only if for any \( \{u,v\} \in [0,1] \) either of the following hold

\[
C(u,v) = u + v - 1 + C(1 - u, 1 - v)
\]

or

\[
u + v - 1 + C(1 - u, 1 - v) - C(u,v) 
\equiv 0.
\]

Thus, if the equality of the copula \( C(u,v) = \hat{C}(u,v) \) (the survival copula), then radial symmetry exists: \( \text{COP} = \text{surCOP} \) or \( C(u,v) = C(1 - u, 1 - v) \). The computation is (can be) CPU intensive.

Usage

```
isCOP.radsym(cop=NULL, para=NULL, delta=0.005, tol=1e-4, ...)
```

Arguments

- `cop` A copula function;
- `para` Vector of parameters, if needed, to pass to the copula;
delta The increments of \( \{u, v\} \mapsto [0 + \Delta \delta, 1 - \Delta \delta, \Delta \delta] \);
tol A tolerance on the check for symmetry, default 1 part in 10,000, which is the
test for the \( \equiv 0 \) (zero equivalence, see source code); and
...
Additional arguments to pass to the copula or derivative of a copula function.

Value
A logical TRUE or FALSE is returned.

Note
An open research question possibly exists: Is a radially symmetric copula characterized by the L-
comoments for orders \( r \geq 3 \) as having values of zero? The author asks this question partly out of
intuition stemming from numerical experiments (some not show here) suggesting this condition, and
review of copula literature does not seem to directly address this question. Let us consider the two
symmetrical copulas: the parameterless \( \text{PSP} (u, v) \) (see \( \text{PSP} \)) and the \( \text{PL} (u, v) \) (see \( \text{PLACKETTcop} \))
with the \( \rho_C = 4.709 \) (see \( \text{rhoCOP} \)). The two copulas have different radial symmetries as shown
below.

```r
plackpar <- PLACKETTpar(rho=rhoCOP(cop=PSP)) # Spearman Rho = 0.4784176
isCOP.radSym(cop=PSP) # FALSE
isCOP.radSym(cop=PLACKETTcop, para=plackpar) # TRUE
```

Now, let us compute the L-comoments from the \texttt{lmomco R} package for \( n = 10,000 \) simulations
from each copula. The L-correlations are each about 0.48, which agree with the given \( \rho_C \).

```r
set.seed(639)
UVa <- simCOP(n=10000, cop=PSP, para=NA, graphics=FALSE)
set.seed(639)
UVb <- simCOP(n=10000, cop=PLACKETTcop, para=plackpar, graphics=FALSE)

lmomco::lcomoms2(UVa, nmom=4)$T3[2,1] # Only show L-coskew of V wrt U.

lmomco::lcomoms2(UVb, nmom=4)$T3[2,1] # Only show L-coskew of V wrt U.
```

The L-coskew for the \( \text{PSP} \) is about \(-0.13\) and that for the \( \text{PL} \) copula is about zero (\(< 0.003\)).
L-cokurtosis provides a similar result if \( T3 \) is changed to \( T4 \). The \( \text{PSP} \) L-cokurtosis is about 0.04,
whereas the \( \text{PL} \) L-cokurtosis is about \(< 0.004\).

Author(s)
W.H. Asquith

References

See Also

\( \text{isCOP.permsym} \)
Examples

# Radially symmetry is computationally intensive and relies on a gridded [0,1]x[0,1]
# space and laborious check on equality. Thus these examples are commented out for
# R --timings check. Note that the proof of radial symmetry absent of algebraic
# manipulation or verification is difficult and subject to the fineness of the grid
# to find a nonequality from which to immediately conclude FALSE.
## Not run:
isCOP.radsym(cop=P) # TRUE

para <- list(cop1=PLACKETTcop, cop2=M, para1=c(.3), para2=NA, alpha=0.8, beta=0.5)
isCOP.radsym(composite2COP, para=para) # FALSE

## End(Not run)
## Not run:

gh <- simCOP(n=34, cop=GHcop, para=theta, ploton=FALSE, points=FALSE) * 150  
# Pretend gh is real data, the * 150 is to clearly get into an arbitrary unit system.

# The sort=FALSE is critical in the following two calls
fakeU <- lmomco::pp(gh[,1], sort=FALSE) # Weibull plotting position i/(n+1)
fakeV <- lmomco::pp(gh[,2], sort=FALSE) # Weibull plotting position i/(n+1)

uv <- data.frame(U=fakeU, V=fakeV); # our U-statistics

set.seed(120); theta <- 2
gh <- simCOP(n=34, cop=GHcop, para=theta, ploton=FALSE, points=FALSE) * 150
# Pretend psp is real data, the * 150 is to clearly get into an arbitrary unit system.

# The sort=FALSE is critical in the following two calls
fakeU <- lmomco::pp(gh[,1], sort=FALSE) # Weibull plotting position i/(n+1)
fakeV <- lmomco::pp(gh[,2], sort=FALSE) # Weibull plotting position i/(n+1)

uv <- data.frame(U=fakeU, V=fakeV); # our U-statistics

isCOP.radsym(cop=EMPIRcop, para=uv) # FALSE
isCOP.LTD(cop=EMPIRcop, para=uv) # TRUE
isCOP.RTI(cop=EMPIRcop, para=uv) # FALSE
isCOP.PQD(cop=EMPIRcop, para=uv, empirical=TRUE) # TRUE

isCOP.radsym(cop=GHcop, para=theta) # FALSE
isCOP.LTD(cop=GHcop, para=theta) # TRUE
isCOP.RTI(cop=GHcop, para=theta) # TRUE
isCOP.PQD(cop=GHcop, para=theta) # TRUE

# Notice that isCOP.RTI is not the same for empirical and theoretical.
# This shows the difficulty in tail dependence parameter estimation for
# small samples (see Salvadori et al., 2007 p. 175).
## End(Not run)
isCOP.RTI

Description

Numerically set a logical whether a copula is right-tail increasing (RTI) as described by Nelsen (2006, pp. 192–193) and Salvadori et al. (2007, p. 222). A copula $C(u, v)$ is right-tail decreasing for $\text{RTI}(V | U)$ if and only if for any $v \in [0, 1]$, 

$$
\frac{\delta C(u, v)}{\delta u} \leq \frac{v - C(u, v)}{1 - u}
$$

for almost all $u \in [0, 1]$. Similarly, a copula $C(u, v)$ is right-tail decreasing for $\text{RTI}(U | V)$ if and only if for any $u \in [0, 1]$, 

$$
\frac{\delta C(u, v)}{\delta v} \leq \frac{u - C(u, v)}{1 - v}
$$

for almost all $v \in [0, 1]$ where the later definition is controlled by the $\text{wrtV}=$TRUE argument.

The RTI concept is associated with the concept of tail monotonicity (Nelsen, 2006, p. 191). Specifically, but reference to Nelsen (2006) definitions and geometric interpretations is recommended, RTI($V | U$) (or RTI($V | U$)) means that the probability $P[Y > y | X > x]$ (or $P[X > x | Y > y]$) is a nondecreasing function of $x$ (or $y$) for all $y$ (or $x$).

A positive RTI of either RTI($V | U$) or RTI($U | V$) implies positively quadrant dependency (PQD, isCOP.PQD) but the condition of PQD does not imply RTI. Finally, the accuracy of the numerical assessment of the returned logical by isCOP.RTI is dependent on the the smallness of the delta argument passed into the function.

Usage

isCOP.RTI(cop=NULL, para=NULL, wrtV=FALSE, delta=0.005, ...)

Arguments

cop A copula function;
para Vector of parameters, if needed, to pass to the copula;
wrtV A logical to toggle between with respect to $v$ or $u$ (default);
delta The increment of $\{u, v\} \rightarrow [0 + \Delta \delta, 1 - \Delta \delta, \Delta \delta]$ set by wrtV; and
... Additional arguments to pass to the copula or derivative of a copula function.

Value

A logical TRUE or FALSE is returned.

Author(s)

W.H. Asquith

References

See Also

isCOP.LTD, isCOP.PQD

Examples

## Not run:
isCOP.RTI(cop=P, delta=0.01)  # independence should be FALSE
# but function returns TRUE, note that same logic for isCOP.LTD returns FALSE.
isCOP.RTI(cop=PSP)            # TRUE : positive assoc.
isCOP.RTI(cop=PLACKETTcop, para=.15) # FALSE : negative assoc. Plackett
isCOP.RTI(cop=PLACKETTcop, para=15)  # TRUE : positive assoc. Plackett
isCOP.RTI(cop=PLACKETTcop, wrtV=TRUE, para=.15) # FALSE : negative assoc. Plackett
isCOP.RTI(cop=PLACKETTcop, wrtV=TRUE, para=15)  # TRUE : positive assoc. Plackett
## End(Not run)

Description

Is a general bivariate function a copula? Three properties are identified by Nelsen (2006, p. 10) for a bivariate copula \( C(u, v) \):

\[
C(u, 0) = 0 = C(0, v) \quad \text{Nelsen 2.2.2a}, \]

\[
C(u, 1) = u \quad \text{and} \quad C(1, v) = v \quad \text{Nelsen 2.2.2b}, \]

and for every \( u_1, u_2, v_1, v_2 \) in \( I^2 \) such that \( u_1 \leq u_2 \) and \( v_1 \leq v_2 \),

\[
C(u_2, v_2) - C(u_2, v_1) - C(u_1, v_2) + C(u_1, v_1) \geq 0 \quad \text{Nelsen 2.2.2c}.
\]

The last condition is known also as “2-increasing.” The \texttt{isfuncCOP} works along a gridded search in the domain \( I^2 = [0, 1] \times [0, 1] \) for the 2-increasing check with a resolution \( \Delta u = \Delta v = \text{delta} \). Because there are plenty of true copula functions available in the literature it seems unlikely that this function provides much production utility in investigations. This function is provided because part of the objectives of the \texttt{copBasic} package is for instructional purposes. The computational overhead is far too great for relative benefit to somehow dispatch to this function all the time using the other copula utilities in this package.

Usage

\texttt{isfuncCOP(cop=NULL, para=NULL, delta=0.002, ...)}

Arguments

\begin{itemize}
  \item \texttt{cop} \quad A potential bivariate copula function that accepts two arguments for the \textit{u} and \textit{v} and parameters along argument \texttt{para} with option of additional arguments through the \ldots argument;
  \item \texttt{para} \quad Vector of parameters or other data structure, if needed, to pass to the copula;
  \item \texttt{delta} \quad The \( \Delta u = \Delta v \) of the grid edges; and
  \item \ldots \quad Additional arguments to pass to the copula function.
\end{itemize}
Value

A logical value of TRUE or FALSE is returned.

Author(s)

S. Kloibhofer (idea and most code) and W.H. Asquith (documentation)

References


See Also

densityCOP

Examples

```r
## Not run:
"NelsenEx2.11" <- function(u,v, ...) { # Nelsen (2006, exer. 2.11, p. 16)
  if(length(u) > 1 & length(v) > 1 & length(u) != length(v)) return(NA)
  if(length(u) == 1) u <- rep(u, length(v))
  if(length(v) == 1) v <- rep(v, length(u))
  return(sapply(1:length(u), function(i) { upv <- u[i] + v[i]
    if(2/3 <= upv & upv <= 4/3) return(min(c(u,v,1/3,upv-(2/3))))
    max(u[i]+v[i]-1, 0) )))
}
isfuncCOP(cop=NelsenEx2.11) # FALSE
## End(Not run)
```

The Joe/B5 Copula (B5)

Description

The Joe/B5 copula (Joe, 2014, p. 170) is

$$ C_\Theta(u,v) = B_5(u,v) = 1 - [(1-u)^\Theta + (1-v)^\Theta - (1-u)^\Theta (1-v)^\Theta], $$

where $\Theta \in [1, \infty)$. The copula as $\Theta \to \infty$ limits to the comonotonicity coupla ($M(u,v)$ and $M$), as $\Theta \to 1^+$ limits to independence copula ($\Pi(u,v); P$). Finally, the parameter $\Theta$ is readily computed from a Kendall Tau ($\tauCOP$) by

$$ \tauC = 1 + \frac{2}{2-\Theta} (\psi(2) - \psi(1 + 2/\Theta)), $$

where $\psi$ is the digamma() function and as $\Theta \to 2$ then

$$ \tauC(\Theta \to 2) = 1 - \psi'(2) $$

where $\psi'$ is the trigamma() function.
**Usage**

\[ \text{JOcopB5}(u, v, \text{para=NULL, tau=NULL, ...}) \]

**Arguments**

- \( u \) Nonexceedance probability \( u \) in the \( X \) direction;
- \( v \) Nonexceedance probability \( v \) in the \( Y \) direction;
- \( \text{para} \) A vector (single element) of parameters—the \( \Theta \) parameter of the copula;
- \( \text{tau} \) Optional Kendall Tau; and
- \( ... \) Additional arguments to pass.

**Value**

Value(s) for the copula are returned. Otherwise if \( \text{tau} \) is given, then the \( \Theta \) is computed and a list having

- \( \text{para} \) The parameter \( \Theta \), and
- \( \text{tau} \) Kendall Tau.

and if \( \text{para=NULL} \) and \( \text{tau=NULL} \), then the values within \( u \) and \( v \) are used to compute Kendall Tau and then compute the parameter, and these are returned in the aforementioned list.

**Author(s)**

W.H. Asquith

**References**


**See Also**

M, P

**Examples**

# Upper tail dependency of Theta = pi --> 2 - 2^(1/pi) = 0.753131 (Joe, 2014, p. 171).
\[
\text{taildepCOP(cop=JOcopB5, para=pi)$lambdaU} \quad \# \text{0.753131}
\]

# Blomqvist Beta of Theta = pi (Joe, 2014, p. 171).
\[
\text{blomCOP(cop=JOcopB5, para=pi)} \quad \# \text{0.5521328}
\]
\[
3 - 4 \times (2 \times (1/2)^{pi} - (1/4)^{pi})^{(1/pi)} \quad \# \text{0.5521328}
\]

### Not run:
# A test near the limiting Theta for trigamma()
\[
\text{UV <- simCOP(cop=JOcopB5, para=2, n=10000)}
\]
\[
\text{para <- JOcopB5(UV[,1], UV[,2])$para}
\]
\[
\text{message("Tau difference ", round(2-para, digits=2), " is small.")} \quad \#
\]
\[
\text{## End(Not run)}
\]
**joeskewCOP**

*Joe’s Nu-Skew and the copBasic Nu-Star of a Copula*

**Description**

Compute the measure of permutation asymmetry, which can be thought of as bivariate skewness, named for the copBasic package as Nu-Skew $\nu_C$ of a copula according to Joe (2014, p. 66) by

$$\nu_C = 3E[UV - U^2V] = 6 \int\int_{I^2} (v - u) C(u,v) \, du \, dv.$$  

This definition is effectively the `type="nu"` for the function for which the multiplier 6 has been converted to 96 as explained in the Note.

Numerical results indicate $\nu_W \approx 0$ ($W$), $\nu_\Pi = 0$ ($P$), $\nu_M \approx 0$ ($M$), $\nu_{PL} \approx 0$ for all $\Theta$ (PLcop), and the $\nu_{GH} = 0$ (GHcop); copulas with mirror symmetry across the equal value line have $\nu_C = 0$.

Asymmetric copulas do exist. For example, consider an asymmetric Gumbel–Hougaard GH copula with $\Theta_p = (5, 0.8, p)$:

```r
optimize(function(p) { nuskewCOP(cop=GHcop, para=c(5,0.8, p)) },
c(0,0.99))$minimum
```

```r
UV <- simCOP(n=10000, cop=GHcop, c(5,0.8, 0.2836485)) # inspect the graphics
48*mean(UV$U*$V^2 - UV$U^2*UV$V) # -0.2847953 (not the 3rd parameter)
```

The minimization yields $\nu_{GH}(5,0.8,0.2836485) = -0.2796104$, which is close the expectation computed where $48 = 96/2$.

A complementary definition is supported, triggered by `type="nustar"`, and is computed by

$$\nu_\star^C = 12 \int\int_{I^2} (v + u) C(u,v) \, du \, dv - 4,$$

which has been for the copBasic package, $\nu_\star^C$ is named as Nu-Star, which the 12 and the −4 have been chosen so that numerical results indicate $\nu_\star^W = -1$ ($W$), $\nu_\star^\Pi = 0$ ($P$), and $\nu_\star^M = +1$ ($M$).

Lastly, the uvlmoms function provides for a quantile-based measure of bivariate skewness based on the difference $U - V$ that also is discussed by Joe (2014, p. 66).

**Usage**

```r
joeskewCOP(cop=NULL, para=NULL, type=c("nu", "nustar", "nuskew"),
as.sample=FALSE, brute=FALSE, delta=0.002, ...)
nuskewCOP(cop=NULL, para=NULL, as.sample=FALSE, brute=FALSE, delta=0.002, ...)
nustarCOP(cop=NULL, para=NULL, as.sample=FALSE, brute=FALSE, delta=0.002, ...)
```
Arguments

cop        A copula function;
para       Vector of parameters or other data structure, if needed, to pass to the copula;
type       The type of metric to compute (nu and nuskew are synonymous for \( \nu_C \) and nustar is for \( \hat\nu_C \));
brute      Should brute force be used instead of two nested integrate() functions to perform the double integration;
delta      The \( du \) and \( dv \) for the brute force integration using brute;
as.sample  A logical controlling whether an optional R data.frame in para is used to compute the sample \( \hat\nu \) or \( \hat\nu^* \) (see Note). If set to -1, then the message concerning CPU effort will be surpressed; and
...        Additional arguments to pass.

Details

The implementation of joeskewCOP for copBasic provides the second metric of asymmetry, but why? Consider the results that follow:

```r
demo <- joeskewCOP(cop=GHcop, para=c(5, 0.8, 0.2836485), type="nu")
# -0.2796104
demo <- joeskewCOP(cop=GHcop, para=c(5, 0.2836485, 0.8), type="nu")
# +0.2796103
demo <- joeskewCOP(cop=GHcop, para=c(5, 0.8, 0.2836485), type="nu")
# 0.3571276
demo <- joeskewCOP(cop=GHcop, para=c(5, 0.2836485, 0.8), type="nu")
# 0.3571279
tauCOP( cop=GHcop, para=c(5, 0.2836485, 0.8))
# 0.2443377
```

The demonstration shows—at least for the symmetry (switchability) of the 2nd and 3rd parameters (\( \pi_2 \) and \( \pi_3 \)) of the asymmetric GH copula—that the first definition \( \nu \) is magnitude symmetric but carries a sign change. The demonstration shows magnitude and sign stability for \( \nu^* \), and ends with Kendall Tau (tauCOP). Collectively, Kendall Tau (or the other symmetric measures of association, e.g. blomCOP, footCOP, giniCOP, hoefCOP, rhoCOP, wolfCOP) when combined with \( \nu \) and \( \nu^* \) might provide a framework for parameter optimization of the asymmetric GH copula (see below).

The asymmetric GH\((5,0.2836485,0.8)\) is not radial (isCOP.radsym) or permutation (isCOP.permsym), but if \( \pi_2 = \pi_3 \) then the resulting GH copula is not radially symmetric but is permutation symmetric:

```r
isCOP.radsym( cop=GHcop, para=c(5, 0.2836485, 0.8)) \# FALSE
isCOP.permsym(cop=GHcop, para=c(5, 0.2836485, 0.8)) \# FALSE
isCOP.radsym( cop=GHcop, para=c(5, 0.8, 0.8)) \# FALSE
isCOP.permsym(cop=GHcop, para=c(5, 0.8, 0.8)) \# TRUE
```

The use of \( \nu_C \) and \( \nu^*_C \), with a measure of association is just suggested above for parameter optimization. Suppose we have GH\((5,0.5,0.7)\) with Spearman Rho \( \rho = 0.4888 \), \( \nu = 0.001475 \), and \( \nu^* = 0.04223 \), and the asymmetric GH coupla is to be fit. Parameter estimation for the asymmetric GH is accomplished by
"fitGHcop" <- function(hats, assocfunc=rhoCOP, init=NA, eps=1E-4, ...) {
  H <- GHcop # shorthand for the copula
  "objfunc" <- function(par) {
    par[1] <- ifelse(par[1] < 1, return(Inf), exp(par[1])) # edge check
    par[2:3] <- pnorm(par[2:3]) # detransform
    hp <- c(assocfunc(H, par), nuskewCOP(H, par), nustarCOP(H, par))
    return(sum((hats-hp)^2))
  }
  # Theta=1 and Pi2 = Pi3 = 1/2 # as default initial estimates
  if(is.na(init)) init <- c(1, rep(1/2, times=2))
  opt <- optim(init, objfunc, ...); par <- opt$par
  para <- c( exp(par[1]), pnorm(par[2:3])
  names(para) <- c("Theta", "Pi2", "Pi3")
  fit <- c(assocfunc(H, para), nuskewCOP(H, para), nustarCOP(H, para))
  txt <- c("AssocMeasure", "NuSkew", "NuStar")
  names(fit) <- txt; names(hats) <- txt
  if(opt$value > eps) warning("inspect the fit")
  return(list(para=para, fit=fit, given=hats, optim=opt))
}

father <- c(5,.5,.7)
densityCOPplot(cop=GHcop, para=father, contour.col=8)
fRho <- rhoCOP( cop=GHcop, father)
fNu <- nuskewCOP(cop=GHcop, father)
fStar <- nustarCOP(cop=GHcop, father)

child <- fitGHcop(c(fRho, fNu, fStar))$para

The initial parameter estimate has the value $\Theta = 1$, which is *independence* for the one parameter $GH$. The two other parameters are set as $\pi_2 = \pi_3 = 1/2$ to be in the mid-point of their domain. The transformations using the log() $\leftrightarrow$ exp() and qnorm() $\leftrightarrow$ pnorm() functions in $R$ are used to keep the optimization in the viable parameter domain. The results produce a fitted copula of $GH_{(4.907,0.5006,0.7014)}$. This fit aligns well with the parent, and the three statistics are essentially matched during the fitting.

The $\nu^*_C$ can be similar to rhoCOP, but differences do exist. In the presence of radial symmetry, ($\nu_C = 0$), the $\nu^*_C$ is nearly equal to *Spearman Rho* for some copulas. Let us test further:

```
p <- 10^seq(0,2,by=.01)
s <- sapply(p, function(t) nustarCOP(cop=GHcop, para=c(t)))
r <- sapply(p, function(t) rhoCOP(cop=GHcop, para=c(t)))
plot(p,s, log="x", type="l", col=3, lwd=3); lines(p,r)
```
Now let us add some asymmetry

```r
s <- sapply(p, function(t) nustarCOP(cop=GHcop, para=c(t, 0.25, 0.75)))
r <- sapply(p, function(t) rhoCOP(cop=GHcop, para=c(t, 0.25, 0.75)))
plot(p, s, log="x", type="l", col=3, lwd=3); lines(p, r)
```

Now let us choose a different (the \textit{Clayton}) copula

```r
s <- sapply(p, function(t) nustarCOP(cop=CLcop, para=c(t)))
r <- sapply(p, function(t) rhoCOP(cop=CLcop, para=c(t)))
plot(p, s, log="x", type="l", col=3, lwd=3); lines(p, r)
```

**Value**

The value for $\nu_C$ or $\nu^*_C$ is returned.

**Note**

The $\nu_C$ definition is given with a multiplier of 6 on the integrals in order to agree with Joe (2014) relation that is also shown. However, in mutual parameter estimation experiments using a simple sum-of-square errors as shown in the \textbf{Details}, it is preferred to have $\nu_C$ measured on a larger scale. Where does the 96 then come from? It is heuristically made so that the upright and rotated \textit{cophalf} (see \textbf{Examples} under \texttt{asCOP} and \texttt{bilmoms} for this copula) acquires $\nu_C$ values of $+1$ and $-1$, respectively. As a result to make back comparisons to Joe results, the ratios of 96 are made in this documentation.

The source code shows slightly different styles of division by the sample size as part of the sample estimation of the statistics. The $\hat{\nu}$ using just division by the sample size as testing indicates that this statistic is reasonably unbiased for simple copula. The $\hat{\nu}^*$ with similar division is a biased statistic and the bias is not symmetrical in magnitude or sign it seems whether the $\hat{\nu}^*$ is positive or negative. The salient code is $\text{spm} <- \text{ifelse}(\text{corsgn} == -1, +2.4, +1.1)$ within the sources for which the corrections were determined heuristically through simulation, and \texttt{corsgn} is the sign of the sample Spearman Rho through the \texttt{cor()} function of \texttt{R}.

**Author(s)**

W.H. Asquith

**References**


**See Also**

\texttt{uvskew, blomCOP, footCOP, giniCOP, hoefCOP, rhoCOP, tauCOP, wolfCOP}
Examples

nuskewCOP(cop=GHcop, para=c(1.43, 1/2, 1))*(6/96) # 0.005886 (Joe, 2014, p. 184; 0.0059)

## Not run:
joeskewCOP(cop=GHcop, para=c(8, .7, .5)) # -0.1523491
joeskewCOP(cop=GHcop, para=c(8, .5, .7)) # +0.1523472
UV <- simCOP(n=1000, cop=GHcop, para=c(8, .7, .5)) # see the switch in curvature

## End(Not run)

## Not run:
para=c(19,0.3,0.8); set.seed(341)
nuskew <- nuskewCOP(cop=GHcop, para=para) # 0.3057744
UV <- simCOP(n=10000, cop=GHcop, para=para) # a large simulation
mean((UV$U - UV$V)^3)/(6/96) # 0.3127398

# Two other definitions of skewness follow and are not numerically the same.
uvskew(u=UV$U, v=UV$V, umv=TRUE) # 0.3738987 (see documentation
uvskew(u=UV$U, v=UV$V, umv=FALSE) # 0.3592739 (or documentation
# Yet another definition of skew, which requires large sample approximation
# using the L-comoments (3rd L-comoment is L-coskew).
lmomco::lcomoms2(UV)$T3 # L-coskew of the simulated values [1,2] and [2,1]
 [,1]        [,2]
[1,]  0.007398438  0.17076600
[2,] -0.061060260 -0.00006613
# See the asymmetry in the two L-coskew values and consider this in light of
# the graphic produced by the simCOP() called for n=10,000. The T3[1,1] is
# the sampled L-skew (univariate) of the U margin and T3[2,2] is the same
# but for the V margin. Because the margins are uniform (ideally) then these
# for suitable large sample must be zero because the L-skew of the uniform
# distribution is by definition zero.
#
# Now let us check the sample estimator for sample of size n=300, and the
# t-test will (should) result in acceptance of the NULL hypothesis.
S <- replicate(60, nuskewCOP(para=simCOP(n=300, cop=GHcop, para=para,
                               graphics=FALSE), as.sample=TRUE))
t.test(S, mu=nuskew)
# t = 0.004633, df = 59, p-value = 0.9963
# alternative hypothesis: true mean is not equal to 0.3057744
# 95 percent confidence interval:
# 0.2854282 0.3262150
# sample estimates:
# mean of x
# 0.3058216
## End(Not run)

## Not run:
# Let us run a large ensemble of copula properties that use the whole copula
# (not tail properties). We composite a Plackett with a Gumbel-Hougaard for
# which the over all association (correlation) sign is negative, but amongst
# these statistics with nuskew and nustar at the bottom, there are various
# quantities that can be extracted. These could be used for fitting.
set.seed(873)
para <- list(cop1=PLcop, cop2=GHcop, alpha=0.6, beta=0.9,
para1=.005, para2=c(8.3,0.25,0.7))
UV <- simCOP(1000, cop=composite2COP, para=para) # just to show
blomCOP(composite2COP, para) # -0.4078657
footCOP(composite2COP, para) # -0.2854227
hoefCOP(composite2COP, para) # +0.5713775
lcomCOP(composite2COP, para)$lcomUV[3] # +0.1816084
lcomCOP(composite2COP, para)$lcomVU[3] # +0.1279844
rhoCOP(composite2COP, para) # -0.5688417
rhobevCOP(composite2COP, para) # -0.2005210
tauCOP(composite2COP, para) # -0.4514693
wolfCOP(composite2COP, para) # +0.5691933
nustarCOP(composite2COP, para) # -0.5172434
nuskewCOP(composite2COP, para) # +0.0714987
## End(Not run)

---

**joint.curvesCOP**

**Compute Coordinates of the Marginal Probabilities given joint AND OR Probabilities**

### Description

Compute the coordinates of the bivariate marginal probabilities for variables $U$ and $V$ given selected probabilities levels $t$ for a copula $C(u, v)$ for $v$ with respect to $u$. For the case of a joint AND probability, symbolically the solution is

$$Pr[U \leq v, V \leq v] = t = C(u, v),$$

where $U \mapsto [t_i, u_j, u_{j+1}, \ldots, 1; \Delta t]$ (an irregular sequence of $u$ values from the $i$th value of $t_i$ provided through to unity) and thus

$$t_i \mapsto C(u = U, v),$$

and solving for the sequence of $v$. The index $j$ is to indicate that a separate loop is involved and is distinct from $i$. The pairings $\{u(t_i), v(t_i)\}$ for each $t$ are packaged as an R data.frame. This operation is very similar to the plotting capabilities in `level.curvesCOP` for level curves (Nelsen, 2006, pp. 12–13) but implemented in the function `joint.curvesCOP` for alternative utility.

For the case of a joint OR probability, the dual of a copula (function) or $\tilde{C}(u, v)$ from a copula (Nelsen, 2006, pp. 33–34) is used and symbolically the solution is:

$$Pr[U \leq v \text{ or } V \leq v] = t = \tilde{C}(u, v) = u + v - C(u, v),$$

where $U \mapsto [0, u_j, u_{j+1}, \ldots, t_i; \Delta t]$ (an irregular sequence of $u$ values from zero through to the $i$th value of $t$) and thus

$$t_i \mapsto \tilde{C}(u = U, v),$$

and solving for the sequence of $v$. The index $j$ is to indicate that a separate loop is involved and is distinct from $i$. The pairings $\{u(t_i), v(t_i)\}$ for each $t$ are packaged as an R data.frame.
Usage

`joint.curvesCOP(cop=NULL, para=NULL, type=c("and", "or"),
probs=c(0.5, 0.8, 0.90, 0.96, 0.99, 0.995, 0.998),
zero2small=TRUE, small=1E-6, divisor=100, delu=0.001, ...)`

Arguments

cop A copula function;
para Vector of parameters or other data structure, if needed, to pass to the copula;
type What type of joint probability is to be computed;
probs The joint probabilities \( t_i \) from which to compute the coordinates. The default values represent especially useful annual return period equivalents that are useful in hydrologic risk analyses;
zero2small A logical controlling whether exactly zero value for probability are converted to a small value and exactly unity values for probability are converted to the value \( 1 - \text{small} \); this logical is useful if transformation from probability space into standard normal variates or Gumbel reduced variates (see function `prob2grv()` in package `lmomco`) is later desired by the user for attendant graphics (see Examples section);
small The value for `small` described for `zero2small`;
divisor A divisor on a computation of a \( \Delta t \) for incrementing through the irregularly-spaced \( u \) domain as part of the coordinate computation (see source code);
delu A \( \Delta u \) for setup of the incrementing through the irregularly-space \( u \) domain as part of the coordinate computation (see source code); and
...
Additional arguments to pass to the `duCOP` function of `copBasic` or `uniroot()` function in R.

Value

An `R` list is returned with elements each of the given `probs`.

Note

The arguments `divisor` and `delu` provide flexibility to obtain sufficient smoothness in the coordinate curvatures for a given \( t \). The pairings \( \{u(t_i), v(t_i)\} \) for each \( t \) packaged as `data.frame`s within the returned `list` each have their own unique length, and this is the reason that a single “master” `data.frame` is not returned by this function.

Extended Example—The code below shows both types of joint probability being computed using the default `probs`. The plotting is made in Gumbel reduced variates (GRV; see `prob2grv` in package `lmomco`). This transformation is somewhat suitable for the magnitude variation in and at tail depth of the `probs`. Also with transformation is being used, the zero2small logical is kept at `TRUE`, which is appropriate. The zero2small being set is also useful if standard normal variate transformation (by the `qnorm()` function in R) were used instead.

The Gumbel–Hougaard copula (`GHcop`) and a reversed Gumbel–Hougaard copula `rGH` are composited together by `composite2COP`. These were chosen so that some asymmetry in the solutions
by `joint.curvesCOP` could be seen. We begin by specifying symmetrical plotting limits for later use and then creating a function for the reversed Gumbel–Hougaard and setting the parameters and composition weights:

```r
grvlim <- lmomco::prob2grv(c(0.25, 0.999)) # out to 1,000 years
"rGHcop" <- function(u, v, ...) {u + v - 1 + GHCop(1-u, 1-v, ...)}
para <- list(alpha=0.16, beta=0.67, cop1 =GHCop, cop2 =rGHcop,
para1=4.5, para2=2.2)
tau <- tauCOP(cop=composite2COP, para=para) # Tau = 0.351219
nuskew <- nuskewCOP(cop=composite2COP, para=para) # Nuskew = 0.084262
UV <- simCOP(n=1000, cop=composite2COP, para=para, snv=TRUE)
```

The code also computed the *Kendall Tau* (`tauCOP`) and *Nu-Skew* (`nuskewCOP`) and the results shown. The code finishes with a simulation by `simCOP` of the copula composition just for reference.

Next, we compute and plot the joint probability curves. The tolerance for the `uniroot` calls is reset from the `R` defaults because slight wobble in the numerical solutions exists otherwise. The `AND` and `OR` lists each provide data frames from which successive graphic calls plot the lines. The second `plot()` call is commented out so that both sets of joint probability curves are drawn on the same plot.

```r
AND <- joint.curvesCOP(cop=composite2COP, para=para, type="and",
             divisor=1000, tol=.Machine$double.eps)
plot(grvlim, grvlim, type="n",
        xlab="GUMBEL REDUCED VARIATE IN U", ylab="GUMBEL REDUCED VARIATE IN V")
for(t in sort(as.numeric(names(AND)))) {
        UV <- get(as.character(t), AND)
        lines(lmomco::prob2grv(UV$U), lmomco::prob2grv(UV$V))
        text(lmomco::prob2grv(median(UV$U)), lmomco::prob2grv(median(UV$V)),
             as.character(round(1/(1-t), digits=0))
}

OR <- joint.curvesCOP(cop=composite2COP, para=para, type="or",
             divisor=1000, tol=.Machine$double.eps)
for(t in sort(as.numeric(names(OR)))) {
        UV <- get(as.character(t), OR)
        lines(lmomco::prob2grv(UV$U), lmomco::prob2grv(UV$V), col=2)
        text(lmomco::prob2grv(median(UV$U)), lmomco::prob2grv(median(UV$V)),
             as.character(round(1/(1-t), digits=0), col=2)
}
```

Black Curves—The black curves represent the nonexceedance `joint and` condition. The black curves are a form of *level curve* (see `level.curvesCOP`), but it seems appropriate to not name them as such because Nelsen’s examples and others usually have the level curves on an even step interval of probability such as 10-percent level curves. The complement of nonexceedance `joint and` is the probability level that either or both random variables (say “hazards”) `U` or `V` causes “failure” at the respective return period.
Red Curves—The red curves represent the nonexceedance joint or (inclusive) condition. The complement of nonexceedance joint or (inclusive) is the probability level that both random variables (say “hazards”) U or V must simultaneously (or dually) occur for “failure” at the respective return period. Note that there is obviously asymmetry in the joint or curves.

Interpretation—Because the two hazards can “cooperate” to cause failure (see coCOP) for an equal level of protection (say 500-year event) relative to the complement of nonexceedance joint or (inclusive) condition (see surCOP), the marginal probabilities must be considerably higher. Users are encouraged to review copBasic-package and the figure therein.

Author(s)
W.H. Asquith

References

See Also
diagCOPatf, duCOP, jointCOP, joint.curvesCOP2, level.curvesCOP

Examples
# See Note section

### joint.curvesCOP2

**Compute Coordinates of the Marginal Probabilities given joint AND or OR Probability**

**Description**
Compute the coordinates of the bivariate marginal probabilities for variables U and V given selected probabilities levels t for a copula *C*(u, v) for u with respect to v. For the case of a joint and probability, symbolically the solution is

\[
\Pr[U \leq v, V \leq v] = t = C(u, v),
\]

where \( V \mapsto [t_i, t_j, t_{j+1}, \cdots, 1; \Delta] \) (an irregular sequence of v values from the ith value of \( t_i \) provided through to unity) and thus

\[
t_i \mapsto C(u, v = V),
\]

and solving for the sequence of u. The index \( j \) is to indicate that a separate loop is involved and is distinct from \( i \). The pairings \{u(\( t_i \)), v(\( t_i \))\} for each \( t \) are packaged as an R data.frame. This operation is very similar to the plotting capabilities in level.curvesCOP2 for level curves (Nelsen, 2006, pp. 12–13) but implemented in the function joint.curvesCOP2 for alternative utility.
For the case of a joint or probability, the dual of a copula (function) or \( \tilde{C}(u, v) \) from a copula (Nelsen, 2006, pp. 33–34) is used and symbolically the solution is:

\[
Pr[U \leq v \text{ or } V \leq v] = t = \tilde{C}(u, v) = u + v - C(u, v),
\]

where \( V \mapsto [0, v_j, v_{j+1}, \ldots, t_i; \Delta] \) (an irregular sequence of \( v \) values from zero through to the \( i \)th value of \( t \)) and thus

\[
t_i \mapsto \tilde{C}(u, v = V),
\]

and solving for the sequence of \( u \). The index \( j \) is to indicate that a separate loop is involved and is distinct from \( i \). The pairings \( \{u(t_i), v(t_i)\} \) for each \( t \) are packaged as an R data.frame.

Usage

```r
joint.curvesCOP2(cop=NULL, para=NULL, type=c("and", "or"),
probs=c(0.5, 0.8, 0.90, 0.96, 0.99, 0.995, 0.998),
zero2small=TRUE, small=1E-6, divisor=100, delv=0.001, ...)
```

Arguments

- `cop` A copula function;
- `para` Vector of parameters or other data structure, if needed, to pass to the copula;
- `type` What type of joint probability is to be computed;
- `probs` The joint probabilities for which to compute the coordinates. The default values represent especially useful annual return period equivalents that are useful in hydrologic risk analyses;
- `zero2small` A logical controlling whether precise zero value for probability are converted to a small value and precise unity values for probability are converted to the value 1 - small; this logical is useful if transformation from probability space into standard normal variates or Gumbel reduced variates (GRV; see function `prob2grv()` in package `lmomco`) is later desired by the user for attendant graphics (see Examples section);
- `small` The value for `small` described for `zero2small`;
- `divisor` A divisor on a computation of a \( \Delta \) for incrementing through the \( v \) domain as part of the coordinate computation (see source code);
- `delv` A \( \Delta v \) for setup of the incrementing through the \( v \) domain as part of the coordinate computation (see source code); and
- `...` Additional arguments to pass to the `duCOP` function of `copBasic` or `uniroot()` function.

Value

An R list is returned with elements each of the given `probs`.

Author(s)

W.H. Asquith
jointCOP

Compute Equal Marginal Probabilities Given a Single Joint AND or OR Probability for a Copula

Description

Given a single joint probability denoted as \( t \) for a copula \( C(u, v) \) numerically solve for bivariate marginal probabilities \( U \) and \( V \) such that they are also equal to each other (\( u = v = w \)). For the case of a joint and probability, the primary diagonal of the copula (Nelsen, 2006, pp. 12 and 16) is solved for by a simple dispatch to the diagCOPatf function instead. Symbolically the solution is

\[
\Pr[U \leq v, V \leq v] = t = C(w, w).
\]

For the case of a joint or probability, the dual of a copula (function) or \( \tilde{C}(u, v) \) from a copula (Nelsen, 2006, pp. 33–34; duCOP) is used where symbolically the solution is

\[
\Pr[U \leq v \text{ or } V \leq v] = t = \tilde{C}(u, v) = u + v - C(u, v),
\]

or

\[
\Pr[U \leq v \text{ or } V \leq v] = t = 2w - C(w, w).
\]

The function for type="or" tests \( C(0, 0) \) and if it returns NA or NaN then the lower limit for the rooting is treated as .Machine$double.eps instead of 0 (zero).

Usage

\[
\text{jointCOP}(t, \text{cop=NULL}, \text{para=NULL}, \text{type=c("and", "or"), ...})
\]
Arguments

t  The joint probability level \( t \);

cop  A copula function;

para  Vector of parameters or other data structure, if needed, to pass to the copula;

type  The type of joint probability is to be computed; and

...  Additional arguments to pass to the duCOP function of copBasic or uniroot() function.

Value

A vector of the equal \( u \) and \( v \) probabilities for the given type at the joint probability level of \( t \). The vector includes the \( t \) as the third element.

Note

ENSEMBLE 1—Counting and Copula Probabilities from a Massive Sample Size: Simulations can be used to check/verify select copula concepts. We begin with a Gumbel–Hougaard copula \( \text{GH}(u, v) = C_{\Theta}(u, v) \) having parameter \( \Theta = 1.5 \), which corresponds to a Kendall Tau \( \tau_C = 1/3 \) (GHcop). Next, simulate and count the number of either \( U \) or \( V \) exceeding the 99th percentile “event.” The event can occur either from the random variable \( U \) or from \( V \) with equal “loss.” If the event occurs in both \( U \) and \( V \), the loss is just the same as if \( U \) or \( V \) occurred. So, if a design were at the 100-year level and thus a 0.01 chance of loss each year, then 500 losses in 50,000 years would be expected.

```r
set.seed(89); n <- 50000
UV <- simCOP(n, cop=GHcop, para=1.5, graphics=FALSE)
length(UV$U[UV$U > 0.99 | UV$V > 0.99]) # 799 times (losses)
length(UV$U[UV$U > 0.99356 | UV$V > 0.99356]) # 500 times (losses)
```

Letting \( JP \) equal 0.99356, which forces the required acceptance of 500 losses for the design has conditions of \( UV$U > JP \) or \( UV$V > JP \) as well as condition of \( UV$U > JP \) and \( UV$V > JP \). These three conditions are captured using the structure of the R code listed. Up until now, manual searching resulted in a value for \( JP \) equaling 0.99356, which produces the 500 count losses (acceptable losses). Thus, \( JP \) is a marginal bivariate probability (in this case equality between \( U_{\text{crit.}} = V_{\text{crit.}} \), declared) necessary to attain a 99th percentile joint protection from loss. The magnitude for either \( U \) or \( V \) thus must exceed the 99th percentile, and this is what the code shows with 799 losses.

It is important to consider that unless \( U \) and \( V \) are in perfect positive correlation (e.g. \( M(u, v) \), M, Fréchet–Hoeffding upper-bound copula), that protection from loss needs to be higher than 0.99 if the marginal risk is set at that level. Continuing, if the 99th percentile is the 100-year event, then design criteria should be about 155 years instead \([\text{lmomco::prob2T}(0.99356)]\). The user can readily see this with the switch to perfect independence with the \( \text{GH}(u, v) \) copula with \( \Theta = 1 \) and produce quite different results or extreme correlation with say \( \Theta > 20 \).

To provide the protection for 500 exceedances in \( n = 50,000 \) trials and for purposes of demonstration, balance the protection between \( U \) and \( V \) by setting their probabilities equal, then the theoretical joint probability is
and these two probabilities, which in reality are actually based on same computation (\texttt{diagCOPatf}), and nearly are the same as \( JP = 0.99356 \) that was determined by the manual searching on the simulated data.

A \textbf{mutually inclusive and} condition can be arranged as follows, and it is implicit in the definition that both loss events occur at the same time:

\[
\text{length(UV} \cdot \text{U[UV} \cdot \text{U > 0.99 & UV} \cdot \text{V > 0.99])} \quad \# \quad 208 \text{ losses (simulated)}
\]

\[
\text{COP(0.99, 0.99, cop=GHcop, para=1.5)} \quad \# \quad 0.98402 \quad \text{(simulated)}
\]

\[
\text{surfuncCOP(0.99, 0.99, cop=GHcop, para=1.5)} \quad \# \quad 0.9841727 \quad \text{(theoretical)}
\]

What are the expected number of exceedances if designs for \( U \) and \( V \) are built at \( U = V = 0.99 \) marginal probabilities for protection?

\[
\text{coCOP(1-0.99, 1-0.99, cop=GHcop, para=1.5)} \quad \# \quad 791 \text{ losses (theoretical)}
\]

\[
(1-\text{COP(0.99, 0.99, cop=GHcop, para=1.5)}) \quad \# \quad 791 \text{ losses (theoretical)}
\]

Note that the 791 losses is nearly equal to 799, but obviously not 500 as one might incorrectly have imagined strictly in a univariate world.

Now a couple of questions can be asked about the \textbf{joint and} and \textbf{joint or} probabilities using the definition of a copula \texttt{COP} and then the \textit{dual of a copula (function)} \((\tilde{C}(u, v), \text{duCOP})\), respectively:

\[
\text{length(UV} \cdot \text{U[UV} \cdot \text{U <= 0.99 & UV} \cdot \text{V <= 0.99])} / \text{n} \quad \# \quad 0.98402 \quad \text{(simulated)}
\]

\[
\text{COP(0.99, 0.99, cop=GHcop, para=1.5)} \quad \# \quad 0.9841727 \quad \text{(theoretical)}
\]

\[
\text{length(UV} \cdot \text{U[UV} \cdot \text{U <= 0.99 | UV} \cdot \text{V <= 0.99])} / \text{n} \quad \# \quad 0.99584 \quad \text{(simulated)}
\]

\[
\text{duCOP(0.99, 0.99, cop=GHcop, para=1.5)} \quad \# \quad 0.9958273 \quad \text{(theoretical)}
\]

How about inversion of \( \tilde{C}(u, v) \) by \texttt{jointCOP} and check against the simulation?

\[
\text{jointCOP(0.99, cop=GHcop, para=1.5, type="or")[1]} \quad \# \quad 0.9763951 \quad \text{(theor.)}
\]

\[
\text{length(UV} \cdot \text{U[UV} \cdot \text{U <= 0.9763951 | UV} \cdot \text{V <= 0.9763951])/n} \quad \# \quad 0.98982 \quad \text{(simulated)}
\]

The second probability is a value quite near to 0.99. So, if one wants mutual loss protection, compute the inversion of the dual of a copula using \texttt{jointCOP(..., type="or")}. Let us say that 0.80 mutual loss protection is wanted

\[
\text{jointCOP(0.80, cop=GHcop, para=1.5, type="or")[1]} \quad \# \quad 0.6561286
\]

\[
n - \text{length(UV} \cdot \text{U[UV} \cdot \text{U <= 0.6561286 | UV} \cdot \text{V <= 0.6561286])} \# \quad 10049 \text{ losses (sim.)}
\]

\[
n - (1-0.8)n \quad \# \quad 10000 \text{ losses (theoretical)}
\]

The example here shows numerical congruence of 10,049 \( \approx \) 10,000. An opposing question is also useful. How about a \textbf{mutually exclusive or} condition as nonexceedances?
# The mutually exclusive OR as nonexceedances:
length((UV$U [ (UV$U <= 0.99 | UV$V <= 0.99) & !(UV$U <= 0.99 & UV$V <= 0.99)])) # 591 losses (simulated)

# The mutually exclusive OR as exceedances:
length(UV$U [ (UV$U > 0.99 | UV$V > 0.99) & !(UV$U > 0.99 & UV$V > 0.99)]) # 591 losses (simulated)

It is clear that $208 + 591 = 799$ as shown earlier. Readers are asked to notice how there are two ways to get at the 591 count. There are 208 mutual loss events and 591 occasions where either $U$ or $V$ is the causation of loss. For comparison, how many observed events by random variable?

length(UV$U[ UV$U > 0.99]) # 519 (simulated)
length(UV$U[ UV$V > 0.99]) # 491 (simulated)

which if they were perfectly uncorrelated ($P(u, v), P, independence copula$) would be $519 + 491 = 1,007$ losses. But for the simulations here, 799 losses occurred, so $1,007 - 799 = 208$ losses were at the same time caused by $U$ and $V$.

Both of the following lengths A and B are 799 and both represent a joint or condition—the operations do not represent a mutually exclusive or condition.

A <- length(UV$U[ UV$U > 0.99]) + length(UV$U[ UV$V > 0.99]) - length(UV$U[ UV$U > 0.99 & UV$V > 0.99])
B <- length(UV$U[ UV$U > 0.99 & UV$V > 0.99]) * A == B == 799

**ENSEMBLE 2—Simulation Study using a Real-World Sample Size:** Now with identities of sorts shown and described above, let us test a theoretically consistent version of a sample of size 150 repeated 1,000 times at the 98th percentile against loss by one or the other random variables or both for slightly correlated $U$ or $V$ again following the Gumbel–Hougaard copula. The losses incurred by mutual event occurrence is the same as if one or the other variables produced an event causing loss.

n <- 250; nsim <- 1000; EitherEvent <- 0.98; MutualEvent <- 0.99; Theta <- 1.5
PT <- jointCOP(EitherEvent, cop=GHcop, para=Theta, type="and")[1] # 0.9873537
DU <- jointCOP(MutualEvent, cop=GHcop, para=Theta, type="or")[1] # 0.9763951

This next code listing is a redundant example to the one that follows but it is shown anyway because a slight possibility of confusion in the vectorized conditional evaluations in R. This first example concretely changes the loss events into binary states and adds them up prior to the condition.

set.seed(894234)
EX1a <- sapply(1:nsim, function(i) {
    uv <- simCOP(n, cop=GHcop, para=Theta, graphics=FALSE)
    length(uv$U[ as.numeric(uv$U > PT) + as.numeric(uv$V > PT) >= 1 ])
} )
t.test(EX1a, mu=(1-EitherEvent) * n)

The expected count $E[EX1a] = 5$ and the simulation run yielded 5.058. The t.test() function in R results in a statistically insignificant difference. The following two example use a similar there. For sake of both code brevity and clarity, the examples here all restart the simulations at the expense of computation time.
The expected count $E[EX1b] = 5$—the same results are shown as in the first example listing. Next, let us demonstrate the dual of a copula for mutually occurring events.

\begin{verbatim}
set.seed(894234)
EX2 <- sapply(1:nsim, function(i) {
    uv <- simCOP(n, cop=GHcop, para=Theta, graphics=FALSE)
    length(uv$U[uv$U > DU & uv$V > DU])
})
t.test(EX2, mu=(1-MutualEvent) * n)
\end{verbatim}

The expected count $E[EX2] = 2.5$ and the simulation run yielded 2.49. The t.test() again results in a statistically insignificant difference.

Now the $U = V = 0.9873537$ for the “and” and $U = V = 0.9763951$ for the “or” are not equal marginal probabilities. Taking the larger of the two marginal probabilities, the actual joint protection from mutual event occurrence can be computed:

\begin{verbatim}
duCOP(0.9873537, 0.9873537, cop=GHcop, para=Theta) # 0.9947075
(1-0.9947075)*n # which is about 1.32 events per 250 trials.
\end{verbatim}

So, the larger protection in terms of joint probabilities provided by EitherEvent at 0.98 instead of MutualEvent at 0.99 with respective protection at the 0.9873537 provides a MutualEvent protection of 0.9947075.

\begin{verbatim}
set.seed(894234)
EX3 <- sapply(1:nsim, function(i) {
    uv <- simCOP(n, cop=GHcop, para=Theta, graphics=FALSE)
    length(uv$U[uv$U > 0.9873537 & uv$V > 0.9873537])
})
t.test(EX3, mu=(1-0.9947075) * n)
\end{verbatim}

The expected count $E[EX3] = 1.32$ and the simulation run yielded 1.31. The t.test() again results in a statistically insignificant difference, and thus the result indistinguishable from the expectation.

Author(s)

W.H. Asquith

References


See Also

diagCOPatf, duCOP, joint.curvesCOP, level.curvesCOP
Examples

```r
 jointCOP(0.50, cop=GHcop, para=1.5, type="and") # 0.6461941 0.6461941 0.5000000
 jointCOP(2/3, cop=GHcop, para=1.5, type="or") # 0.4994036 0.4994036 0.6666667
```

# See extended code listings and discussion in the Note section

---

**kfuncCOP**

The Kendall (Distribution) Function of a Copula

Description

To begin, there are at least three terms in the literature for what appear as the same function supported by the *kfuncCOP* function. The Kendall Function also is known as Kendall Distribution Function (Nelsen, 2006, p. 163) and Kendall Measure (Salvadori et al., 2007, p. 148). Each of these is dealt with in sequel to set the manner of the rather lengthy documentation for this function.

**KENDALL FUNCTION**—The Kendall Function \( F_K(z) \) (Joe, 2014, pp. 419–422) is the cumulative distribution function (CDF) of the vector \( U = (U_1, U_2, \ldots) \) or \( U = (u, v) \) (bivariate) where \( U \) is distributed as the copula: \( U \sim C(u, v) \). Letting \( Z \) be the random variable for \( C(u, v) : Z = C(u, v) \), the Kendall Function is defined as

\[
F_K(z; C) = \Pr[Z \leq z; U \sim C(u, v)],
\]

where \( F_K \) is the nonexceedance probability of the joint probability \( z \) stemming from the \( C \). Note that unlike its univariate counterpart, \( F_K(z) \) is rarely uniformly distributed (Nelsen et al., 2001, p. 278). The inverse \( F_K^{-1}(z) \) is implemented by the *kfuncCOPinv* function, which could be used for simulation of the correct joint probability using a single uniformly distributed \( \sim U(0,1) \) random variable. A reminder is needed that \( Z \) is the joint probability and \( F_K(z) \) is the Kendall Function.

Joe (2014) and others as cited list various special cases of \( F_K(z) \), inequalities, and some useful identities suitable for validation study:

- For \( M(u, v) \) (see \( M \)): \( F_K(z) = z \) for all \( 0 < z < 1 \) for all \( d \geq 2 \) dimensions;
- For \( W(u, v) \) (see \( W \)): \( F_K(z) = 1 \) for all \( 0 < z < 1 \) for \( d = 2 \) (bivariate only);
- For \( \Pi(u, v) \) (see \( P \)): \( F_K(z) = z - z \log z \) for \( 0 < z < 1 \) for \( d = 2 \) (bivariate only);
- For any \( C \): \( z \leq F_K(z) \) for \( 0 < z < 1 \); and
- For any \( C \): \( E[Z] = 1 - \int_0^1 F_K(t) \, dt \geq z \) (Nelsen, 2001, p. 281) — \( Z \) expectation, not \( F_K \)!
- For any \( C \): \( \tau_C = 3 - 4 \int_0^1 F_K(t) \, dt \) (Nelsen, 2006, p. 163; see *tauCOP* [Examples]).
- For any \( C \): \( F_K(t) \) does not uniquely determine the copula.

The last item is from Durante and Sempi (2015, p. 118), and later discussion herein will concern an example of theirs. By coincidence within a few days before receipt of the Durante and Sempi book, experiments using *kfuncCOP* suggested that numerically the Galambos (GLcop), Gumbel–Hougaard (GHcop), and Hülsler–Reiss (HRcop) extreme value copulas for the same Kendall Tau \( \tau_C \) all have the same \( F_K(t) \). Therefore, do all EV-copulas have the same Kendall Function? Well in fact, they do and Durante and Sempi (2015, p. 207) show that \( F_K(z) = z - (1 - \tau_C)z \log(z) \) for an EV-copula.

Joe (2014, p. 420) also indicates that strength of lower-tail dependence \( \text{taildepCOP} \) affects \( F_K(z) \) as \( z \to 0^+ \), whereas strength of upper-tail dependence affects \( F_K(z) \) as \( z \to 1^- \). (A demonstration...
of tail dependence dependence is made in section Note. Also compared to comonotonicity copula \([M]\) there are no countermonotonicity copula \([W_{d>2}]\) for dimensions greater the bivariate (Joe, 2014, p. 214).

Joe (2014) does not explicitly list an expression of \(F_K(z)\) that is computable directly for any \(C(u,v)\), and Nelsen (2006, p. 163) only lists a form (see later in documentation) for Archimedean copulas. Salvadori et al. (2007, eq. 3.47, p. 147) also list the Archimedean form; however, Salvadori et al. (2007, eq. 3.49, p. 148) also list a form computable directly for any \(C(u,v)\). Considerable numerical experiments and derivations involving the \(\Pi(u,v)\) copula and results for \(K_C(z)\) shown later, indicate that the correct Kendall form for any \(C(u,v)\) is

\[
F_K(z) \equiv z + \int_z^1 \frac{\delta C(u,t)}{\delta u} \, du,
\]

where \(t = C^{(-1)}(u,z)\) for \(0 \leq z \leq 1\), \(t\) can be computed by the \texttt{COPinv} function, and the partial derivative \(\delta C(u,t)/\delta u\) can be computed by the \texttt{derCOP} function. It is a curiosity that this form is not in Joe (2014), Nelsen et al. (2001, 2003), or Nelsen (2006), but actually in Salvadori et al. (2007).

**KENDALL MEASURE**—The actual expression for any \(C(u,v)\) by Salvadori et al. (2007, eq. 3.49, p. 148) is for Kendall Measure \((K_C)\) of a copula:

\[
K_C(z) = z - \int_z^1 \frac{\delta C(u,t)}{\delta u} \, du,
\]

where \(t = C^{(-1)}(u,z)\) for \(0 \leq z \leq 1\). Those authors report that \(K_C(z)\) is the CDF of a random variable \(Z\) whose distribution is \(C(u,v)\). This is clearly appears to be the same meaning as Joe (2014) and Nelsen (2006). The minus "−" in the above equation is very important. Salvadori et al. (2007, p. 148) report that “the function \(K_C(z)\) represents a fundamental tool for calculating the return period of extreme events.” The complement of \(K_C(z)\) is \(\overline{K}_C(z)\), and the \(K_C(z)\) inverse

\[
\frac{1}{1 - K_C(z)} = \frac{1}{\overline{K}_C(z)} = T_{KC}
\]

is referred to as a secondary return period (Salvadori et al., 2007, pp. 161–170).

**KENDALL DISTRIBUTION FUNCTION**—Nelsen (2006, p. 163) defines the Kendall Distribution Function (say \(K_C^*(t)\)) as

\[
K_C^*(t) = t - \frac{\phi(t)}{\phi'(t^+)�},
\]

where \(\phi(t)\) is a generator function of an Archimedean copula and \(\phi'(t^+)\) is a one-sided derivative (Nelsen, 2006, p. 125), and \(\phi(t) = \phi(C(u,v)) = \phi(u) + \phi(v)\). This same form is listed by Salvadori et al. (2007, eq. 3.47).

Nelsen (2006) does not seem to list a more general definition for any \(C(u,v)\). Because there is considerable support for Archimedean copulas in \(\text{R}\), \texttt{copBasic} has deliberately been kept from being yet another Archimedean-based package. This is made for more fundamental theory and
The similarity of $F_K(z)$, $K_C(z)$, and $K_C^*(t)$, however, is obvious—research shows that there are no syntactic differences between $F_K(z)$ and $K_C(t)$ and $K_C^*(z)$—they all are the CDF of the joint probability $Z$ of the copula. Consider now that Salvadori et al. show $K_C$ having the form $a - b$ and not a form $a + b$ as previously shown for $F_K(z)$. Which form is thus correct? The greater bulk of this documentation seeks to answer that question, and it must be concluded that Salvadori et al. (2007, eq. 3.49) definition for $K_C(z)$ has a typesetting error.

**Usage**

```r
kfuncCOP(z, cop=NULL, para=NULL, wrtV=FALSE, as.sample=FALSE, verbose=FALSE,...)
kmeasCOP(z, cop=NULL, para=NULL, wrtV=FALSE, as.sample=FALSE, verbose=FALSE,...)
```

**Arguments**

- `z` The values for $z$;
- `cop` A copula function;
- `para` Vector of parameters or other data structure, if needed, to pass to the copula;
- `wrtV` A logical to toggle between with respect to $v$ or $u$ (default);
- `as.sample` A control on whether an optional R data.frame in `para` is used to compute the empirical $\hat{F}_K(z)$. Let vector length of `para` be denoted $m$ and $i = (1, \ldots, m)$, if `as.sample=TRUE`, then for each of the probability pairs in `para`, the empirical copula (`EMPIRcop`) function with additional arguments ... is used to generate a vector $(0, F^i_{K,m}, 1)$ of length $m + 2$, then a vector of $(0, z^2_i, 1)$ again of length $m + 2$ where $z^2_i = (i - 0.5)/m$ is created and linear interpolation by the approx() function in R for each of the $z$ values in $z$ is used to estimate $\hat{F}_K(z)$ (see source code). If `as.sample="genest"`, then the $\hat{F}_K(z)$ is estimated without interpolation using a simple empirical copula basis and “pseudo-observations” after Genest et al. (2006, p. 339). The default $\hat{F}_K$ (as.sample=TRUE) is the linear interpolation based on the default call of Weibull form of the empirical copula, but that can be confirmed by `ctype="weibull"` (see Note);
- `verbose` A logical supressing warnings from `integrate()` in R that are usually related to “integral divergence” for $z \to 0^\tau$. The constructed behavior of `kfuncCOP` is to return $F_K(z \to 0^\tau) = 0$ if numerical integration returns `NULL`, and such a construction is made to avoid “end points not of opposite sign” during $F_K(z)$ inversion by `kfuncCOPinv`; and
- `...` Additional arguments to pass.

**Value**

The value(s) for $F_K(z)$ is returned.
Note

VALIDATION STUDY—A validation study using the Independence copula ($\Pi = uv; P$) with theoretical results of Joe (2014) and empiricism is readily performed using the expression for $F_K(z)$:

```
Z <- sort(c(0.01, seq(0,1, by=0.05), 0.99))  # ** Joint probabilities **
UV <- simCOP(n=4000, cop=P, graphics=FALSE); kendF <- Z - Z*log(Z)
emp.kendF <- kfuncCOP(Z, para=UV, as.sample="genest")  # emp. Kendall func
theo.kendF <- kfuncCOP(Z, cop=P)  # theo. Kendall func, numeric integration
plot(Z, kendF, type="l", col=3, lwd=4, lty=2, xlab="COPULA(u,v) VALUE [JOINT PROBABILITY]",
     ylab="KENDALL FUNCTION, AS NONEXCEEDANCE PROBABILITY")  # analytical
points(Z, kendF, col=3, lwd=1, lty=2, pch=16)  # green (theo. values)
points(Z, emp.kendF, col=4, lwd=2, cex=1.5)  # blue (empirical values)
lines(Z, theo.kendF, col=2)  # red (theoretical line by numerical integration)
mtext("Kendall Functions: Independence Copula")
```

The figure so produced shows that the theoretical relation in Joe (2014) is correct because the empirical values from the simulated sample (Empirical Kendall Function; Nelsen et al., 2003) match other curves quite well. Rerunning the above code with either M (M) or W (W) copulas will show that the special cases listed above are consistent with the empirical estimates. The case of W(u,v) is degenerate at z = 0 so the empirical computation is in error for the smallest z given because of interpolation. The M copula has $F_K$ along the equal value line (1:1) line.

Now consider a more comprehensive demonstration using the N4212cop copula with some relatively weak dependence in $\Theta = 1.17$.

```
Theta <- 1.17; print(rhoCOP(cop=N4212cop, para=Theta))  # Spearman Rho = 6/10
Z <- sort(c(0.01, seq(0,1, by=0.05), 0.99))  # ** Joint probabilities **
UV <- simCOP(n=16000, cop=N4212cop, para=Theta, graphics=TRUE)
empir.kendF <- kfuncCOP(Z, as.sample="weibull")  # analytical
kwrtU <- kfuncCOP(Z, cop=N4212cop, para=Theta, wrtV=FALSE)
kwrtV <- kfuncCOP(Z, cop=N4212cop, para=Theta, wrtV=TRUE)
plot(Z, empir.kendF, type="p", col=2, lwd=7, lty=2, xlab="COPULA(u,v) VALUE [JOINT PROBABILITY]",
     ylab="KENDALL FUNCTION, AS NONEXCEEDANCE PROBABILITY")
points(Z, empir.kendF, col=4, lwd=2, cex=1.5)  # blue (empirical values)
lines(Z, kwrtU, col=4, lwd=4, lty=2); lines(Z, kwrtV, col=3, lwd=1, lty=2)
```

The figure so produced again shows congruence between the two theoretical computations and the empirical curve. Now consider another comprehensive demonstration using the PLACKETTcop copula with some strong negative dependence in $\Theta = 0.04$.

```
Theta <- 0.04
Z <- sort(c(0.01, seq(0,1, by=0.05), 0.99))  # ** Joint probabilities **
UV <- simCOP(n=2600, cop=PLACKETTcop, para=Theta, graphics=TRUE)
empir.kendF <- kfuncCOP(Z, as.sample="hazen")  # analytical
kwrtU <- kfuncCOP(Z, cop=PLACKETTcop, para=Theta, wrtV=FALSE)
kwrtV <- kfuncCOP(Z, cop=PLACKETTcop, para=Theta, wrtV=TRUE)
plot(Z, empir.kendF, type="p", col=2, lwd=7, lty=2, xlab="COPULA(u,v) VALUE [JOINT PROBABILITY]",
     ylab="KENDALL FUNCTION, AS NONEXCEEDANCE PROBABILITY")
points(Z, empir.kendF, col=4, lwd=2, cex=1.5)  # blue (empirical values)
lines(Z, kwrtU, col=4, lwd=4, lty=2); lines(Z, kwrtV, col=3, lwd=1, lty=2)
```

The figure so produced again shows congruence between the two theoretical computations and the empirical curve. Now consider another comprehensive demonstration using the PLACKETTcop copula with some strong negative dependence in $\Theta = 0.04$. 
The figure so produced again shows congruence between the two theoretical computations and the empirical curve.

Another comparison of \( F_K(z) \) is useful and concerns lower- and upper-tail dependency parameters (\emph{taildepCOP}) with a comparison of three different copula all having the same Kendall Tau. The following code computes and draws the respective \( F_K(z) \):

```
# Given a Kendall Tau of 0.4 and the GHcop, N4212, and Plackett copulas
# parameters respectively are:
Phi <- 1.666667; Nu <- 1.111119; Mu <- 6.60344
Z <- seq(0.005, 0.995, by=0.005) # ** Joint probabilities **
GHkenf <- kfuncCOP(Z, cop=GHcop, para=Phi, wrtV=FALSE)
N4212kenf <- kfuncCOP(Z, cop=N4212cop, para=Nu, wrtV=FALSE)
PLkenf <- kfuncCOP(Z, cop=PLACKETTcop, para=Mu, wrtV=FALSE)
plot(qnorm(GHkenf), Z, type="l", col=1, lwd=2, xlim=c(-3,3), ylim=c(0,1),
      xlab="KENDALL FUNCTION, AS STANDARD NORMAL VARIATES",
      ylab="COPULA(u,v) VALUE, AS NONEXCEEDANCE PROBABILITY") # black curve
lines(qnorm(N4212kenf), Z, col=2, lwd=2) # red curve
lines(qnorm(PLkenf), Z, col=4, lwd=2) # blue curve
```

The red curve for the \textbf{N4212}(\( \Theta=1.1 \)) copula (\texttt{N4212cop}) is higher on the left, which shows the impact of its larger lower-tail dependency (\( \lambda^L_{N4212}=0.535 \)), whereas the black curve for the \textbf{GH}(\( \Theta=1.67 \)) copula (\texttt{GHcop}) is similarly (about same magnitude) higher on the right, which shows the impact of its larger upper-tail dependency (\( \lambda^U_{GH}=0.484 \)). The blue curve for the \textbf{PL}(\( \Theta = 6.60 \)) copula (\texttt{PLACKETTcop}) nearly overwrites on the left the GH curve, which is a reflection of both copulae having zero lower-tail dependencies (\( \lambda^L_{GH} = \lambda^L_{PL} = 0 \)). Finally, as anticipated by \( \lambda^U \), the curve on the right for the \textbf{N4212} is just slightly larger for the PL because the \( \lambda^U_{PL}=0 < \lambda^U_{N4212}=0.134 \) (a small difference however), and again on the right, the \textbf{N4212} curve is considerably smaller than the GH because \( \lambda^U_{N4212}=0 < \lambda^U_{GH}=0.484 \).

Durante and Sempi (2015, p. 118) provide an example of two copula (\( C_1 \) and \( C_2 \)) having the same \( F_K(z) = \min(2z, 1) \). Let us check that out:

```
"C1" <- function(u,v, ...) {
  if (length(u) == 1) { u <- rep(u, length(v)) } else
  if (length(v) == 1) { v <- rep(v, length(u)) }
  sapply(1:length(u), function(i) {
    min(c(u[i], max(c(v[i]/2, u[i]+v[i]-1))))
  })
}

"C2" <- function(u,v, ...) {
  if (length(u) == 1) { u <- rep(u, length(v)) } else
  if (length(v) == 1) { v <- rep(v, length(u)) }
  sapply(1:length(u), function(i) {
    g <- 1/2
    max(c(0, u[i]+v[i]-1, min(c(u[i], v[i]-g)), min(c(u[i]-g, v[i]))))
  })
}
```
DSkf <- function(t) sapply(t, function(z) min(c(2*z, 1)))
zs <- seq(0,1, by=.01); plot(zs, DSkf(zs), col=2, cex=3) # red dots (theory)
lines(zs, kfuncCOP(zs, cop=C1), lwd=4, col=7) # thick yellow line
lines(zs, kfuncCOP(zs, cop=C2), lwd=1, col=1) # thin black line

The plot so produced shows indeed that the numerical operations by kfuncCOP solidly work on these two strictly singular copulas $C_1$ and $C_2$ that are different from the two singular $M$ and $W$ copulas. The $F_K(z)$ curves exactly matching the theoretical curve provided by Durante and Sempi are produced.

CONVERSATIONAL ASIDE—Interestingly, Durante and Sempi (2015, pp. 115–121), like other authors of major works cited herein, do not list a general expression for the $F_K(z)$ as a function of any $C(u,v)$. Those authors develop the idea of Kendall Function well and show results but for the author (Asquith) the jump based of Theorem 3.9.2 to an expression, such as shown above for $F_K(z)$ based on $C(u,t)/\delta u$, to an usable form for any $C(u,v)$ is difficult.

This is a fascinating topic because if the Kendall Function is a reasonably important component of copula theory, then why so much difficulty in finding a canonical reference? For this one piece, whereas so much of copBasic features are quite nomenclaturely clear in say Nelsen (2006) or Joe (2014) but somehow not for the Kendall Function.

Perhaps to the professional mathematicians, the descriptions (nomenclature) used in all but Salvadori et al. (2007) are clear to intended readers. But even Salvadori et al. seemingly show theirs in error—perhaps the author (Asquith) has missed something fundamental, but the validations shown in this documentation prove at least that kfuncCOP does what it is supposed to be doing but perhaps for the wrong reasoning. Lastly, Durante and Sempi have an error in their expression for Kendall Tau as a function of $F_K(z)$ (see tauCOP).

SECONDARY RETURN PERIOD—As for “Kendall Measure” (after the lead of Salvadori et al. [2007, pp. 161–170]), the following code shows for purposes of discussing secondary return period that $F_K(z)$ is correct and once and for all as defined in this documentation $F_K(z) \neq K_C(z)$. The secondary return period ($T_K$) is the expected interarrival time between events exceeding a $T$-year joint probability either from $U$, from $V$, or both. Let us use the 100-year level (primary return period; $T = 100$), the Gumbel–Hougaard copula ($\text{GHcop}$) $F_{\text{GH}(3.055)}(z)$ where the choice of $\Theta = 3.055$ is made to match discussion in copBasic-package and Salvadori et al. (2007, table 3.3, p. 166).

# Gumbel-Hougaard [Kendall Tau=0.67267 (Salvadori et al. [2007])]
Tyear <- 100; ANEP <- 1-1/Tyear; nsim <- 30000; ix <- 1:nsim
1/(1-kfuncCOP(ANEP, cop=GHcop, para=3.055)) # 148.28 years
BarT <- sapply(1:20, function(i) {
  UV <- simCOP(n=nsim, cop=GHcop, para=3.055, graphics=FALSE)
nsim/sum(GHcop(UV$U, UV$V, para=3.055) > ANEP) })
message("# BarBarT="), round(mean(BarT), digits=2), " years and StdDev(BarT)="), round( sd(BarT), digits=2)," years")
# BarBarT=149.61 years and StdDev(BarT)=11.12 years

The mean of 20 repeats of a large sample simulation run for $F_{\text{GH}(\Theta=3.055)}(z=0.99$) demonstrates empirical results that closely approximate theory $149.61 \approx 148.28$, and thus congruence is shown that the definition for $F_K(z)$ must be correct and that for $K_C(z)$ is incorrect. The table 3.3 in Salvadori et al. (2007) lists the secondary return period as 148.3 years, which obviously matches the output of kfuncCOP and empirical results shown.
Some additional details on secondary return period from Salvadori et al. (2007). Letting $t_\star$ be some critical joint exceedance probability level, $\overline{F}_K(z)$ be the complement of $F_K(z)$, those authors (p. 166) name super-critical events having $\overline{F}_K(t_\star) < 1 - t_\star$. Thus, the secondary return period $(\overline{F}_K(t_\star)^{-1} = T_K)$ must be greater than the primary return period $(t_\star^{-1})$, which is the case here ($T_K = 148.3 > T = 100$) (Salvadori et al., 2007, p. 166).

Salvadori et al. (2007) provide considerable discussion of $T_K$ and some highlights are:

- (p. 162) events equally or exceeding probability $F_K(1 - 1/t_\star)$ or having return intervals $\geq T_K$ “represent [a] class of potentially dangerous events, the outliers, and [F_K can be used to] introduce an ad hoc return period for such destructive events.”
- (p. 162) “primary return period $[T]$ …only takes into account the fact that a prescribed critical event is expected to appear once in a given time interval $[T]$ …$\overline{F}_K(t_\star)$ provides the exact probability that a potentially destructive event will happen at any given realization of $Z$ …and $T_K$ gives the expected time for such an outlier to occur.”
- (p. 164) “[T] only predicts that a critical event is expected to appear once in a given time interval …would be more important to be able to calculate (1) the probability that a super-critical [sic.] event will occur at any given realization of $Z$ …and (2) how long it takes, on average, for a super-critical event to appear.”
- (p. 164) “the function $\overline{F}_K$ turns the difficult analysis of bivariate dynamics of $X$ and $Y$ into a simpler one-dimensional problem.”

Author(s)

W.H. Asquith

Source

The comprehensive demonstrations are shown in the Note because of a sign convention and (or) probability convention incompatibility with the equation shown by Salvadori et al. (2007, p. 148). Initial source code development for copBasic was based on an hypothesis that the terms the “Kendall Function” and “Kendall Measure” might somehow have separate meanings—that the author must be blamed for misunderstanding the requisite nomenclature—this is evidently not true.

The $K_C(z)$ as shown herein simply can not reproduce $F_K(z; \Pi) = z - z \log z$ for the $\Pi$ copula unless the “−” sign in the $K_C(z)$ equation is changed to a “+” to become the $F_K(z)$ definition as shown. The detective work needed for a valid function kmeasCOP was further complicated by fact that neither Durante and Sempi (2015), Joe (2014), Nelsen (2006), and others do not actually present a general equation for $F_K(z)$ computation for any $C(u, v)$.

Because of the subtle differences evidently between “Kendall functions” (lower case), an explicit derivation for $F_K(z; \Pi)$ is informative to confirm what is meant by the Kendall Function as defined by $F_K(z)$. Starting with $z = \Pi(u, v) = uv$, then

$$v(z) = t = \Pi^{(-1)}(u, z) = z/u, \text{ and}$$

$$\frac{\delta}{\delta u} \Pi(u, t) = \frac{\delta}{\delta u} uv \rightarrow v = \frac{z}{u},$$

substitution can now proceed:

$$F_K(z; \Pi) = z + \int_{z}^{1} \frac{\delta}{\delta u} \Pi(u, t) \, du = z + \int_{z}^{1} \frac{z}{u} \, du,$$
which simplifies to

\[ F_K(z; \Pi) = z + [z \log(u)] \bigg|_z^1 = z + z[\log(1) - \log(z)] = z - z \log z, \]

which matches the special case shown by Joe (2014) for the independence copula \((\Pi; P)\). It is obvious thus that the “+” is needed in the \(F_K(z)\) definition in order to stay consistent with the basic form and integration limits shown by Salvadori et al. (2007) for \(K_C(z)\).

References


See Also

kfuncCOPinv, tauCOP, derCOP, derCOP2, derCOPinv, derCOPinv2

Examples

```r
## Not run:
# Salvadori et al. (2007, p. 148, fig. 3.5 [right])
zs <- seq(0,1, by=.01)
plot(zs, kmeasCOP(zs, cop=GHcop, para=5), log="y", type="l", lwd=4,
     xlab="Z <= z", ylab="Kendall Function", xlim=c(0,1), ylim=c(0.001,1)) #
## End(Not run)
```

**kfuncCOPinv**

_The Inverse Kendall Function of a Copula_

**Description**

Compute the (numerical) inverse \(F_K^{-1}(z) \equiv z(F_K)\) of the Kendall Function \(F_K(z; C)\) (kfuncCOP) of a copula \(C(u, v)\) given nonexceedance probability \(F_K\). The \(z\) is the joint probability of the random variables \(U\) and \(V\) coupled to each other through the copula \(C(u, v)\) and the nonexceedance probability of the probability \(z\) is \(F_K\)—statements such as “probabilities of probabilities” are rhetorically complex so pursuit of word precision is made herein.
Usage

```r
kfuncCOPinv(f, cop=NULL, para=NULL, ...)  
```

Arguments

- `f` Nonexceedance probability \(0 \leq F_K \leq 1\);
- `cop` A copula function;
- `para` Vector of parameters or other data structure, if needed, to pass to the copula; and
- `...` Additional arguments to pass.

Value

The value(s) for \(z(F_K)\) are returned.

Note

The L-moments of Kendall Functions appear to be unresearched. Therefore, the `kfuncCOPlmom` and `kfuncCOPlmoms` functions were written. These compute L-moments on the CDF \(F_K(z)\) and not the quantile function \(z(F_K)\) and thus are much faster than trying to use `kfuncCOPinv` in the more common definitions of L-moments. A demonstration of the mean (first L-moment) of the Kendall Function numerical computation follows:

```r
# First approach
"afunc" <- function(f) kfuncCOPinv(f, cop=GHcop, para=pi)
integrate(afunc, 0, 1) # 0.4204238 with absolute error < 2.5e-05
# Second approach
kfuncCOPlmom(1, cop=GHcop, para=pi) # 0.4204222
```

where the first approach uses \(z(F_K)\), whereas the second method uses integration for the mean on \(F_K(z)\).

Author(s)

W.H. Asquith

References


See Also

`kfuncCOP`
Examples

```r
# Not run:
Z <- c(0, 0.25, 0.50, 0.75, 1)  # Joint probabilities of a N412cop
kfuncCOPinv(kfuncCOP(Z, cop=N412cop, para=4.3), cop=N412cop, para=4.3)
# [1] 0.0000000 0.2499984 0.5000224 0.7500112 1.0000000
# End(Not run)
```

## Description

The L-moments of the Kendall Function of a Copula

Compute the L-moments of the Kendall Function \(F_K(z; C)\) of a copula \(C(u,v)\) where the \(z\) is the joint probability of the \(C(u,v)\). The Kendall Function is the cumulative distribution function (CDF) of the joint probability \(Z\) of the copula. The expected value of the \(z(F_K)\) (mean, first L-moment \(\lambda_1\)), because \(Z\) has nonzero probability for \(0 \leq Z \leq \infty\), is

\[
E[Z] = \lambda_1 = \int_0^\infty [1 - F_K(t)] \, dt = \int_0^1 [1 - F_K(t)] \, dt,
\]

where for circumstances here \(0 \leq Z \leq 1\). The \(\infty\) is mentioned only because expectations of such CDFs are usually shown using \((0, \infty)\) limits, whereas integration of quantile functions (CDF inverses) use limits \((0, 1)\). Because the support of \(Z\) is \((0, 1)\), like the probability \(F_K\), showing just it \((\infty)\) as the upper limit could be confusing—statements such as “probabilities of probabilities” are rhetorically complex so pursuit of word precision is made herein.

An expression for \(\lambda_r\) for \(r \geq 2\) in terms of the \(F_K(z)\) is

\[
\lambda_r = \frac{1}{r} \sum_{j=0}^{r-2} (-1)^j \binom{r-2}{j} \binom{r}{j+1} \int_0^1 [F_K(t)]^{r-j-1} \times [1 - F_K(t)]^{j+1} \, dt,
\]

where because of these circumstances the limits of integration are \((0, 1)\) and not \((-\infty, \infty)\) as in the usual definition of L-moments in terms of a distribution’s CDF.

The mean, L-scale, coefficient of L-variation \((\tau_2, \text{LCV, L-scale/mean})\), L-skew \((\tau_3, \text{TAU3})\), L-kurtosis \((\tau_4, \text{TAU4})\), and \(\tau_5\) (TAU5) are computed. In usual nomenclature, the L-moments are \(\lambda_1 = \text{mean, } \lambda_2 = \text{L-scale, } \lambda_3 = \text{third L-moment, } \lambda_4 = \text{fourth L-moment, and } \lambda_5 = \text{fifth L-moment}\), whereas the L-moment ratios are \(\tau_2 = \lambda_2/\lambda_1 = \text{coefficient of L-variation, } \tau_3 = \lambda_3/\lambda_2 = \text{L-skew, } \tau_4 = \lambda_4/\lambda_2 = \text{L-kurtosis, and } \tau_5 = \lambda_5/\lambda_2 = \text{not named}\). It is common amongst practitioners to lump the L-moment ratios into the general term “L-moments” and remain inclusive of the L-moment ratios. For example, L-skew then is referred to as the 3rd L-moment when it technically is the 3rd L-moment ratio. There is no first L-moment ratio (meaningless) so results from this function will canonically show a NA in that slot. The coefficient of L-variation is \(\tau_2\) (subscript 2) and not Kendall Tau (\(\tau\)). Sample L-moments are readily computed by several packages in \(R\) (e.g. \texttt{lmomco, lmom, Lmoments, POT}).
Usage

kfuncCOPlmom(r, cop=NULL, para=NULL, ...)  
kfuncCOPlmoms(cop=NULL, para=NULL, nmom=5, begin.mom=1, ...)

Arguments

- `r`: The rth order of a single L-moment to compute;
- `cop`: A copula function;
- `para`: Vector of parameters or other data structure, if needed, to pass to the copula;
- `nmom`: The number of L-moments to compute;
- `begin.mom`: The rth order to begin the sequence lambegr:nmom for L-moment computation. The rarely used argument is means to bypass the computation of the mean if the user has an alternative method for the mean or other central tendency characterization in which case begin.mom = 2; and ...
- `...`: Additional arguments to pass.

Value

An R list is returned by kfuncCOPlmoms and only the scalar value of $\lambda_r$ by kfuncCOPlmom.

- `lamdas`: Vector of the L-moments. First element is $\lambda_1$, second element is $\lambda_2$, and so on;
- `ratios`: Vector of the L-moment ratios. Second element is $\tau$, third element is $\tau_3$ and so on; and
- `source`: An attribute identifying the computational source of the L-moments: “kfuncCOPlmoms”.

Note

The L-moments of Kendall Functions appear to be not yet fully researched. An interesting research direction would be the trajectories of the L-moments or L-moment ratio diagrams for the Kendall Function and the degree to which distinction between copulas becomes evident—such diagrams are in wide-spread use for distinguishing between univariate distributions. It is noted, however, that Kendall Function L-moment ratio diagrams might be of less utility that in the univariate world because different copulas can have the same $F_K(z)$, such as all bivariate extreme value copulas.

```r
Rhos <- seq(0, 0.9, by=0.05)  
L1 <- T2 <- T3 <- T4 <- Thetas <- vector(mode="numeric", length(Rhos))  
for(i in 1:length(Thetas)) {  
  Thetas[i] <- uniroot(function(p)  
    Rhos[i] - rhoCOP(cop=PARETOcop, para=p), c(0,100))$root  
  message("Rho = ", Rhos[i], " and Pareto theta = ",  
    round(Thetas[i], digits=4))  
  lmr <- kfuncCOPlmoms(cop=PARETOcop, para=Thetas[i], nmom=4)  
  L1[i] <- lmr$lambdaas[1]; T2[i] <- lmr$ratios[2]  
}
LMR <- data.frame(Rho=Rhos, Theta=Thetas, L1=L1, T2=T2, T3=T3, T4=T4)
plot(LMR$Rho, LMR$T2, ylim=c(-0.04, 0.5), xlim=c(0,1),
     xlab="Spearman Rho or coefficient of L-variation",
     ylab="L-moment ratio", type="l", col=1) # black
lines(LMR$Rho, LMR$T3, col=2) # red
lines(LMR$Rho, LMR$T4, col=3) # green
lines(LMR$T2, LMR$T3, lty=2, col=4) # dashed blue
lines(LMR$T2, LMR$T4, lty=2, col=5) # dashed cyan
lines(LMR$T3, LMR$T4, lty=2, col=6) # dashed purple

Author(s)
W.H. Asquith

References

See Also
kfuncCOP

Examples
```r
## Not run:
kfuncCOPlmom(1, cop=P) # 0.5 * 0.5 = 0.25 is expected joint prob. of independence
#[1] 0.2499999 (in agreement with theory)
ls.str(kfuncCOPlmoms(cop=GHcop, para=4.21)) # Gumbel-Hougaard copula
# lambdas : num [1:5] 0.440617 0.169085 0.011228 -0.000797 0.000249
# ratios : num [1:5] NA 0.38375 0.0664 -0.00472 0.00147
# source : chr "kfuncCOPlmoms" # e.g. L-skew = 0.0664
## End(Not run)

## Not run:
UV <- simCOP(200, cop=PLcop, para=1/pi, graphics=FALSE)
theta <- PLpar(UV[,1], UV[,2]); zs <- seq(0.01,0.99, by=.01) # for later
# Take the sample estimated parameter and convert to joint probabilities Z
# Convert the Z to the Kendall Function estimates again with the sample parameter
Z <- PLcop(UV[,1], UV[,2], para=theta); KF <- kfuncCOP(Z, cop=PLcop, para=theta)
# Compute L-moments of the "Kendall function" and the sample versions
# and again see that the L-moment are for the distribution of the Z!
KNFlmr <- kfuncCOPlmoms(cop=PLcop, para=theta); SAMlmr <- lmomco::lmoms(Z)
knftxt <- paste0("Kendall L-moments: ",
    paste(round(KNFlmr$lambdas, digits=4), collapse=" ", "))
samtxt <- paste0("Sample L-moments: ",
    paste(round(SAMlmr$lambdas, digits=4), collapse=" ", "))
plot(Z, KF, xlim=c(0,1), ylim=c(0,1), lwd=0.8, col=1,
```

**kullCOP**

Kullback–Leibler Divergence, Jeffrey Divergence, and Kullback–Leibler Sample Size

**Description**

Compute the Kullback–Leibler Divergence, Jeffrey Divergence, and Kullback–Leibler sample size following Joe (2014, pp. 234–237). Consider two densities \( f = c_1(u, v; \Theta_f) \) and \( g = c_2(u, v; \Theta_g) \) for two different bivariate copulas \( C_1(\Theta_1) \) and \( C_2(\Theta_2) \) having respective parameters \( \Theta \), then the Kullback–Leibler Divergence of \( f \) relative to \( g \) is

\[
KL(f | g) = \int \int_{I^2} g \log(g/f) \, du \, dv,
\]

and Kullback–Leibler Divergence of \( g \) relative to \( f \) is

\[
KL(g | f) = \int \int_{I^2} f \log(f/g) \, du \, dv,
\]

where the limits of integration \( I^2 \) theoretically are closed on \([0, 1]^2\) but an open interval \((0, 1)^2\) might be needed for numerical integration. Note that in general \( KL(f | g) \neq KL(g | f) \). The \( KL(f | g) \) is the expected log-likelihood ratios of \( g \) to \( f \) when \( g \) is the true density (Joe, 2014, p. 234), whereas \( KL(g | f) \) is the opposite.

This asymmetry leads to Jeffrey Divergence, which is defined as a symmetrized version of the two Kullback–Leibler Divergences, and is

\[
J(f, g) = KL(f | g) + KL(g | f) = \int \int_{I^2} (g - f) \log(g/f) \, du \, dv.
\]

The variances of the Kullback–Leibler Divergences are defined as

\[
\sigma^2_{KL(f | g)} = \int \int_{I^2} g \, [\log(g/f)]^2 \, du \, dv - [KL(f | g)]^2,
\]

## End(Not run)
and
\[ \sigma_{KL(g|f)}^2 = \int \int f \left[ \log(f/g) \right]^2 \, du \, dv - [KL(g|f)]^2. \]

For comparison of copula families \( f \) and \( g \) and taking an \( \alpha = 0.05 \), the Kullback–Leibler sample size is defined as
\[ n_{fg} = \left[ \Phi^{-1}(1 - \alpha) \times \eta_{KL} \right]^2, \]
where \( \Phi^{-1}(t) \) is the quantile function for the standard normal distribution \( \sim N(0,1) \) for nonexceedance probability \( t \), and \( \eta_{KL} \) is the maximum of
\[ \eta_{KL} = \max \left[ \sigma_{KL(f|g)}/KL(f|g), \sigma_{KL(g|f)}/KL(g|f) \right]. \]

The \( n_{fg} \) gives an indication of the sample size needed to distinguish \( f \) and \( g \) with a probability of at least \( 1 - \alpha = 1 - 0.05 = 0.95 \) or 95 percent.

The \texttt{copBasic} features a naive \textit{Monte Carlo integration} scheme in the primary interface \texttt{kullCOP}, although the function \texttt{kullCOPint} provides for nested numerical integration. This later function is generally fast but suffers too much for general application from integral divergencies issued from the \texttt{integrate()} function in \texttt{R}—this must be judged in the light that the \texttt{copBasic} package focuses only on implementation of the function of the copula itself and numerical estimation of copula density (\texttt{densityCOP}) and not analytical copula densities or hybrid representations thereof. Sufficient “bread crumbs” are left among the code and documentation for users to re-implement if speed is paramount. Numerical comparison to the results of Joe (2014) (see \texttt{Examples}) suggests that the default Monte Carlo sample size should be more than sufficient for general inference with the expense of considerable CPU time; however, a couple of repeated calls of \texttt{kullCOP} would be advised and compute say the mean of the resulting sample sizes.

Usage

\begin{verbatim}
kullCOP(cop1=NULL, cop2=NULL, para1=NULL, para2=NULL, alpha=0.05, 
  del=0, n=1E5, verbose=TRUE, sobol=FALSE, scrambling=0, ...)
kullCOPint(cop1=NULL, cop2=NULL, para1=NULL, para2=NULL, alpha=0.05, 
  del=.Machine$double.eps^0.25, verbose=TRUE, ...)
\end{verbatim}

Arguments

- \texttt{cop1}: A copula function corresponding to copula \( f \) in Joe (2014);
- \texttt{para1}: Vector of parameters or other data structure, if needed, to pass to the copula \( f \);
- \texttt{cop2}: A copula function corresponding to copula \( g \) in Joe (2014);
- \texttt{para2}: Vector of parameters or other data structure, if needed, to pass to the copula \( g \);
- \texttt{alpha}: The \( \alpha \) in the Kullback–Leibler sample size equation;
- \texttt{del}: A small value used to denote the \( lo \) and \( hi \) values of the numerical integration: \( lo = del \) and \( hi = 1 - del \). If \( del = 0 \), then \( lo = 0 \) and \( hi = 1 \), which corresponds to the theoretical limits \( I^2 = [0, 1]^2 \) and are defaulted here to \( [0, 1]^2 \) because the Monte Carlo algorithm is preferred for general application. The end point control, however, is maintained just in case pathological situations should arise;
- \texttt{n}: \texttt{kullCOP} (Monte Carlo integration) only—the Monte Carlo integration simulation size;
verbose A logical trigging a couple of status lines of output through the `message()` function in R;

sobol A logical trigging Sobol sequences for the Monte Carlo integration instead of the bivariate uniform distribution. The Sobol sequences are dependent on the `randtoolbox` package and the `sobol()` function of the `randtoolbox` package, and the Sobol sequences canvas the $I^2$ domain for smaller $n$ values than required if statistical independence is used for the Monte Carlo integration. Note that `randtoolbox` at least at version 2.0.+ has “scrambling” of Sobol sequences temporarily disabled, and hence `scrambling=0` as default for `kullCOP`;

scrambling The argument of the same name for `randtoolbox::sobol`; and

... Additional arguments to pass to the `densityCOP` function.

Value

An R list is returned having the following components:

- **MonteCarlo.sim.size**
  - `kullCOP` (Monte Carlo integration) only—The simulation size for numerical integration;

- **divergences**
  - A vector of the Kullback–Leibler Divergences and their standard deviations: $\text{KL}(f|g), \sigma_{\text{KL}(f|g)}, \text{KL}(g|f)$, and $\sigma_{\text{KL}(g|f)}$, respectively;

- **stdev.divergences**
  - `kullCOP` (Monte Carlo integration) only—The standard deviation of the divergences and the variances;

- **Jeffrey.divergence**
  - Jeffrey Divergence $J(f, g)$;

- **KL.sample.size**
  - Kullback–Leibler sample size $n_{fg}$; and

- **integrations**
  - `kullCOPint` (numerical integration) only—An R list of the outer call of the `integrate()` function for the respective numerical integrals shown in this documentation.

Author(s)

W.H. Asquith

References


See Also

densityCOP, vuongCOP
Examples

# See another demonstration under the Note section of statTn().
## Not run:
# Joe (2014, p. 237, table 5.2)
# Gumbel-Hougaard and Plackett copulas below each have a Kendall Tau of about 0.5, and
# Joe (2014) lists in the table that Jeffrey Divergence is about 0.110 and Kullback-Leibler
# sample size is 133. Joe (2014) does not list the copula parameters just says Tau = 0.5.
# Joe (2014) likely did the numerical integrations using analytical solutions to probability
# densities and not rectangular approximations as in copBasic::densityCOP().
set.seed(1)
KL <- kullCOP(cop1=GHcop, para1=2,
  cop2=PLACKETTcop, para2=11.40484, sobol=FALSE)
message("Jeffery Divergence is ",
  round(KL$Jeffrey.divergence, digits=4),
  " and Kullback-Leibler sample size is ", KL$KL.sample.size, ")
# Jeffery Divergence is 0.1106 and Kullback-Leibler sample size is 137.
set.seed(1)
KL <- kullCOP(cop1=GHcop, para1=2,
  cop2=PLACKETTcop, para2=11.40484, sobol=TRUE)
message("Jeffery Divergence is ",
  round(KL$Jeffrey.divergence, digits=4),
  " and Kullback-Leibler sample size is ", KL$KL.sample.size, ")
# Jeffery Divergence is 0.3062 and Kullback-Leibler sample size is 136.

set.seed(1)
S <- replicate(20, kullCOP(cop1=GHcop, para1=2, cop2=PLACKETTcop, sobol=FALSE, para2=11.40484, verbose=FALSE)$KL.sample.size)
print(as.integer(c(mean(S), sd(S)))) # 132 plus/minus 5
S <- replicate(2, kullCOP(cop1=GHcop, para1=2, cop2=PLACKETTcop, sobol=TRUE, para2=11.40484, verbose=FALSE)$KL.sample.size)
# The two S in the later replication are both the same (136) for a sobol=TRUE
# does not produce variation and this is thought (June 2023) as a result
# of the disabled scrambling in the randtoolbox::sobol() function.
## End(Not run)

## Not run:
# Joe (2014, p. 237, table 5.3)
# Gumbel-Hougaard and Plackett copulas below each have a Spearman Rho of about 0.5, and
# Joe (2014) lists in the table that Jeffrey Divergence is about 0.063 and Kullback-Leibler
# sample size is 210. Joe (2014) does not list the parameters and just says that Rho = 0.5.
# Joe (2014) likely did the numerical integrations using analytical solutions to probability
# densities and not rectangular approximations as in copBasic::densityCOP().
set.seed(1)
KL <- kullCOP(cop1=GHcop, para1=1.541071,
  cop2=PLACKETTcop, para2=5.115658, sobol=FALSE)
message("Jeffery Divergence is ",
  round(KL$Jeffrey.divergence, digits=4),
  " and Kullback-Leibler sample size is ", KL$KL.sample.size, ")
# Jeffery Divergence is 0.0642 and Kullback-Leibler sample size is 213.
set.seed(1)
KL <- kullCOP(cop1=GHcop, para1=1.541071,
  cop2=PLACKETTcop, para2=5.115658, sobol=TRUE)
message("Jeffery Divergence is ",
  round(KL$Jeffrey.divergence, digits=4),
  " and Kullback-Leibler sample size is ", KL$KL.sample.size, ")
```r
# Jeffery Divergence is 0.2001 and Kullback-Leibler sample size is 206.

set.seed(1)
S <- replicate(20, kullCOP(cop1=GHcop, para1=1.541071, cop2=PLACKETTcop,
                          para2=5.115658, verbose=FALSE)$KL.sample.size)
print(as.integer(c(mean(S), sd(S)))) # 220 plus/minus 19
## End(Not run)

## Not run:
# Compare Jeffery Divergence estimates as functions of sample size when computed
# using Sobol sequences or not for Gumbel-Hougaard and Pareto copulas.
GHpar <- PApar <- 2 # Spearman Rho = 0.6822339
Ns <- as.integer(10^c(seq(2.0, 3.5, by=0.01), seq(3.6, 5, by=0.05)))
JDuni <- sapply(1:length(Ns), function(i) {
  kullCOP(cop1=GHcop, para1=GHpar, verbose=FALSE,
           cop2=PAcop, para2=PApar, n=Ns[i],
           sobol=FALSE)$Jeffrey.divergence })
JDsob <- sapply(1:length(Ns), function(i) {
  kullCOP(cop1=GHcop, para1=GHpar, verbose=FALSE,
           cop2=PAcop, para2=PApar, n=Ns[i],
           sobol=TRUE )$Jeffrey.divergence })
plot(Ns, JDuni, type="l", log="x", # black line, notice likely outliers too
     xlab="Simulation Sample Size", ylab="Jeffery Divergence")
lines(Ns, JDsob, col="red") # red line
legend("topright", c("Monte Carlo", "Sobol sequence"),
       lwd=c(1,1), col=c("black", "red"), bty="n")
print( c( mean(JDuni), sd(JDuni) ) ) # [1] 0.05915608 0.01284682
print( c( mean(JDsob), sd(JDsob) ) ) # [1] 0.07274190 0.01838939

# The developer notes that plotting KL.sample.size for sobol=TRUE shows
# what appears to be numerical blow up but the Jeffery Divergence does not.
KUni <- sapply(1:length(Ns), function(i) {
  kullCOP(cop1=GHcop, para1=GHpar, verbose=FALSE,
           cop2=PAcop, para2=PApar, n=Ns[i],
           sobol=FALSE)$KL.sample.size })
KLsob <- sapply(1:length(Ns), function(i) {
  kullCOP(cop1=GHcop, para1=GHpar, verbose=FALSE,
           cop2=PAcop, para2=PApar, n=Ns[i],
           sobol=TRUE )$KL.sample.size })
plot(Ns, KUni, type="l", log="xy", # black line, notice likely outliers too
     xlab="Simulation Sample Size", ylab="Kullback-Leibler Sample Size")
lines(Ns, KLsob, col="red") # red line
nideal <- kullCOPint(cop1=GHcop, para1=GHpar, cop2=PAcop, para2=PApar)$KL.sample.size
abline(h=nideal, col="green", lwd=3) # nideal sample size is about 210
legend("topright", c("Monte Carlo", "Sobol sequence", "Judged ideal sample size"),
       lwd=c(1,1,3), col=c("black", "red", "green"), bty="n")

# Let us attempt a visualization to highlight the differences in the two copula by
# simulation. First, using this n = nideal, being the apparent sample size to distinguish
# generally between the two copula having the same Spearman Rho. Do the segments help
# to visually highlight the differences? Next, ask would one judge the parents in the
# simulation being different knowing same Spearman Rho? (Note that the segments are all
# vertical because the U axis is the simulation and the V axis is the conditional
# probability given the U.)

set.seed(1); UVgh <- simCOP(nideal, GHcop, para=GHpar, graphics=FALSE)
set.seed(1); UVpa <- simCOP(nideal, PAcop, para=PApar, graphics=FALSE)

plot(c(0,1), c(0,1), type="n", xlab="U, nonexceedance probability",
ylab="V, nonexceedance probability")
segments(UVgh[,1], UVgh[,2], x1=UVpa[,1], y1=UVpa[,2])
points(UVgh, col="lightgreen", pch=16, cex=0.8) # dots
points(UVpa, col="darkgreen", pch=1, lwd=0.8) # circles

# Repeat the above n = nideal visualizations but with a change to n = nideal*10, and see
# then that there are visually striking shifts systematically in both both tails but also
# in the U in the interval (0.3, 0.7) belt but to a smaller degree than seen in the tails.
## End(Not run)

lcomCOP

L-comoments and Bivariate L-moments of a Copula

Description

Compute the L-comoments (Serfling and Xiao, 2007; Asquith, 2011) through the bivariate L-moments (ratios) \( \delta_{k;C}^{[12]} \) of a copula \( C(u,v; \Theta) \). The L-comoments include L-correlation (Spearman Rho), L-coskew, and L-cokurtosis. As described by Brahimi et al. (2015), the first four bivariate L-moments \( \delta_{k}^{[12]} \) for random variable \( X^{(1)} \) or \( U \) with respect to (wrt) random variable \( X^{(2)} \) or \( V \) are defined as

\[
\delta_{1;C}^{[12]} = 2 \int \int_{I^2} C(u,v) \, du \, dv - \frac{1}{2},
\]

\[
\delta_{2;C}^{[12]} = \int \int_{I^2} (12v - 6)C(u,v) \, du \, dv - \frac{1}{2},
\]

\[
\delta_{3;C}^{[12]} = \int \int_{I^2} (60v^2 - 60v + 12)C(u,v) \, du \, dv - \frac{1}{2}, \text{ and}
\]

\[
\delta_{k;C}^{[12]} = \int \int_{I^2} (280v^3 - 420v^2 + 180v - 20)C(u,v) \, du \, dv - \frac{1}{2},
\]

where the bivariate L-moments are related to the L-comoment ratios by

\[
6\delta_{k}^{[12]} = \tau_{k+1}^{[12]} \quad \text{and} \quad 6\delta_{k}^{[21]} = \tau_{k+1}^{[21]},
\]

where in otherwords, “the third bivariate L-moment \( \delta_{3}^{[12]} \) is one sixth the L-cokurtosis \( \tau_{4}^{[12]} \).” The first four bivariate L-moments yield the first five L-comoments. The terms and nomenclature are not easy and also the English grammar adjective “ratios” is not always consistent in the literature. The \( \delta_{k;C}^{[12]} \) are ratios. The sample L-comoments are supported by the \texttt{lmomco} package, and in particular for the bivariate case, they are supported by the \texttt{lcomoms2()} function of that package.

Similarly, the \( \delta_{k}^{[21]} \) are computed by switching \( u \rightarrow v \) in the polynomials within the above integrals multiplied to the copula in the system of equations with \( u \). In general, \( \delta_{k}^{[12]} \neq \delta_{k}^{[21]} \) for \( k > 1 \) unless in the case of permutation symmetric (isCOP.permsym) copulas. By theory, \( \delta_{1}^{[12]} = \delta_{1}^{[21]} = \rho_C/6 \) where \( \rho_C \) is the Spearman Rho \texttt{rhoCOP}. 

## End(Not run)
The integral for $\delta_{4,C}^{[12]}$ does not appear in Brahimi et al. (2015) but this and the other forms are verified in the Examples and discussion in Note. The four $k \in (1, 2, 3, 4)$ for $U \text{ wrt } V$ and $V \text{ wrt } U$ comprise a full spectrum of system of seven (not eight) equations. One equation is lost because $\delta_{1}^{[12]} = \delta_{1}^{[21]}$.

Chine and Benatia (2017) describe trimmed L-comoments as the multivariate extensions of the univariate trimmed L-moments (Elamir and Seheult, 2003) that are implemented in lmomco. These are not yet implemented in copBasic.

Usage

```r
lcomCOP(cop=NULL, para=NULL, as.bilmoms=FALSE, orders=2:5, ...)
```

Arguments

- `cop`: A copula function;
- `para`: Vector of parameters or other data structure, if needed, to pass to the copula;
- `as.bilmoms`: A logical to trigger return of the $\delta_k$ and the return vectors will be named differently;
- `orders`: The orders of the L-comoments to return, which is internally adjusted if the argument `as.bilmoms` is set. There is no first order L-comoment and the first index on returned values is set to NA to remain index consistent with the lmomco package. An order greater than 5 is not supported; and
- `...`: Additional arguments to pass to the `densityCOP` function.

Value

An R list of the L-comoments or bivariate L-moments is returned depending on `as.bilmoms` setting.

- `bilmomUV`: The bivariate L-moments $\delta_{k}^{[12]}$ of $U$ with respect to $V$ for $k \in [1, 2, 3, 4]$ if orders is 2:5 and there is no NA index as for the L-comoments;
- `bilmomVU`: The bivariate L-moments $\delta_{k}^{[21]}$ of $V$ with respect to $U$ for $k \in [1, 2, 3, 4]$ if orders is 2:5 and there is no NA index as for the L-comoments;
- `lcomUV`: The L-comoments $\tau_{k}^{[12]}$ of $V$ with respect to $U$ for $k \in [2, 3, 4, 5]$ if orders is 2:5 and index 1 is NA; and
- `lcomVU`: The L-comoments $\tau_{k}^{[21]}$ of $V$ with respect to $U$ for $k \in [2, 3, 4, 5]$ if orders is 2:5 and index 1 is NA.

Note

The documentation here is highly parallel to `bilmoms` for which that function was developed some years before lmomCOP was developed in January 2019. Also, `bilmoms` is based on gridded or Monte Carlo integration, and `bilmoms` is to be considered deprecated. However, it is deliberate that related background and various algorithm testing are still documented in `bilmoms`.

Author(s)

W.H. Asquith
References


See Also

bilmoms, lcomCOPpv, uvlmoms

Examples

```r
## Not run:
para <- list(alpha=0.5, beta=0.93, para1=4.5, cop1=GLcop, cop2=PSP)
copBasic:::lcomCOP(cop=composite2COP, para=para)$lcomUV[3]
# Lcomom:T3[12]= +0.156

copBasic:::lcomCOP(cop=composite2COP, para=para)$lcomVU[3]
# Lcomom:T3[21]= -0.0668

bilmoms(cop=composite2COP, n=10000, para=para, sobol=TRUE)$bilcomoms$T3
# Tau3[12]=+0.1566, Tau3[21]=--0.0655

# The numerical default Monte Carlo integration of bilmoms()
# matches the numerical integration of lcomCOP albeit with a
# substantially slower and less elegant means in bilmoms().
## End(Not run)

## Not run:

# The following Spearman Rho and L-coskew values are predicted for a monitoring
# location of the relation between peak streamflow (V) and time into the
# water year (U) where U and V are "U-statistics."
site_srho <- 0.15536430
site_T3_12 <- 0.03866056
site_T3_21 <- -0.03090144

# Create an objective function for 3D optimization with some explicit intention that
# transforms are used to keep the parameters in acceptable parameter space for the
# alpha [0,1] and beta [0,1] and theta for the Plackett copula and the composite1COP
# used to add two more parameters to the Plackett.
ofunc <- function(par, srho=NA, T3_12=NA, T3_21=NA) {
  alpha <- pnorm(par[1]) # takes -Inf to +Inf ---> 0 to 1 # compositing domain
  beta <- pnorm(par[2]) # takes -Inf to +Inf ---> 0 to 1 # compositing domain
  theta <- exp(par[3]) # takes -Inf to +Inf ---> 0 to +Inf # Plackett domain
```
lcomCOPpv

Simulating the Sample Distribution(s) of L-correlation, L-coskew, and L-cokurtosis for a Copula

Description

EXPERIMENTAL: The function provides two themes of sampling distribution characterization by simulation of the first three L-comoment ratios (L-correlation $\tau_2$, L-coskew $\tau_3$ and L-cokurtosis $\tau_4$) of a copula. Subsequently, the sampling distribution can be used for inference. First, semi-optional Monte Carlo integration estimation of the L-comoments of the parent copula are computed. Second, simulations involving the sample size $n$ presumed the size of the actual sample from which the estimates of the sample L-comoments given as arguments. These simulations result in a report of the L-moments (not L-comoments) of the sampling distribution and these then are used to compute p-values for the L-comoment matrices provided by the user as a function argument.

Usage

lcomCOPpv(n, lcom, cop=NULL, para=NULL, repcoe=5E3, type="gno", mcn=1E4, mcrep=10, usemcmu=FALSE, digits=5, ...)

Arguments

n
The sample size $n$. This argument is semi-optional because $n = 0$ can be given to skip corresponding simulations and the ntable on return will only contain NA; this feature permits rapid extraction of the Ntable and thus the lcom contents are simply not used;

lcom
The sample L-comoments (see below);

cop
A copula function;
para Vector of parameters, if needed, to pass to the copula;
repcoe The replication coefficient \( \phi \) affecting the number of simulations of size \( n \);
type The distribution type used for modeling the distribution of the sampling values. The generalized normal (see distribution type "gno" in package \texttt{lmomco}) accommodates some skewness compared to the symmetry of the normal ("nor") just in case situations arise in which non-ignorable skewness in the sample distribution exists. The distribution abbreviations of package \texttt{lmomco} are recognized for the type argument, but in reality the "nor" and "gno" should be more than sufficient;
mcn The sample size \( N \) passed to the \texttt{bilmoms} function for the Monte Carlo integration. If \( N = 0 \) then the Monte Carlo integration is not used, otherwise the minimum sample size is internally reset to \( N = 4 \) so that first four L-moments are computable;
mcrep The number of replications of the Monte Carlo simulation by \texttt{bilmoms};
usemcmu A logical toggling whether the mean value computed from the replicated Monte Carlo integrations is used instead of the mean values for the small sample simulation for the p-value computations;
digits The number of digits to round numerical entries in the returned tables and can be \texttt{NA} for no rounding; and
... Additional arguments to pass to the \texttt{bilmoms} function or to the copula.

Details

The notation \( r[\ldots] \) refers to two specific types of L-comoment definitions and a blend between the two. The notation \( r[12] \) means that the \( r \)th L-comoment for random variables \( \{X^{(1)}, X^{(2)}\} \) where \( X^{(2)} \) is the sorted variable and \( X^{(1)} \) is shuffled by the sorting index. Conversely, the notation \( r[21] \) means that the \( r \)th L-comoment for random variables \( \{X^{(1)}, X^{(2)}\} \) where \( X^{(1)} \) is the sorted variable and \( X^{(2)} \) is shuffled by the sorting index. The notation \( r[12 : 21] \) means that the average between the \( r[21] \) and \( r[21] \) is computed, which might prove useful in circumstances of known or expected symmetry of the L-comoments.

Continuing, \( \hat{\tau}_{2[12]} \) is the sample L-correlation, \( \hat{\tau}_{3[12]} \) is the sample L-coskew, and \( \hat{\tau}_{4[12]} \) is the sample L-cokurtosis all with respect to the sorting of the second variable. The computation of these L-comoment matrices can be made by functions such as function \texttt{1comoms2()} in the \texttt{lmomco} package. The number of replications for the simulations involving the \( n \) sample size is computed by

\[
m = \phi / \sqrt{n},
\]

where \( \phi \) is the repcoe replication factor or coefficient. If \texttt{usemcmu} is TRUE then \( \texttt{mcn} > 0 \) else \texttt{usemcmu} is reset to FALSE.

Value

An \texttt{R list} is returned.

text A string functioning as a label for the remaining tables;
Ntable Another \texttt{R list} holding tables of the L-moments of the L-comoments derived from Monte Carlo integration for samples of size \( N = \texttt{mcn} \). The simulations are replicated \texttt{mcrep} times; and
Another \texttt{R} list holding tables of the L-moments of the L-comoments derived from the small sample simulations for samples of size $n = n$ as well as the p-values estimated by a generalized normal distribution (see \texttt{lmomco} package documentation) of the L-moments using either the small sample means or the mean of the replicated Monte Carlo integrations as dictated by \texttt{usemcmu}. In all circumstances, however, the results for the small sample simulations are tabulated in \texttt{ntable} only the p-value will be reflective the setting of \texttt{usemcmu}.

**Note**

A significance column for the p-values is added to the right side of the returned \texttt{ntable} and is used to guide the eye in interpretation of results. The significant codes having the following definitions for a two-tailed form:

"_" \> 0.1; ".", 0.1; "\*", 0.05; "\*", 0.01; "***", 0.001

**Author(s)**

W.H. Asquith

**References**


**See Also**

\texttt{lcomCOP, COP, kullCOP, vuongCOP}

**Examples**

# See Note section of \texttt{vuongCOP()} for an extended discussion of copula inference  
## Not run:

```
Tau <- 0.6410811; para <- GHcop(tau=Tau)$para # This Tau is from a situation of  
# two river tributaries. These three L-comoments with univariate L-moments on the  
T2 <- c(1, 0.79908960, 0.79908960, 1) # diagonals are derived from those river  
# tributaries and downstream of the junction data.  
T3 <- c(0, -0.04999318, 0.07689082, 0)  
T4 <- c(0, 0.01773833, 0.04756257, 0) # Is the Ho:GHcop rejectable?  
LCOM <- list(T2=matrix(T2, nrow=2), T3=matrix(T3, nrow=2), T4=matrix(T4, nrow=2))  
set.seed(30312)  
ZZ1 <- lcomCOPpv(75, LCOM, cop=GHcop, para=para, repcoe=2000, usemcmu=FALSE)  
print(ZZ1)  
set.seed(30312)  
ZZ2 <- lcomCOPpv(75, LCOM, cop=GHcop, para=para, repcoe=2000, usemcmu=TRUE)  
print(ZZ2)  
```

# The results here suggest that the GHcop is not rejectable.  
## End(Not run)
lcomoms2.ABcop2parameter

Convert L-comoments to Parameters of Alpha-Beta Compositions of Two One-Parameter Copulas

Description

EXPERIMENTAL—This function converts the L-comoments of a bivariate sample to the four parameters of a composition of two one-parameter copulas. Critical inputs are of course the first three dimensionless L-comoments: L-correlation, L-coskew, and L-cokurtosis. The most complex input is the solutionenvir, which is an environment containing arbitrarily long, but individual tables, of L-comoment and parameter pairings. These pairings could be computed from the examples in simcompositeCOP.

The individual tables are prescanned for potentially acceptable solutions and the absolute additive error of both L-comoments for a given order is controlled by the tNeps arguments. The default values seem acceptable. The purpose of the prescanning is to reduce the computation space from perhaps millions of solutions to a few orders of magnitude. The computation of the solution error can be further controlled by X or u with respect to Y or v using the comptNerrXY arguments, but experiments thus far indicate that the defaults are likely the most desired. A solution “matching” the L-correlation is always sought; thus there is no uset2err argument. The arguments uset3err and uset4err provide some level of granular control on addition error minimization; the defaults seek to “match” L-coskew and ignore L-cokurtosis. The setreturn controls which rank of computed solution is returned; users might want to manually inspect a few of the most favorable solutions, which can be done by the setreturn or inspection of the returned object from the lcomoms2.cop2parameter function. The examples are detailed and self-contained to the copBasic package; curious users are asked to test these.

Usage

lcomoms2.ABcop2parameter(solutionenvir=NULL, T2.12=NULL, T2.21=NULL, T3.12=NULL, T3.21=NULL, T4.12=NULL, T4.21=NULL, t2eps=0.1, t3eps=0.1, t4eps=0.1, compt2erruv=TRUE, compt2errvu=TRUE, compt3erruv=TRUE, compt3errvu=TRUE, compt4erruv=TRUE, compt4errvu=TRUE, uset3err=TRUE, uset4err=FALSE, setreturn=1, maxtokeep=1e5)

Arguments

solutionenvir The environment containing solutions;
T2.12 L-correlation $\tau_{2}^{[12]}$;
T2.21 L-correlation $\tau_{2}^{[21]}$;
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>T3.12</td>
<td>L-coskew $\tau_3^{[12]}$;</td>
</tr>
<tr>
<td>T3.21</td>
<td>L-coskew $\tau_3^{[21]}$;</td>
</tr>
<tr>
<td>T4.12</td>
<td>L-cokurtosis $\tau_4^{[12]}$;</td>
</tr>
<tr>
<td>T4.21</td>
<td>L-cokurtosis $\tau_4^{[21]}$;</td>
</tr>
<tr>
<td>t2eps</td>
<td>An error term in which to pick a potential solution as close enough on preliminary processing for $\tau_2^{[1\leftrightarrow 2]}$;</td>
</tr>
<tr>
<td>t3eps</td>
<td>An error term in which to pick a potential solution as close enough on preliminary processing for $\tau_3^{[1\leftrightarrow 2]}$;</td>
</tr>
<tr>
<td>t4eps</td>
<td>An error term in which to pick a potential solution as close enough on preliminary processing for $\tau_4^{[1\leftrightarrow 2]}$;</td>
</tr>
<tr>
<td>compt2erruv</td>
<td>Compute an L-correlation error using the 1 with respect to 2 (or $u$ wrt $v$);</td>
</tr>
<tr>
<td>compt2errvu</td>
<td>Compute an L-correlation error using the 2 with respect to 1 (or $v$ wrt $u$);</td>
</tr>
<tr>
<td>compt3erruv</td>
<td>Compute an L-coskew error using the 1 with respect to 2 (or $u$ wrt $v$);</td>
</tr>
<tr>
<td>compt3errvu</td>
<td>Compute an L-coskew error using the 2 with respect to 1 (or $v$ wrt $u$);</td>
</tr>
<tr>
<td>compt4erruv</td>
<td>Compute an L-cokurtosis error using the 1 with respect to 2 (or $u$ wrt $v$);</td>
</tr>
<tr>
<td>compt4errvu</td>
<td>Compute an L-cokurtosis error using the 2 with respect to 1 (or $v$ wrt $u$);</td>
</tr>
<tr>
<td>uset3err</td>
<td>Use the L-coskew error in the determination of the solution. The L-correlation error is always used;</td>
</tr>
<tr>
<td>uset4err</td>
<td>Use the L-cokurtosis error in the determination of the solution. The L-correlation error is always used;</td>
</tr>
<tr>
<td>setreturn</td>
<td>Set (index) number of the solution to return. The default of 1 returns the preferred solutions based on the controls for the minimization; and</td>
</tr>
<tr>
<td>maxtokeep</td>
<td>The value presets the number of rows in the solution matrix. This matrix is filled with potential solutions as the various subfiles of the solutionenvir are scanned. The matrix is trimmed of NAs and error trapping is in place for too small values of maxtokeep. The default value appears appropriate for the feeding of massively large simulated parameter spaces.</td>
</tr>
</tbody>
</table>

**Value**

An R data.frame is returned.

**Author(s)**

W.H. Asquith

**References**


See Also

simCOP, simcompositeCOP, composite2COP

Examples

```r
## Not run:
# Build an initial parameter to L-comoment mapping table.
mainpara <- list(cop1=PLACKETTcop, cop2=PLACKETTcop,
                  para1gen=function() { return(10*runif(1, min=-5, max=0)) },
                  para2gen=function() { return(10*runif(1, min=0, max=5)) })
nsim <- 1E4
sample.size.for.estimation <- 1000 # really use vastly larger sample size
PlackettPlackettNP <-
  simcompositeCOP(n=sample.size.for.estimation, nsim=nsim, parent=mainpara)
save(PlackettPlackettNP, file="PlackettPlackettNP.RData", compress="xz")

# Plackett-Plackett composited copula from the copBasic package
# Then create an environment to house the "table."
PlackettPlackett <- new.env()
assign("NeedToCreateForDemo", PlackettPlackettNP, envir=PlackettPlackett)
# Now that the table is assigned into the environment, the parameter
# estimation function can be used. In reality, a much much larger
# solution set is needed, but this effort is experimental.
# Now grab the closest Plackett-Plackett solution having the following six
# arbitrary L-comoments. Then simulate 1000 values and plot them to show
# the underlying bivariate distribution.
PPcop <- lcomoms2.ABcop2parameter(solutionenvir=PlackettPlackett,
                                  T2.12=-0.5059, T2.21=-0.5110,
                                  T3.12= 0.1500, T3.21= 0.1700,
                                  T4.12=-0.0500, T4.21= 0.0329,
                                  uset3err=TRUE, uset4err=TRUE)
# A user in encouraged to inspect the contents of PPcop to "assess" the
# solution by a method of L-comoments, we will now proceed with showing the
# copula via a simulation of the fitted version.
para <- list(cop1=PLACKETTcop, cop2=PLACKETTcop, alpha=PPcop$alpha, beta=PPcop$beta,
             para1=PPcop$Cop1Thetas, para2=PPcop$Cop2Thetas)
D <- simCOP(n=5000, cop=composite2COP, para=para, col=rgb(0,0,0,0.1), pch=16)
# The sample L-comoments of the fitted Plackett-Plackett may be found by
lmomco::lcomoms2(D, nmom=4) # from the lmomco package, and six sample values shown
T2.12 <- -0.5151547; T2.21 <- -0.5139863
T3.12 <- 0.1502336; T3.21 <- 0.1721355
T4.12 <- -0.0326277; T4.21 <- 0.0233979
PPcop <- lcomoms2.ABcop2parameter(solutionenvir=PlackettPlackett,
                                  T2.12=T2.12, T2.21=T2.21,
                                  T3.12=T3.12, T3.21=T3.21,
                                  T4.12=T4.12, T4.21=T4.21, uset4err=TRUE)
para <- list(cop1=PLACKETTcop, cop2=PLACKETTcop, alpha=PPcop$alpha, beta=PPcop$beta,
             para1=PPcop$Cop1Thetas, para2=PPcop$Cop2Thetas)
D <- simCOP(n=5000, cop=composite2COP, para=para, col=rgb(0,0,0,0.1), pch=16)
level.curvesCOP(cop=composite2COP, para=para, delt=.1, ploton=FALSE)
```
Convert L-comoments to Parameters of Alpha-Beta-Kappa-Gamma Compositions of Two One-Parameter Copulas

Description

EXPERIMENTAL—This function converts the L-comoments of a bivariate sample to the four parameters of a composition of two one-parameter copulas. Critical inputs are of course the first three dimensionless L-comoments: L-correlation, L-coskew, and L-cokurtosis. The most complex input is the solutionenvir, which is an environment containing arbitrarily long, but individual tables, of L-comoment and parameter pairings. These pairings could be computed from the examples in simcompositeCOP.

The individual tables are prescanned for potentially acceptable solutions and the absolute additive error of both L-comoments for a given order is controlled by the tNeps arguments. The default values seem acceptable. The purpose of the prescanning is to reduce the computation space from perhaps millions of solutions to a few orders of magnitude. The computation of the solution error can be further controlled by X or u with respect to Y or v using the comptNerrXY arguments, but experiments thus far indicate that the defaults are likely the most desired. A solution “matching” the L-correlation is always sought; thus there is nouset2err argument. The argumentsuset3err anduset4err provide some level of granular control on addition error minimization; the defaults seek to “match” L-coskew and ignore L-cokurtosis. The setreturn controls which rank of computed solution is returned; users might want to manually inspect a few of the most favorable solutions, which can be done by the setreturn or inspection of the returned object from the lcomoms2.ABKGcop2parameter function. The examples are detailed and self-contained to the copBasic package; curious users are asked to test these.

Usage

lcomoms2.ABKGcop2parameter(solutionenvir=NULL,
                            T2.12=NULL, T2.21=NULL,
                            T3.12=NULL, T3.21=NULL,
                            T4.12=NULL, T4.21=NULL,
                            t2eps=0.1, t3eps=0.1, t4eps=0.1,
compt2erruv=TRUE, compt2errvu=TRUE, compt3erruv=TRUE, compt3errvu=TRUE, compt4erruv=TRUE, compt4errvu=TRUE, uset3err=TRUE, uset4err=FALSE, setreturn=1, maxtokeep=1e5)

Arguments

solutionenvir The environment containing solutions;
T2.12 L-correlation $\tau_{2}^{[12]}$;
T2.21 L-correlation $\tau_{2}^{[21]}$;
T3.12 L-coskew $\tau_{3}^{[12]}$;
T3.21 L-coskew $\tau_{3}^{[21]}$;
T4.12 L-cokurtosis $\tau_{4}^{[12]}$;
T4.21 L-cokurtosis $\tau_{4}^{[21]}$;
t2eps An error term in which to pick a potential solution as close enough on preliminary processing for $\tau_{2}^{[1\leftrightarrow 2]}$;
t3eps An error term in which to pick a potential solution as close enough on preliminary processing for $\tau_{3}^{[1\leftrightarrow 2]}$;
t4eps An error term in which to pick a potential solution as close enough on preliminary processing for $\tau_{4}^{[1\leftrightarrow 2]}$;
compt2erruv Compute an L-correlation error using the 1 with respect to 2 (or $u$ wrt $v$);
compt2errvu Compute an L-correlation error using the 2 with respect to 1 (or $v$ wrt $u$);
compt3erruv Compute an L-coskew error using the 1 with respect to 2 (or $u$ wrt $v$);
compt3errvu Compute an L-coskew error using the 2 with respect to 1 (or $v$ wrt $u$);
compt4erruv Compute an L-cokurtosis error using the 1 with respect to 2 (or $u$ wrt $v$);
compt4errvu Compute an L-cokurtosis error using the 2 with respect to 1 (or $v$ wrt $u$);
uset3err Use the L-coskew error in the determination of the solution. The L-correlation error is always used;
uset4err Use the L-cokurtosis error in the determination of the solution. The L-correlation error is always used;
setreturn Set (index) number of the solution to return. The default of 1 returns the preferred solutions based on the controls for the minimization; and
maxtokeep The value presets the number of rows in the solution matrix. This matrix is filled with potential solutions as the various subfiles of the solutionenvir are scanned. The matrix is trimmed of NAs and error trapping is in place for too small values of maxtokeep. The default value appears appropriate for the feeding of massively large simulated parameter spaces.

Value

An R data.frame is returned.
Author(s)

W.H. Asquith

References


See Also

`simCOP`, `simcompositeCOP`, `composite3COP`

Examples

```r
# Not run:
mainpara <- list(cop1=PLACKETTcop, cop2=PLACKETTcop,
                 para1gen=function() { return(10^runif(1, min=-5, max=5)) },
                 para2gen=function() { return(10^runif(1, min=-5, max=5)) })
nsim <- 1E4
sample.size.for.estimation <- 1000
PlackettPlackettABKGtest <-
    simcomposite3COP(n=sample.size.for.estimation, nsim=nsim, parent=mainpara)
save(PlackettPlackettABKGtest,file="PlackettPlackettABKG.RData",compress="xz")

# Plackett-Plackett composited copula from the copBasic package
# Then create an environment to house the "table".
PlackettPlackettABKG <- new.env()
assign("NeedToCreateForDemo", PlackettPlackettABKGtest, envir=PlackettPlackettABKG)

# Now that the table is assigned into the environment, the parameter estimation
# function can be used. In reality a much much larger solution set is needed.
# Assume one had the following six L-moments, extract a possible solution.
PPcop <- lcomoms2.ABKGcop2parameter(solutionenvir=PlackettPlackettABKG,
                   T2.12=-0.5059, T2.21=-0.5110,
                   T3.12= 0.1500, T3.21= 0.1700,
                   T4.12=-0.0500, T4.21= 0.0329,
                   uset3err=TRUE, uset4err=TRUE)

# Now take that solution and setup a parameter object.
para <- list(cop1=PLACKETTcop, cop2=PLACKETTcop,
              alpha=PPcop$alpha, beta=PPcop$beta, kappa=PPcop$kappa, gamma=PPcop$gamma,
              para1=PPcop$Cop1Thetas, para2=PPcop$Cop2Thetas)

# Example Plot Number 1
D <- simCOP(n=2000, cop=composite3COP, para=para, col=rgb(0,0,0,0.1), pch=16)
print(lmomco::lcomoms2(D, nmom=4)) # See the six extracted sample values for this seed.
T2.12 <- -0.4877171; T2.21 <- -0.4907403
T3.12 <- 0.1642508; T3.21 <- 0.1715944
T4.12 <- -0.0560310; T4.21 <- -0.0350028
PPcop <- lcomoms2.ABKGcop2parameter(solutionenvir=PlackettPlackettABKG,
```
level.curvesCOP

**Compute and Plot Level Curves of a Copula V with respect to U**

**Description**

Compute and plot *level curves* or *level sets* of a copula for $V$ with respect to $U$ (Nelsen, 2006, pp. 12–13). The level curves at levels $t \mapsto [0 + \Delta t, 1 - \Delta t, \Delta t]$ are defined for $U \mapsto [0 + \Delta u, 1 - \Delta u, \Delta u]$ by

$$t \mapsto C(u = U, v),$$

and solving for $v$. Plotting is provided by this function because level curves are such an important visual attribute of a copula and highly useful for pedagogic purposes. The above equation is implemented by the *inverse of a copula* using **COPinv**.

**Usage**

```r
level.curvesCOP(cop=NULL, para=NULL, ploton=TRUE, lines=TRUE, plotMW=FALSE, ramp=TRUE, delu=0.001, delt=0.10, getlevel=NULL, ...)
```

**Arguments**

- **cop** A copula function;
- **para** Vector of parameters or other data structure, if needed, to pass to the copula;
- **ploton** A logical to toggle on the plot;
- **lines** A logical to toggle calls to the `lines()` function in R to draw the lines;
plotMW

A logical to toggle to use the abline() function in R to plot cross lines for the M (M) and W (W) copulas;

ramp

A logical to toggle whether the level curves are ramped in thickness according to the probability of the line;

delu

The increment for \(\Delta u\). The default is 1 part in 1,000, which should often provide enough smoothness for many copulas in practice;

delt

The increment \(\Delta t\) for the level curves to plot, defaults to 10-percent intervals. If \(\text{delt}=0.5\), then only the median plus the consequences of a defined getlevel is used. If \(\text{NULL}\), then a sequence of \(t\) values is not made and only getlevel is used (if available);

getlevel

If defined, then it is inserted into the sequence of levels \(t\) and that level \(t = \text{getlevel}\) is returned in an R list data structure. If more than one level is desired, then instead of repeated calls to this function, the joint.curvesCOP function could be considered; and

... Additional arguments to pass to the lines() function in R.

Value

Typically no values are returned because this function is used for its side effects, but the arguments can be such that the \(\{u, v\}\) for \(C(u, v) = t\) are returned within an R list.

Author(s)

W.H. Asquith

References


See Also

COPinv, level.curvesCOP2, level.setCOP, joint.curvesCOP

Examples

```r
## Not run:
level.curvesCOP(cop=M, para=NULL, delt=.02) # Upper bounds copula
## End(Not run)
## Not run:
D <- level.curvesCOP(cop=P, getlevel=0.56)
str(D) # empty
D <- level.curvesCOP(cop=P, getlevel=0.5)
str(D) # contains stuff
D <- level.curvesCOP(cop=PSP, getlevel=0.8)
str(D) # contains stuff
## End(Not run)
```
level.curvesCOP2  \hspace{1.0cm} \textit{Compute and Plot Level Curves of a Copula U with respect to V}

\textbf{Description}

Compute and plot level curves or level sets of a copula for $U$ with respect to $V$ \citep[][pp. 12–13]{nelsen2006}. The level curves at a levels $t \mapsto [0 + \Delta t, 1 - \Delta t, \Delta t]$ are defined for $V \mapsto [0 + \Delta v, 1 - \Delta v, \Delta v]$ by

$$t = C(u, v = V),$$

and solving for $u$. Plotting is provided by this function because level curves are such an important visual attribute of a copula and highly useful for pedagogic purposes. The above equation is implemented by the inverse of a copula using \texttt{COPinv2}.

\textbf{Usage}

\begin{verbatim}
level.curvesCOP2(cop=NULL, para=NULL, ploton=TRUE, lines=TRUE, plotMW=FALSE, ramp=TRUE, delv=0.001, delt=0.10, getlevel=NULL, ...)
\end{verbatim}

\textbf{Arguments}

- \texttt{cop}  \hspace{1.0cm} A copula function;
- \texttt{para}  \hspace{1.0cm} Vector of parameters or other data structure, if needed, to pass to the copula;
- \texttt{ploton}  \hspace{1.0cm} A logical to toggle on the plot;
- \texttt{lines}  \hspace{1.0cm} A logical to toggle calls to the \texttt{lines()} function in R to draw the lines;
- \texttt{plotMW}  \hspace{1.0cm} A logical to toggle to use \texttt{abline()} function in R to plot cross lines for the M (M) and W (W) copulas;
- \texttt{ramp}  \hspace{1.0cm} A logical to toggle whether the level curves are ramped in thickness according to the probability of the line;
- \texttt{delv}  \hspace{1.0cm} The increment of $\Delta v$. The default is 1 part in 1,000, which should often provide enough smoothness for many copulas in practice;
- \texttt{delt}  \hspace{1.0cm} The increment of $\Delta t$ for the level curves to plot, defaults to 10-percent intervals;
- \texttt{getlevel}  \hspace{1.0cm} If defined and level exists upon stepping through using \texttt{delt}, then the level curve at the \texttt{getlevel} is returned in an R list data structure; and
- \texttt{...}  \hspace{1.0cm} Additional arguments to pass to the \texttt{lines()} function in R.

\textbf{Value}

Typically no values are returned because this function is used for its side effects, but the arguments can be such that the $\{u, v\}$ for $C(u, v) = t$ are returned within an R list.

\textbf{Author(s)}

W.H. Asquith
References

See Also
COPinv2, level.curvesCOP, level.setCOP2, joint.curvesCOP2

Examples
## Not run:
level.curvesCOP2(cop=M, para=NULL, delt=.02) # Upper bounds copula
## End(Not run)

level.setCOP
Compute a Level Set of a Copula V with respect to U

Description
Compute a level curve or level set of a copula for V with respect to U (Nelsen, 2006, pp. 12–13). The level curve at level $t$ is defined for $U \mapsto [0 + \Delta u, 1 - \Delta u, \Delta u]$ by

$$t \mapsto C(u=U, v),$$

and solving for $v$. The function is largely a dispatcher to features implemented in level.curvesCOP.

Usage
level.setCOP(cop=NULL, para=NULL, getlevel=NULL, delu=0.001, lines=FALSE, ...)

Arguments
cop A copula function;
para Vector of parameters or other data structure, if needed, to pass to the copula;
getlevel The level set for $t$;
delu The increment for $\Delta u$. The default is 1 part in 1,000, which should often in practice provide enough smoothness for many copulas;
lines A logical that matches the argument of the same name in level.curvesCOP; and
... Additional arguments to pass to the lines() function in R.

Value
The level set for $t = \text{getlevel}$ is returned.

Author(s)
W.H. Asquith
level.setCOP2

Compute a Level Set of a Copula U with respect to V

Description

Compute a *level curve* or *level set* of a copula for \( U \) with respect to \( V \) (Nelsen, 2006, pp. 12–13). The level curve at level \( t \) is defined for \( V \mapsto [0 + \Delta v, 1 - \Delta v, \Delta v] \) by

\[
t \mapsto C(u, v=V),
\]

and solving for \( u \). The function is largely a dispatcher to features of `level.curvesCOP2`.

Usage

`level.setCOP2(cop=NULL, para=NULL, getlevel=NULL, delv=0.001, lines=FALSE, ...)`

Arguments

- `cop` A copula function;
- `para` Vector of parameters or other data structure, if needed, to pass to the copula;
- `getlevel` The level set for \( t \);
- `delv` The increment for \( \Delta v \). The default is 1 part in 1,000, which should often in practice provide enough smoothness for many copulas;
- `lines` A logical that matches the argument of the same name in `level.curvesCOP2` and
- `...` Additional arguments to pass to the `lines()` function in R.

Value

The level set for \( t = \text{getlevel} \) is returned.
Author(s)
W.H. Asquith

References

See Also
level.setCOP, level.curvesCOP2

Examples
## Not run:
set <- level.setCOP2(cop=N4212cop, para=3.1, getlevel=0.23, delu=0.005)
level.curvesCOP2(cop=N4212cop, para=3.1, delv=0.001, delt=0.02)
lines(set$U, set$V, col=2, lwd=2) # manually draw the 23rd percentile
set <- level.setCOP2(cop=N4212cop, para=3.1, getlevel=0.17, lines=TRUE, col=4, lwd=4)
# Notice the change in the lines argument and using levelsetCOP2 to draw.
mtree("Level Curves and Special Level Sets for N4212 copula") #
## End(Not run)

The Fréchet–Hoeffding Upper-Bound Copula

Description
Compute the Fréchet–Hoeffding upper-bound copula (Nelsen, 2006, p. 11), which is defined as

\[ M(u, v) = \min(u, v). \]

This is the copula of perfect association (comonotonicity, perfectly positive dependence) between \( U \) and \( V \) and is sometimes referred to as the comonotonicity copula. Its opposite is the \( W(u, v) \) copula (countermonotonicity copula; \( W \)), and statistical independence is the \( \Pi(u, v) \) copula (\( P \)).

Usage
\[ M(u, v, \ldots) \]

Arguments
\begin{itemize}
  \item \( u \) Nonexceedance probability \( u \) in the \( X \) direction;
  \item \( v \) Nonexceedance probability \( v \) in the \( Y \) direction; and
  \item \( \ldots \) Additional arguments to pass.
\end{itemize}

Value
Value(s) for the copula are returned.
Author(s)

W.H. Asquith

References


See Also

W, P

Examples

M(0.4, 0.6)
M(0, 0)
M(1, 1)

med.regressCOP

Perform Median Regression using a Copula by Numerical Derivative Method for V with respect to U

Description

Perform median regression (Nelsen, 2006, pp. 217–218) of a copula by inversion of numerical derivatives of the copula (derCOPinv). The documentation for qua.regressCOP provides mathematical details. The qua.regressCOP.draw supports so-called quantile regression along various probability levels (see Examples).

Usage

med.regressCOP(u=seq(0.01, 0.99, by=0.01), cop=NULL, para=NULL, level=NA, ...)

Arguments

u Nonexceedance probability u in the X direction;
cop A copula function;
para Vector of parameters or other data structure, if needed, to pass to the copula;
level The level of the prediction interval to compute. For example, level=0.95 will compute the 95-percent prediction interval as will level=0.05 because internally a reflection check is made; and
... Additional arguments to pass such qua.regressCOP and derCOPinv that are called in succession.
Value

An R data.frame of the median regressed probabilities of $V$ and provided $U$ values is returned. Note: if level is used, then the column ordering of the returned data.frame changes—please access the columns by the named idiom. The lower- and upper-prediction interval is contained in the columns respectively titled $V_{lwr}$ and $V_{upr}$ to mimic nomenclature somewhat of function `predict.lm()` in R.

Note

An extended demonstration is needed concerning prediction intervals by median regression and a comparison to well-known linear regression. This also affords and opportunity to have `copBasic` interact with the `copula` package to gain access to the Gaussian copula and a maximum pseudo-likelihood estimation of the parameter.

First, a function `NORMcop()` is defined to form the interconnect between the two packages. It is critically important that the user recognize that the so-called copula object as built by the `copula` package is treated as the canonical para argument in `copBasic` calls herein.

```
"NORMcop" <- # pCoupla() from package copula is analogous to COP()
function(u,v, para=NULL, ...) {
  if(length(u) == 1) u <- rep(u, length(v)) # see asCOP() for reasoning of
  if(length(v) == 1) v <- rep(v, length(u)) # this "vectorization" hack
  return(copula::pCopula(matrix(c(u,v), ncol=2), para))
}
```

The parameter $\Theta \in [-1,1]$ (Pearson $R$) and $\rho_C(\Theta)$ (Spearman $Rho$, rhoCOP) and $\tau_C(\Theta)$ (Kendall $Tau$, tauCOP) are according to Salvadori et al. (2007, p. 255) the values

$$\rho_C(\Theta) = \frac{2}{\pi} \arcsin(\Theta)$$

and

$$\tau_C(\Theta) = \frac{6}{\pi} \arcsin(\Theta/2).$$

Second, a bivariate Gaussian copula is defined with a parameter $\Theta = 0.7$ (thus $\rho_C = 0.6829105$, rhoCOP(NORMcop, norm.cop)) and then $n = 255$ samples simulated from it. These are then cast into standard normal variates to mimic the idea of bivariate data in nonprobability units and facilitation regression comparison.

```
norm.cop <- copula::normalCopula(c(0.7), dim = 2) # define a Gaussian copula
UVs <- copula::rCopula(255, norm.cop) # draw 255 samples
UVs <- as.data.frame(UVs); X <- qnorm(UVs[,1]); Y <- qnorm(UVs[,2])
```

Third, the Weibull plotting positions from the pp() function of package `lmomco` are used to estimate the empirical probabilities of the data in UV that are casted into an R matrix because the `copula` package expects the data as a matrix for the default parameter estimation. The code is completed by the specification of the fitted Gaussian copula in fnorm.cop.

```
UV <- as.matrix(data.frame(U=lmomco::pp(X, sort=FALSE), V=lmomco::pp(Y, sort=FALSE)))
```
para <- copula::fitCopula(copula::normalCopula(dim=2), UV)
fnorm.cop <- copula::normalCopula(para, dim=2)

Fourth, ordinary-least-squares (OLS) linear regression for \( Y \mid X \) and \( X \mid Y \) is computed, and the results plotted on top of the data points. The 2/3rd-prediction limits are computed by `predict.lm()` and also shown.

```r
# Classical linear regressions from two perspectives.
LMyx <- lm(Y~X); LMxy <- lm(X~Y)
YonX <- summary(LMyx); XonY <- summary(LMxy)
QUorV <- seq(-3, 3, by=0.05) # vector for graphical operations
plot(X, Y, col=8, pch=21)
lines(QUorV, YonX$coefficients[1]+YonX$coefficients[2]*QUorV, col=2, lwd=4)
tmp <- predict.lm(LMyx, list(X=X), level=2/3, interval="prediction")
lines(X, tmp[,2], col=2); lines(X, tmp[,3], col="red")
lines(XonY$coefficients[1]+XonY$coefficients[2]*QUorV, QUorV, col=3, lwd=4)
tmp <- predict.lm(LMxy, list(Y=Y), level=2/3, interval="prediction")
lines(tmp[,2], Y, col="green"); lines(tmp[,3], Y, col="green")
```

Finally, the demonstration ends with plotting of the median regression for the Gaussian copula and drawing the regression lines. The two median regression lines are nearly coincident with the OLS regression lines as anticipated with a reasonably large sample size albeit maximum pseudo-likelihood was used to estimate the copula parameter. The mean of a uniform distributed variable given say \( U = u \) (horizontal axis) is 1/2, which coincides with the median. The median regression lines thus are coincident with the OLS lines even though OLS and real-space (native units of \( X \) and \( Y \)) were not used for their computation. The 2/3-prediction intervals also are plotted for comparison.

```r
UorV <- c(0.001, seq(0.02,0.98, by=.02), 0.999)
MEDreg <- med.regressCOP( u=UorV, level=2/3, cop=NORMcop, para=fnorm.cop)
MEDreg2 <- med.regressCOP2(v=UorV, level=2/3, cop=NORMcop, para=fnorm.cop)
lines(qnorm(UorV), qnorm(MEDreg$V), col="blue", lty=2)
lines(qnorm(UorV), qnorm(MEDreg$Vlwr), col="blue", lty=2)
lines(qnorm(UorV), qnorm(MEDreg$Vupr), col="blue", lty=2)
lines(qnorm(MEDreg2$U), qnorm(UorV), col="magenta", lty=2)
lines(qnorm(MEDreg2$Ulwr), qnorm(UorV), col="magenta", lty=2)
lines(qnorm(MEDreg2$Uupr), qnorm(UorV), col="magenta", lty=2)
```

A curious aside (Joe, 2014, p. 164) about the Gaussian copula is that Blomqvist Beta (\( \text{blomCOP} \)) is equal to Kendall Tau (\( \text{tauCOP} \)), which can be checked by

```r
blomCOP(cop=NORMcop, para=norm.cop) # 0.4936334
tauCOP( cop=NORMcop, para=norm.cop) # 0.493633
```

and obviously the \( \beta_C = \tau_C \) are numerically the same.
med.regressCOP2

Perform Median Regression using a Copula by Numerical Derivative Method for U with respect to V

Description

Perform median regression of a copula (Nelsen, 2006, pp. 217–218) by inversion of numerical derivatives of the copula (derCOPinv2). The documentation for qua.regressCOP2 provides mathematical details.

Usage

med.regressCOP2(v=seq(0.01, 0.99, by=0.01), cop=NULL, para=NULL, level=NA, ...)

Author(s)

W.H. Asquith

References


See Also

med.regressCOP2, qua.regressCOP, qua.regressCOP.draw

Examples

## Not run:
# INDEPENDENCE YIELDS STRAIGHT LINES, RED IS THE MEDIAN REGRESSION
FF <- seq(0.1, 0.9, by=0.1)
plot(c(0,1),c(0,1), type="n", lwd=3,
     xlab="U, NONEXCEEDANCE PROBABILITY", ylab="V, NONEXCEEDANCE PROBABILITY")
# Draw the regression of V on U and then U on V (wrtV=TRUE)
qua.regressCOP.draw(f=FF, cop=P, para=NA, ploton=FALSE)
qua.regressCOP.draw(f=FF, cop=P, para=NA, ploton=FALSE, wrtV=TRUE, lty=2)#
## End(Not run)

## Not run:
# NEGATIVE PLACKETT THETA YIELDS CURVES DOWN TO RIGHT, RED IS THE MEDIAN REGRESSION
theta <- 0.5; FF <- seq(0.1, 0.9, by=0.1)
plot(c(0,1),c(0,1), type="n", lwd=3,
     xlab="U, NONEXCEEDANCE PROBABILITY", ylab="V, NONEXCEEDANCE PROBABILITY")
# Draw the regression of V on U and then U on V (wrtV=TRUE)
qua.regressCOP.draw(f=FF, cop=PLACKETTcop, ploton=FALSE, para=theta)
qua.regressCOP.draw(f=FF, cop=PLACKETTcop, ploton=FALSE, para=theta, wrtV=TRUE, lty=2)#
## End(Not run)
Arguments

v  Nonexceedance probability $v$ in the $Y$ direction;
cop  A copula function;
para  Vector of parameters or other data structure, if needed, to pass to the copula;
level  The level of the prediction interval to compute. For example, level=$0.95$ will compute the 95-percent prediction interval as will level=$0.05$ because internally a reflection check is made; and

...  Additional arguments to pass such qua.regressCOP2 and derCOPinv2 that are called in succession.

Value

An R data.frame of the median regressed probabilities of $U$ and provided $V$ values is returned. Note: if level is used, the column ordering of the returned data.frame changes—please access the columns by the named idiom. The lower- and upper-prediction interval bounds are contained in the columns respectively titled $Ulwr$ and $Uupr$ to mimic nomenclature somewhat of function predict.lm() in R.

Author(s)

W.H. Asquith

References


See Also

med.regressCOP, qua.regressCOP2, qua.regressCOP.draw

Examples

# See examples under med.regressCOP

---

**mleCOP**

*Maximum Pseudo-Log-Likelihood Estimation for Copula Parameter Estimation*

Description

Perform maximum pseudo-log-likelihood estimation (pMLE) for copula parameters by maximizing the function:

$$\mathcal{L}(\Theta_p) = \sum_{i=1}^{n} \log \left[ c(F_x(x_i), F_y(y_i); \Theta_p) \right],$$

where $\mathcal{L}(\Theta_p)$ is the log-likelihood for parameter vector $\Theta_p$ of dimension $p$, and $c(u, v; \Theta_p)$ is the bivariate copula density. The $u$ and $v$ are estimated by the respective empirical cumulative distribution functions $u = F_x(\cdots)$ and $v = F_y(\cdots)$ for each of the joint realizations of a sample of size $n$. The $c(u, v)$ is numerically estimated by the copula using the densityCOP function.
Usage

mleCOP(u, v=NULL, cop=NULL, parafn=function(k) return(k),
interval=NULL, init.para=NULL, verbose=FALSE, control=list(),
the.zero=.Machine$double.eps^0.25, s=0, ...)

Arguments

u  Nonexceedance probability \( u \) in the \( X \) direction;

v  Nonexceedance probability \( v \) in the \( Y \) direction and if NULL then \( u \) is treated as a two column R data.frame;

cop  A copula function;

parafn  A function responsible for generating the parameters. This is often just a simple return of a parameter vector as \texttt{copBasic} uses this style of parameterization, but this function can take over parameter remapping to handle boundary conditions to benefit the search or provide an interface into other copula packages in R (see \texttt{Examples});

interval  The search interval for root finding, by \texttt{stats::optimise()}, if the parameter dimension of the copula is \( p = 1 \). The interval is not used for \( p \geq 2 \);

init.para  The initial guesses for the parameters for the \( p \)-dimensional optimization for \( p \geq 2 \). The initial guess is used, by \texttt{stats::optim()}, if the parameter dimension of the copula is \( p = 1 \) and interval is NULL (see \texttt{Examples});

verbose  A logical that internally is converted to integer to trigger 1 (sum of logs of \texttt{densityCOP} shown), 2 (add reporting of the copula parameter on each iteration), or more levels of verbose reporting scheme within the objective function. This is independent from the \texttt{control$trace} of function \texttt{optim}();

control  This argument is the argument of the same name for \texttt{optim}();

the.zero  The value for “the zero” of the copula density function. This argument is the argument of the same name for \texttt{densityCOP}. The default here is intended to suggest that a tiny nonzero value for density will trap the numerical zero densities;

s  A vector of at least two presumably uniformly distributed or regular sequence of nonexceedance probabilities in \( U \) for simulation of \( V \) by \texttt{simCOPv} and plotting of these \( U \) and \( V \). This plotting is only made if the length of \( s \) is nonzero and verbose is greater than or equal to 2. This plotting feature for the \( s \) is pedagogical and intended for demonstration or teaching opportunities. This feature has no utility for the optimization itself; and

...  Additional arguments to pass, see source code for the internally used functions that can pick these additional arguments up.

Value

The value(s) for the estimated parameters are returned within an R list where the elements listed below are populated unique to this package. The other elements of the returned list are generated from either the \texttt{optimise()} (1D, \( p = 1 \)) or \texttt{optim()} (\( pD, p \geq 2 \)) functions of R.
The parameter(s) in a canonical element after the one-dimensional root finding ($p = 1$) or multi-dimensional optimization ($p \geq 2$) solutions are passed through `parafn` so that these are in the parameter units of the copula and not necessarily those transformed for the optimization;

A helpful message unique to the `copBasic` package;

The maximum of the log-likelihood matching the name for the same quantity by the function `fitCopula` in package `copula` though a separate implementation is used in `copBasic`;

Akaike information criterion (AIC) (see also `aicCOP`): $AIC = 2p - 2\mathcal{L}(\Theta_p)$; and

Bayesian information criterion (BIC) (see also `bicCOP`): $BIC = p\log(n) - 2\mathcal{L}(\Theta_p)$.

Note

This section provides for a more thorough assessment of pMLE than shown in the Examples.

INTERFACE TO THE COPULA PACKAGE—A not uncommon question to the author is how can `copBasic` support copulas from other packages? A `copBasic` pMLE implementation to the Gaussian copula from the `copula` package is thus useful for instruction.

Two interface functions are required for the pMLE situation. First, interface the `copula` package in a generic form for the `copBasic` package:

```r
"cB2copula" <- # pCoupla() from package copula is analogous to COP()
function(u,v, para=NULL, ...) {
  if(length(u) == 1) u <- rep(u, length(v)) # see asCOP() for reasoning of
  if(length(v) == 1) v <- rep(v, length(u)) # this "vectorization" hack
  return(copula::pCopula(matrix(c(u,v), ncol=2), para))
}
```

where the `para` argument above must be built by the features of the `copula` package. The following function then provides for parameter setup specific to the Gaussian copula having parameter $\rho$:

```r
copula2cBpara <- function(rho) return(copula::normalCopula(rho, dim = 2))
```

Now, let us perform a parameter estimate for a sample of size $n = 900$:

```r
set.seed(162); UV <- simCOP(n=900, cop=cB2copula, para=copula2cBpara(0.45))
mleCOP(UV, cop=cB2copula, parafn=copula2cBpara, interval=c(-1,1))$para
# rho.1 = 0.4458822
```

The search interval for the Gaussian copula is $\rho \in [-1,1]$, and the final result is $\rho = 0.4458822$.

MULTI-DIMENSIONAL EXAMPLE OF pMLE—Consider a 2-parameter Gumbel–Hougaard copula ($\text{GH}(\Theta_1, \Theta_2)$) but now use the `parafn` argument to provide boundary condition assistance through function `GH2pfunc` to the `optim()` function that performs the maximization.

```r
set.seed(162); UV <- simCOP(n=890, cop=GHcop, para=c(2.4, .06))
GH2pfunc <- function(p) { return(c(exp(p[1])+1, exp(p[2]))) }
ML <- mleCOP(UV$U, UV$V, cop=GHcop, init.para=c(1,1), parafn=GH2pfunc)
print(ML$para) # [1] 2.2755018 0.1194788
```

The search interval for the Gaussian copula is $\rho \in [-1,1]$, and the final result is $\rho = 0.4458822$.
and the result is $\Theta_{1,2} = (2.2755018, 0.1194788)$. Next, consider now a 3-parameter $\text{GH}(\Theta, \pi_1, \pi_2)$ copula and again use the `parafn` argument through function `GH3pfunc` but notice that the 2nd and 3rd parameters are now mapped into $0 \leq \pi_1, \pi_2 \leq 1$ domain using the `pnorm()` function.

```r
set.seed(162); UV <- simCOP(n=500, cop=GHcop, para=c(5.5, .6, .9))
GH3pfunc <- function(p) { return(c(exp(p[1])+1, pnorm(p[2]), pnorm(p[3]))) }
ML <- mleCOP(UV$U, UV$V, cop=GHcop, init.para=c(1, .5, .5), parafn=GH3pfunc)
print(ML$para) # [1] 5.3742229 0.6141652 0.9382638
```

and the result is $\Theta = 5.3742229$ and $\pi_{1,2} = (0.6141652, 0.9382638)$.

**ANOTHER MULTI-DIMENSIONAL EXAMPLE OF pMLE**—Finally, an experiment can be made fitting a 3-parameter $\text{GH}(\Theta, \pi_1, \pi_2)$ to a simulation from a 2-parameter $\text{GH}(\beta_1, \beta_2)$, where the seed is just arbitrary and the Vuong Procedure (`vuongCOP`) is used to compare fits and make inference. The parameter functions `GH2pfunc` and `GH3pfunc` are as before.

```r
set.seed(10); UV <- simCOP(n=500, cop=GHcop, para=c(1.7, 1.86))
GH2pfunc <- function(p) { return(c(exp(p[1])+1, exp(p[2]))) }
GH3pfunc <- function(p) { return(c(exp(p[1])+1, pnorm(p[2]), pnorm(p[3]))) }
para1 <- mleCOP(UV, cop=GHcop, init.para=c(1,1), parafn=GH2pfunc)$para
para2 <- mleCOP(UV, cop=GHcop, init.para=c(1,.5,.5), parafn=GH3pfunc)$para
vuongCOP(UV, cop1=GHcop, para1=para1, cop2=GHcop, para2=para2)$message
#[1] "Copula 1 has better fit than Copula 2 at 100 x (1-alpha) level"
```

The results show the 2-p $\text{GH}$ is a better fit to the simulated data than the 3-p $\text{GH}$, which seems a bit self evident? Plot some same-seeded simulations just to confirm.

```r
set.seed(67) # First the estimated parameters but with the correct model.
UV <- simCOP(n=200, GHcop, para=para1, snv=TRUE, pch=16, col=2)
set.seed(67) # Second, the estimated incorrect model.
UV <- simCOP(n=200, GHcop, para=para2, snv=TRUE, ploton=FALSE)
```

Yes, differences in form are manifest in the produced graphic. Now, let us try another set of parameters and again an arbitrarily-chosen seed.

```r
set.seed(10); UV <- simCOP(n=500, cop=GHcop, para=c(1.91, 0.16))
para1 <- mleCOP(UV, cop=GHcop, init.para=c(1,1), parafn=GH2pfunc)$para
para2 <- mleCOP(UV, cop=GHcop, init.para=c(1,.5,.5), parafn=GH3pfunc)$para
vuongCOP(UV, cop1=GHcop, para1=para1, cop2=GHcop, para2=para2)$message
#[1] "Copulas 1 and 2 are not significantly different at 100 x (1-alpha)"
```

The results show equivalence, let us now check a graphic.

```r
set.seed(67); z <- simCOP(n=200, GHcop, para=para1, snv=TRUE, pch=16, col=2)
set.seed(67); z <- simCOP(n=200, GHcop, para=para2, snv=TRUE, ploton=FALSE)
```

The differences are small but the differences might be inflating into the lower left corner. What sample size could conceivably begin to distinguish between the copula?
The summation yields 6 of 20 for which copula 1 has the better fit, but with \( n = 1,000 \) instead of \( n = 625 \), the sum of the Results is 13 of 20 (so better than half the time). This seems to be in conflict with what the \( n_{fg} \) sample size from \texttt{kullCOP} should be telling. The author thinks it should be 18 to 19 of 20 (95th percentile) based on what the \texttt{kullCOP} is reported to do (NEED TO LOOK INTO THIS).

Author(s)

W.H. Asquith

References


See Also

densityCOP

Examples

# See also extended code listings and discussion in the Note section

## Not run:
# Here, we study the trajectory of the objective function in a simple
# 1-dimensional optimization. See how we must provide the interval.
set.seed(162); UV <- simCOP(n=188, cop=PLcop, para=5.6)
ML <- mleCOP(UV$U, UV$V, cop=PLcop, interval=c(0.1, 40)) # 5.225459 estimated
Thetas <- 10^((seq(log10(0.001), log10(100), by=0.005))
MLs <- sapply(Thetas, function(k)
    densityCOP(UV$U, UV$V, cop=PLcop, para=k, sumlogs=TRUE))
plot(Thetas, MLs, log="x", type="l", # draw the pMLE solution process
    xlab="Plackett Theta", ylab="sum of log densities")
lines(rep(ML$para, 2), c(ML$objective, par()$usr[3]), col="red")
points(ML$para, ML$objective, pch=16, col="red")
## End(Not run)

## Not run:
# Here, we study again 1-dimensional optimization but use the
# multidimensional version with an alert issued.
```r
set.seed(149); UV <- simCOP(1000, cop=CLcop, para=pi)
# Warning messages about using optim() for 1D solution
mleCOP(UV, cop=CLcop, init.para=2)$para # 3.082031
# No warning message, optimise() called instead.
mleCOP(UV, cop=CLcop, interval=c(0,1E2))$para # 3.081699
## End(Not run)

## Not run:
# Here, we evaluate a 2-dimensional problem using a Plackett again but with
# the addition of asymmetry towards high V outliers from the Plackett cloud.
# This example also adds the internal verbose and graphic diagnostics for
# the iterations of the optimizer. Here, we learn that we need on a time have
# some idea where the solution might lay so that we can provide a suitable
# set of initial parameters for the algorithm.
para <- list(beta=-0.1, cop=PLcop, para=1000)
UV <- simCOP(2000, cop=breveCOP, para=para); abline(0, 1, col="red", lwd=3)
PL2pfunc <- function(p) { # see here example of parameter transform
  list(beta=2*pnorm(p[1])-1, para=exp(p[2]), cop=PLcop) # [-1,+1], >=0
}
init.para <- c(0.2535, log(0.02)) # These will not find a solution with this
# guess of negative association, but the next works by using an external
# estimate of the Plackett parameters and here we test with a positive
# skewness (beta for breveCOP > 0) although we know the parent had negative.
init.para <- c(0.2535, log(PLACKETTpar(UV$U, UV$V, byrho=TRUE))) # beta=0.200
rt <- mleCOP(u=UV$U, v=UV$V, init.para=init.para, cop=breveCOP,
  parafn=PL2pfunc, verbose=2, s=seq(0,1, by=0.005)) #
## End(Not run)

N4212cop

The Copula of Equation 4.2.12 of Nelsen’s Book

Description

The \textit{N4212 copula} (Nelsen, 2006, p. 91; eq. 4.2.12) is named by the author (Asquith) for the \texttt{copBasic} package and is defined as

\[
C_{N4212}(u,v; \Theta) = \left(1 + \left[(u^{-1} - 1)^{\Theta} + (v^{-1} - 1)^{\Theta}\right]^{1/\Theta}\right)^{-1}.
\]

The \textit{N4212}(u, v) copula is \textit{not comprehensive} because for \(\Theta = 1\) the copula becomes the so-called \textit{PSP}(u, v) copula (see \textit{PSP}) and as \(\Theta \to \infty\) the copula becomes \textit{M}(u, v) (see \textit{M}). The copula is undefined for \(\Theta < 1\). The \textit{N4212} copula has respective \textit{lower-} and \textit{upper-tail dependency} (see \textit{taildepCOP}).

Although \texttt{copBasic} is intended to not implement or “store house” the enormous suite of copula functions available in the literature, the \textit{N4212} copula is included to give the package another copula to test or test in numerical examples.

Usage

\texttt{N4212cop(u, v, para=NULL, infis=100, ...)}
```
Arguments

- **u**: Nonexceedance probability $u$ in the $X$ direction;
- **v**: Nonexceedance probability $v$ in the $Y$ direction;
- **para**: A vector (single element) of parameters—the $\Theta$ parameter of the copula;
- **infs** (with a spelling error): What is infinity? Testing shows that $\text{infs} = \Theta > 100$ is about right to consider the copula as becoming $M(u,v)$ (see M); and
- ... Additional arguments to pass.

Value

Value(s) for the copula are returned.

Author(s)

W.H. Asquith

References


Examples

```r
N4212cop(0.4,0.6, para=1) == PSP(0.4,0.6) # TRUE
N4212cop(0.4,0.6, para=10) # 0.3999928
taildepCOP(cop=N4212cop, para=10) # LamL = 0.93303; LamU = 0.92823
## Not run:
D <- simCOP(n=400, cop=N4212cop, para=2)
D <- simCOP(n=400, cop=N4212cop, para=10, ploton=FALSE, col=2)
D <- simCOP(n=400, cop=N4212cop, para=100, ploton=FALSE, col=3)#
## End(Not run)
```

---

**P**

The Product (Independence) Copula

Description

Compute the *product copula* (Nelsen, 2006, p. 12), which is defined as

$$\Pi(u,v) = uv.$$

This is the copula of statistical independence between $U$ and $V$ and is sometimes referred to as the *independence copula*. The two extreme antithesis copulas are the Fréchet–Hoeffding upper-bound ($M$) and Fréchet–Hoeffding lower-bound ($W$) copulas.

Usage

```r
P(u, v, ...)
```
Arguments

u Nonexceedance probability $u$ in the $X$ direction;

v Nonexceedance probability $v$ in the $Y$ direction; and

... Additional arguments to pass.

Value

Value(s) for the copula are returned.

Author(s)

W.H. Asquith

References


See Also

M, W, rhoCOP

Examples

\[
P(c(0.4, 0, 1), c(0, 0.6, 1))
\]

## Not run:
n <- 100000 # giant sample size, L-comoments are zero
# PERFECT INDEPENDENCE
UV <- simCOP(n=n, cop=P, graphics=FALSE)
lmomco::lcomoms2(UV, nmom=4)
# The following are $T_{12}$ and $T_{21}$
# L-corr: 0.00265 and 0.00264 ---> ZERO
# L-coskew: -0.00121 and 0.00359 ---> ZERO
# L-cokurtosis: 0.00123 and 0.00262 ---> ZERO

# MODEST POSITIVE CORRELATION
rho <- 0.6; # Spearman Rho
theta <- PLACKETTpar(rho=rho) # Theta = 5.115658
UV <- simCOP(n=n, cop=PLACKETTcop, para=theta, graphics=FALSE)
lmomco::lcomoms2(UV, nmom=4)
# The following are $T_{12}$ and $T_{21}$
# L-corr 0.50136 and 0.50138 ---> Pearson R == Spearman Rho
# L-coskews -0.00641 and -0.00347 ---> ZERO
# L-cokurtosis -0.00153 and 0.00046 ---> ZERO
## End(Not run)
Description

The Pareto copula (Nelsen, 2006, pp. 33) is

\[ C_{\Theta}(u, v) = PA(u, v) = \left[ (1 - u)^{-\Theta} + (1 - v)^{-\Theta} \right]^{-1/\Theta} , \]

where \( \Theta \in [0, \infty) \). As \( \Theta \to 0^+ \), the copula limits to the \( \Pi \) copula (**) and the \( M \) copula (**). The parameterization here has association increasing with increasing \( \Theta \), which differs from Nelsen (2006), and also the Pareto copula is formed with right-tail increasing reflection of the Nelsen (2006) presentation because it is anticipated that copBasic users are more likely to have right-tail dependency situations (say large maxima [right tail] coupling in earth-system data but not small maxima [left tail] coupling).

Usage

PARETOcop(u, v, para=NULL, ...)
PAcop(u, v, para=NULL, ...)

Arguments

- \( u \) Nonexceedance probability \( u \) in the \( X \) direction;
- \( v \) Nonexceedance probability \( v \) in the \( Y \) direction;
- \( \text{para} \) A vector (single element) of parameters—the \( \Theta \) parameter of the copula; and
- \( \ldots \) Additional arguments to pass.

Value

Value(s) for the copula are returned.

Note

The Pareto copula is used in a demonstration of Kendall Function L-moment ratio diagram construction (see kfuncCOPlmoms).

Author(s)

W.H. Asquith

References


See Also

M, P
Examples

```
z <- seq(0.01,0.99, by=0.01) # Both copulas have Kendall Tau = 1/3
plot( z, kfuncCOP(z, cop=PAcop, para=1), lwd=2,
       xlab="z <= Z", ylab="F_K(z)", type="l")
lines(z, kfuncCOP(z, cop=GHcop, para=1.5), lwd=2, col=2) # red line
# All extreme value copulas have the same Kendall Function [F_K(z)], the
# Gumbel-Hougaard is such a copula and the F_K(z) for the Pareto does not
# plot on top and thus is not an extreme value but shares a "closeness."
```

### Not run:

```
# # End(Not run)
```

---

**The Plackett Copula**

**Description**

The Plackett copula (Nelsen, 2006, pp. 89–92) is

\[
C_{\Theta}(u,v) = PL(u,v) = \frac{1 + (\Theta - 1)(u + v)}{2(\Theta - 1)} - \sqrt{1 + (\Theta - 1)(u + v)^2 - 4uv(\Theta - 1)}.
\]

The Plackett copula \(PL(u,v)\) is comprehensive because as \(\Theta \to 0\) the copula becomes \(W(u,v)\) (see \(W\), countermonotonicity), as \(\Theta \to \infty\) the copula becomes \(M(u,v)\) (see \(M\), comonotonicity) and for \(\Theta = 1\) the copula is \(\Pi(u,v)\) (see \(P\), independence).

Nelsen (2006, p. 90) shows that

\[
\Theta = \frac{H(x,y)[1 - F(x) - G(y) + H(x,y)]}{[F(x) - H(x,y)][G(y) - H(x,y)]},
\]

where \(F(x)\) and \(G(y)\) are cumulative distribution function for random variables \(X\) and \(Y\), respectively, and \(H(x,y)\) is the joint distribution function. Only Plackett copulas have a constant \(\Theta\) for any pair \(\{x,y\}\). Hence, Plackett copulas are also known as constant global cross ratio or contingency-type distributions. The copula therefore is intimately tied to contingency tables and in particular the bivariate Plackett defined herein is tied to a \(2 \times 2\) contingency table. Consider the \(2 \times 2\) contingency table shown at the end of this section, then \(\Theta\) is defined as

\[
\Theta = \frac{a/c}{b/d} = \frac{a}{b/c} / \frac{a/c}{a/d} \quad \text{and} \quad \Theta = \frac{a/b}{c/d} = \frac{a}{a+b} / \frac{b}{a+b},
\]

where it is obvious that \(\Theta = ad/bc\) and \(a, b, c,\) and \(d\) can be replaced by proportions for a sample of size \(n\) by \(a/n, b/n, c/n,\) and \(d/n,\) respectively. Finally, this copula has been widely used in modeling and as an alternative to bivariate distributions and has respective lower- and upper-tail dependency parameters of \(\lambda_L = 0\) and \(\lambda_U = 0\) (taildepCOP).

<table>
<thead>
<tr>
<th></th>
<th>Low</th>
<th>High</th>
<th>Sums</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Low</strong></td>
<td>(a)</td>
<td>(b)</td>
<td>(a + b)</td>
</tr>
<tr>
<td><strong>High</strong></td>
<td>(c)</td>
<td>(d)</td>
<td>(c + d)</td>
</tr>
<tr>
<td><strong>Sums</strong></td>
<td>(a + c)</td>
<td>(b + d)</td>
<td>---</td>
</tr>
</tbody>
</table>
PLACKETTcop

Usage

PLACKETTcop(u, v, para=NULL, ...)  
PLcop(u, v, para=NULL, ...)

Arguments

u       Nonexceedance probability \( u \) in the \( X \) direction;  
v       Nonexceedance probability \( v \) in the \( Y \) direction;  
para    A vector (single element) of parameters—the \( \Theta \) parameter of the copula; and  
...     Additional arguments to pass.

Value

Value(s) for the copula are returned.

Note

The Plackett copula was the first (2008) copula implemented in \texttt{copBasic} as part of initial development of the code base for instructional purposes. Thus, this particular copula has a separate parameter estimation function in \texttt{PLACKETTpar} as a historical vestige of a class project.

Author(s)

W.H. Asquith

References


See Also

\texttt{PLACKETTpar, PLpar, PLACKETTsim, M, densityCOP}

Examples

PLACKETTcop(0.4, 0.6, para=1)  
P(0.4, 0.6) # independence copula, same two values because Theta == 1  
PLcop(0.4, 0.6, para=10.25) # joint probability through positive association

## Not run:  
# Joe (2014, p. 164) shows the closed form copula density of the Plackett.  
"dPLACKETTcop" <- function(u,v,para) {  
  eta <- para - 1; A <- para*(1 + eta*(u+v-2*u*v))  
  B <- (((1 + eta*(u+v))^2 - 4*para*eta*u*v)^(3/2); return(A/B)  
}  
u <- 0.08; v <- 0.67 # Two probabilities to make numerical evaluations.  
del <- 0.0001 # a 'small' differential value of probability  
u1 <- u; u2 <- u+del; v1 <- v; v2 <- v+del
PLACKETTpar

Estimate the Parameter of the Plackett Copula

Description

The parameter $\Theta$ of the Plackett copula (Nelsen, 2006, pp. 89–92) (PLACKETTcop or PLcop) is related to the Spearman Rho ($\rho_S \neq 1$, see rhoCOP)

$$\rho_S(\Theta) = \frac{\Theta + 1}{\Theta - 1} - \frac{2\Theta \log(\Theta)}{(\Theta - 1)^2}.$$  

Alternatively, the parameter can be estimated using a median-split estimator, which is best shown as an algorithm. First, compute the two medians:

$$\text{medx} \leftarrow \text{median}(x); \text{medy} \leftarrow \text{median}(y)$$

Second and third, compute the number of occurrences where both values are less than their medians and express that as a probability:

$$k \leftarrow \text{length}(x[x < \text{medx} \& y < \text{medy}]); m \leftarrow k / \text{length}(x)$$

Finally, the median-split estimator of $\Theta$ is computed by

$$\Theta = \frac{4m^2}{(1 - 2m)^2}.$$  

Nelsen (2006, p. 92) and Salvadori et al. (2007, p. 247) provide further details. The input values $x$ and $y$ are not used for the median-split estimator if Spearman Rho (see rhoCOP) is provided by rho.
Usage

PLACKETTpar(x, y, rho=NULL, byrho=FALSE, cor=NULL, ...)  
PLpar(x, y, rho=NULL, byrho=FALSE, cor=NULL, ...)

Arguments

x  Vector of values for random variable X;
y  Vector of values for random variable Y;
rho  Spearman Rho and byrho is set to TRUE automatically;
byrho  Should Spearman Rho be used instead of the median-split estimator;
cor  A copBasic syntax for “the correlation coefficient” suitable for the copula—a synonym for rho; and
...  Additional arguments to pass.

Value

A value for the Plackett copula Θ is returned.

Note

Evidently there “does not appear to be a closed form for τ(Θ)” (Fredricks and Nelsen, 2007, p. 2147), but given ρ(Θ), the equivalent τ(Θ) can be computed by either the tauCOP function or by approximation. One of the Examples sweeps through $\rho \mapsto [0, 1; \Delta \rho=\delta]$, fits the Plackett $\theta(\rho)$, and then solves for Kendall Tau $\tau(\theta)$ using tauCOP. A polynomial is then fit between $|\rho|$ and $\tau$, where the residual standard error is 0.0005705, adjusted R-squared is $\approx 1$, the maximum residual is $\epsilon < 0.006$. Because of symmetry, it is only necessary to fit positive association and reflect the result by the sign of $\rho$. This polynomial is from the Examples is

\[
\text{rho} \leftarrow 0.920698 \\
\text{"getPLACKETTtau"} \leftarrow \text{function(rho) \{} \\
\text{taupoly} \leftarrow 0.6229945*\text{abs(rho)} + 1.1621854*\text{abs(rho)}^2 - 10.7424188*\text{abs(rho)}^3 + 48.9687845*\text{abs(rho)}^4 - 119.0640264*\text{abs(rho)}^5 + 160.0438496*\text{abs(rho)}^6 - 111.8403591*\text{abs(rho)}^7 + 31.8054602*\text{abs(rho)}^8 \\
\text{return(sign(rho)*taupoly)} \\
\} \\
\text{getPLACKETTtau(rho) \# 0.7777726}
\]

The following code might be useful in some applications for the inversion of the polynomial for the $\rho$ as a function of $\tau$:

\[
\text{"fun"} \leftarrow \text{function(rho, tau=NULL) \{ tp \leftarrow \text{getPLACKETTtau(rho); return(tau-tp)} \} \\
\text{tau} \leftarrow 0.78 \\
\text{rho} \leftarrow \text{uniroot(fun, interval=c(0, 1), tau=tau)$root \# 0.9220636}
\]
PLACKETTsim

Author(s)
W.H. Asquith

References


See Also
PLACKETTcop, PLcop, PLACKETTsim, rhoCOP

Examples

## Not run:
Q1 <- rnorm(1000); Q2 <- Q1 + rnorm(1000)
PLpar(Q1, Q2); PLpar(Q1, Q2, byrho=TRUE) # two estimates for same data
PLpar(rho= 0.76) # positive association
PLpar(rho=-0.76) # negative association
tauCOP(cop=PLcop, para=PLpar(rho=-0.15, by.rho=TRUE)) #
## End(Not run)

## Not run:
RHOS <- seq(0, 0.990, by=0.002); TAUS <- rep(NA, length(RHOS))
for(i in 1:length(RHOS)) {
  #message("Spearman Rho: ", RHOS[i])
  theta <- PLACKETTpar(rho=RHOS[i], by.rho=TRUE); tau <- NA
  try(tau <- tauCOP(cop=PLACKETTcop, para=theta), silent=TRUE)
  TAUS[i] <- ifelse(is.null(tau), NA, tau)
}
LM <- lm(TAUS~ RHOS + I(RHOS^2) + I(RHOS^3) + I(RHOS^4) +
         I(RHOS^5) + I(RHOS^6) + I(RHOS^7) + I(RHOS^8) - 1)
plot(RHOS,TAUS, type="l", xlab="abs(Spearman Rho)", ylab="abs(Kendall Tau)"
lines(RHOS,fitted.values(LM), col=3)
## End(Not run)
Description

Simulation of the Plackett copula (Nelsen, 2006, pp. 89–92) is made by this function using analytical formula (Durante, 2007, p. 247; see source code). Thus the PLACKETTsim function can be useful for comparison against the numerical derivative (conditional distribution method) methods (simCOP, simCOPmicro) otherwise used in copBasic. The documentation for PLACKETTcop provides the mathematical formula of the Plackett copula.

Usage

PLACKETTsim(n, para=NULL, ...)

Arguments

n Sample size;
para The \( \Theta \) parameter of the Plackett copula; and
... Additional arguments to pass.

Value

An \texttt{R} data.frame of the values \( U \) and \( V \) for the nonexceedance probabilities is returned.

Author(s)

W.H. Asquith

References


See Also

PLACKETTcop, PLACKETTpar

Examples

PLACKETTsim(10, para=1) # simulate P (independence) copula through a Plackett
PLACKETTsim(10, para=20.3) # simulate strong positive Plackett
The Product of Two Copulas

Description

Perform copula multiplication (so-called “∗-product” or Markov Product) is a continuous analog of matrix multiplication and yields another copula:

\[(C_1 \ast C_2)(u,v) = C_3(u,v) = \int_I \frac{\delta C_1(u,t)}{\delta u} \frac{\delta C_2(t,v)}{\delta u} \, dt,\]

for copulas \(C_1(u,v)\) and \(C_2(u,v)\) are copulas whose ∗-product yields copula \(C_3(u,v)\) in terms of partial derivatives (\(\text{derCOP}\) and \(\text{derCOP2}\)) of the other two. Nelsen (2006, p. 245) lists several identities of the ∗-product involving the product (\(\Pi; P\)), lower bound (\(\text{W}; W\)), and upper bound (\(\text{M}; M\)) copulas:

\[\Pi \ast C = C \ast \Pi = \Pi,\]
\[M \ast C = C \ast M = M,\]
\[(W \ast C)(u,v) = v - C(1 - u, v)\text{ and } (C \ast W)(u,v) = u - C(u, 1 - v),\text{ and}\]

\[W \ast W = M \text{ and } W \ast C \ast W = \hat{C},\]

where \(\hat{C}\) is the survival copula (\(\text{surCOP}\)). The ∗-product is associative:

\[A \ast (B \ast C) = (A \ast B) \ast C,\]

but ∗-product is not commutative (order independent). Nelsen (2006, p. 245) reports that “if we view ∗ as a binary operation on the set of copulas, then \(\Pi\) is the null element, and \(M\) is the identity.”

Copula multiplication is closely linked to Markov Processes (Nelsen, 2006, pp. 244–248).

For other descriptions and computations of copula combination are possible using the \texttt{copBasic} package, see \texttt{convexCOP}, \texttt{convex2COP}, \texttt{composite1COP}, \texttt{composite2COP}, \texttt{composite3COP}, \texttt{glueCOP}, and \texttt{convexCOP}.

Usage

\[
\text{prod2COP}(u,v, \text{cop1=NULL, para1=NULL, cop2=NULL, para2=NULL, para=NULL, interval=NULL, ...})
\]

Arguments

\[
\begin{align*}
u & \quad \text{Nonexceedance probability } u \text{ in the } X \text{ direction;} \\
v & \quad \text{Nonexceedance probability } v \text{ in the } Y \text{ direction;} \\
cop1 & \quad \text{The } C_1(u,v; \Theta_1) \text{ copula function with vectorization as in } \text{asCOP;} \\
\text{para1} & \quad \text{Vector of parameters or other data structures for } \Theta_1, \text{ if needed, to pass to copula } C_1(u,v; \Theta_1); \\
cop2 & \quad \text{The } C_2(u,v; \Theta_2) \text{ copula function with vectorization as in } \text{asCOP;}
\end{align*}
\]
prod2COP

para2  Vector of parameters or other data structures for $\Theta_2$, if needed, to pass to copula $C_2(u,v;\Theta_2)$;

para   An R list that can take the place of the cop1, para1, cop2, and para2 arguments. These four will be populated from same named elements of the list, and if the other four arguments were specified through the function interface, these are silently ignored;

interval An optional interval for the above integral. The default is $I = [0,1]$ but the option of the user to replace exact end points with “small” numbers is possible (e.g. interval=c(lo, 1-lo) for say lo=.Machine$double.eps); and

... Additional arguments to pass to the copulas.

Value

Value(s) for the copula are returned.

Note

The Farlie–Gumbel–Morgenstern copula (FGM$(u,v;\Theta)$; FGMcop) is

$$FGM(u,v;\Theta) = uv[1 + \Theta(1-u)(1-v)],$$

where $-1 \leq \Theta \leq 1$. Nelsen (2006, exer. 6.12, p. 249) asserts that for FGM$(\Theta = \alpha)$ and FGM$(\Theta = \beta)$ with *-product as FGM$_\alpha \ast$FGM$_\beta$ that a closed-form solution exists and is

$$FGM_\alpha \ast FGM_\beta = FGM_{\alpha\beta}/3.$$

This assertion is numerically true as readily verified using the prod2COP function:

```r
u <- c(0.41, 0.87); v <- c(0.13,0.35); A <- -0.532; B <- 0.235
FGMcop( u,v, para= A*B / 3) # 0.0521599 and 0.3034277, matched by prod2COP()
prod2COP(u,v, cop1=FGMcop, para1=A, cop2=FGMcop, para2=B)
```

Author(s)

W.H. Asquith

References


See Also

COP, composite1COP, composite2COP, composite3COP, convexCOP, convex2COP, glueCOP
Examples

```r
## Not run:
# Product P * N4212 ---> P (by identity)
u <- c(0.41, 0.87); v <- c(0.13, 0.35)
prod2COP(u, v, cop1=P, cop2=N4212cop, para1=NA, para2=2.12) # 0.0533 and 0.3045
COP(u, v, cop=P) # 0.0533 and 0.3045
## End(Not run)

## Not run:
para <- list(cop1=PLcop, para1=0.19, cop2=PLcop, para2=34.5)
UV <- simCOP(n=1000, cop=prod2COP, para=para)
# This is large simulation run (with a lot of numerical operations) is expected
# at least for the Placketts and chosen parameters to trigger one or more NAs
# from derCOPinv(). The simCOP() function simply continues on with ignoring the
# solution or lack thereof for certain combinations, and simCOP() will report how
# many of the simulated values for sample of size n were computed. For example,
# for one n=1000, some 965 simulated values were returned. The Spearman Rho is
rhoCOP(cop=prod2COP, para=para) # -0.4271195 (theoretical)
rhoCOP(para=UV, as.sample=TRUE) # -0.4284223 (the test n=1000 --> n=965 sample)
## End(Not run)
```

psepolar

Pseudo-Polar Representation of Bivariate Data

Description

Kiriliouk et al. (2016, pp. 358–360) describe a pseudo-polar representation of bivariate data as a means to explore right-tail extremal dependency between the variables. Let \((X_i, Y_i)\) (real values) or \((U_i, V_i)\) (as probabilities) for \(i = 1, \ldots, n\) be a bivariate sample of size \(n\). When such data are transformed into a “unit-Pareto” scale by

\[
\hat{X}_i^* = \frac{n}{n + 1 - R_{X,i}} \quad \text{and} \quad \hat{Y}_i^* = \frac{n}{n + 1 - R_{Y,i}},
\]

where \(R\) is `rank()`, then letting each component sum or pseudo-polar radius be defined as

\[
\hat{S}_i = \hat{X}_i^* + \hat{Y}_i^*,
\]

and each respective pseudo-polar angle be defined as

\[
\hat{W}_i = \frac{\hat{X}_i^*}{(\hat{X}_i^* + \hat{Y}_i^*)} = \frac{\hat{X}_i^*}{\hat{S}_i},
\]

a pseudo-polar representation is available for study.

A scatter plot of \(\hat{W}_i\) (horizontal) versus \(\hat{S}_i\) (vertical) will depict a pseudo-polar plot of the data. Kiriliouk et al. (2016) approach the pseudo-polar concept as a means to study extremal dependency in the sense of what are the contributions of the \(X\) and \(Y\) to their sum conditional on the sum being large. The largeness of \(\hat{S}_i\) is assessed by its empirical cumulative distribution function and a threshold \(S_f\) stemming from \(f\) as a nonexceedance probability \(f \in [0, 1]\).

A density plot of the \(\hat{W}_i\) is a representation of extremal dependence. If the density plot shows low density for pseudo-polar angles away from 0 and 1 or bimodality on the edges then weak extremal dependency is present. If the density is substantial and uniform away from the the angles 0 and 1 or if the density peaks near \(\hat{W} \approx 0.5\) then extremal dependency is strong.
Usage

\texttt{psepolar(u, v=NULL, f=0.90, ...)}

Arguments

\textbf{u} \hspace{1em} \text{Nonexceedance probability} \textit{u} \text{ in the} \textit{X} \text{ direction (actually the ranks are used so this can be a real-value argument as well)};

\textbf{v} \hspace{1em} \text{Nonexceedance probability} \textit{v} \text{ in the} \textit{Y} \text{ direction (actually the ranks are used so this can be a real-value argument as well) and if} \texttt{NULL} \text{ then} \textit{u} \text{ is treated as a two column} \texttt{R} \text{ data.frame};

\textbf{f} \hspace{1em} \text{The nonexceedance probability of the distal} \hat{S} \text{ to flag in} \texttt{Shat\_ge_Sf} \text{ column of the output; and}

\textbf{...} \hspace{1em} \text{Additional arguments to pass to the} \texttt{dat2bernqua()} \text{ function of the} \texttt{lmomco} \text{ package.}

Value

An \texttt{R} \texttt{data.frame} is returned in the \texttt{table} element and the \texttt{Sf} is in the \texttt{Sf} element.

\textbf{U} \hspace{1em} \text{An echo of the} \textit{u} \text{ input;}

\textbf{V} \hspace{1em} \text{An echo of the} \textit{v} \text{ input;}

\textbf{Xstar} \hspace{1em} \text{The} \hat{X}^*_i \text{ (Kiriliouk \textit{et al.}, 2016, eq. 17.8, p. 359);}

\textbf{Ystar} \hspace{1em} \text{The} \hat{Y}^*_i \text{ (Kiriliouk \textit{et al.}, 2016, eq. 17.8, p. 359);}

\textbf{FXhat1} \hspace{1em} \text{The} F_{X,i} = 1 - 1/X^*_i \text{, which is the inverse of Kiriliouk \textit{et al.} (2016, eq. 17.1, p. 354);}

\textbf{FYhat1} \hspace{1em} \text{The} F_{Y,i} = 1 - 1/Y^*_i \text{, which is the inverse of Kiriliouk \textit{et al.} (2016, eq. 17.1, p. 354);}

\textbf{FXhat3} \hspace{1em} \text{The} F_{3,X,i} = (R_{X,i} - 0.5)/n \text{ corresponding to the \textquote{3} alternative identified by Kiriliouk \textit{et al.} (2016, p. 365);}

\textbf{FYhat3} \hspace{1em} \text{The} F_{3,Y,i} = (R_{Y,i} - 0.5)/n \text{ corresponding to the \textquote{3} alternative identified by Kiriliouk \textit{et al.} (2016, p. 365);}

\textbf{What} \hspace{1em} \text{The} \hat{W}_i \text{ (Kiriliouk \textit{et al.}, 2016, eq. 17.9, p. 359);}

\textbf{Shat} \hspace{1em} \text{The} \hat{S}_i \text{ (Kiriliouk \textit{et al.}, 2016, eq. 17.9, p. 359); and}

\textbf{Shat\_ge_Sf} \hspace{1em} \text{A logical on whether the} \hat{S}_i \text{ are larger than} S_f.

Note

The default of \texttt{f=0.90} means that the upper 90th percentile of the component sum will be identified in the output. This percentile is computed by the Bernstein empirical distribution function provided by the \texttt{lmomco} package through the \texttt{dat2bernqua()} function. Suggested arguments for \textbf{...} are \texttt{poly.type=\textquote{Bernstein}} \text{ and} \texttt{bound.type=\textquote{Carv}} \text{ though the former is redundant because it is the default of} \texttt{dat2bernqua()}. 

Author(s)

William Asquith <william.asquith@ttu.edu>
PSP

References

See Also
spectralmeas, stabtaildepf

Examples
## Not run:
```r
pse <- psepolar(simCOP(n=799, cop=PARETOcop, para=4.3, graphics=FALSE), bound.type="Carv")
pse <- pse$table # The Pareto copula has right-tail extreme dependency
plot(1/(1-pse$U), 1/(1-pse$V), col=pse$Shat_ge_Sf+1, lwd=0.8, cex=0.5, log="xy", pch=16)
plot(pse$What, pse$Shat, log="y", col=pse$Shat_ge_Sf+1, lwd=0.8, cex=0.5, pch=16)
plot(density(pse$What[pse$Shat_ge_Sf]), pch=16, xlim=c(0,1)) # then try the
# non-right tail extremal copula PSP as cop=PSP in the above psepolar() call.
## End(Not run)
```

PSP

The Ratio of the Product Copula to Summation minus Product Copula

Description
Compute *PSP copula* (Nelsen, 2006, p. 23) is named by the author (Asquith) for the *copBasic* package and is

\[
PSP(u,v) = \frac{\Pi}{\Sigma - \Pi} = \frac{uv}{u + v - uv},
\]

where \( \Pi \) is the *independence or product copula (P)* and \( \Sigma \) is the sum \( \Sigma = u + v \). The *PSP* \((u,v)\) copula is a special case of the \( N_{4212}(u,v) \) copula (\( N_{4212}\cop \)). The *PSP* is included in *copBasic* because of its simplicity and for pedagogical purposes. The name “PSP” comes from “Product, Summation, Product” to loosely reflect the mathematical formula shown. Nelsen (2006, p. 114) notes that the PSP copula shows up in several families and designates it as “\( \Pi/\Sigma - \Pi \).” The PSP is undefined for \( u = v = 0 \) but no internal trapping is made; calling functions will have to intercept the NaN so produced for \{0,0\}. The *PSP* is left internally untrapping NaN so as to be available to stress other copula utility functions within the *copBasic* package.

Usage
```r
PSP(u, v, ...)
```

Arguments
- **u** Nonexceedance probability \( u \) in the \( X \) direction;
- **v** Nonexceedance probability \( v \) in the \( Y \) direction; and
- **...** Additional arguments to pass, which for this copula are not needed, but given here to support flexible implementation.
Value

Value(s) for the copula are returned.

Author(s)

W.H. Asquith

References


See Also

P, N4212cop

Examples

PSP(0.4,0.6)  
PSP(0,0)      
PSP(1,1)

qua.regressCOP

Perform Quantile Regression using a Copula by Numerical Derivative Method for V with respect to U

Description

Perform quantile regression (Nelsen, 2006, pp. 217–218) using a copula by numerical derivatives of the copula (derCOPinv). If X and Y are random variables having quantile functions \(x(F)\) and \(y(G)\) and letting \(y = \tilde{y}(x)\) denote a solution to \(\Pr[Y \leq y \mid X = x] = F\), where \(F\) is a nonexceedance probability. Then the curve \(y = \tilde{y}(x)\) is the quantile regression curve of \(V\) or \(Y\) with respect to \(U\) or \(X\), respectively. If \(F = 1/2\), then median regression is performed (med.regressCOP). Using copulas, the quantile regression is expressed as

\[
\Pr[Y \leq y \mid X = x] = \Pr[V \leq G(y) \mid U = F(x)] = \Pr[V \leq v \mid U = v] = \frac{\delta C(u,v)}{\delta u},
\]

where \(v = G(y)\) and \(u = F(x)\). The general algorithm is

1. Set \(\delta C(u,v)/\delta u = F\).
2. Solve the regression curve \(v = \tilde{v}(u)\) (provided by derCOPinv), and
3. Replace \(u\) by \(x(u)\) and \(v\) by \(y(v)\).

The last step is optional as step two produces the regression in probability space, which might be desired, and step 3 actually transforms the probability regressions into the quantiles of the respective random variables.
qua.regressCOP

Usage

qua.regressCOP(f=0.5, u=seq(0.01,0.99, by=0.01), cop=NULL, para=NULL, ...)

Arguments

f  A single value of nonexceedance probability \( F \) to perform regression at and
  defaults to median regression \( F = 1/2 \);

u  Nonexceedance probability \( u \) in the \( X \) direction;

cop  A copula function;

para  Vector of parameters or other data structure, if needed, to pass to the copula; and

...  Additional arguments to pass.

Value

An R data.frame of the regressed probabilities of \( V \) and provided \( U = u \) values is returned.

Author(s)

W.H. Asquith

References


See Also

med.regressCOP, derCOPinv, qua.regressCOP.draw

Examples

## Not run:
# Use a positively associated Plackett copula and perform quantile regression
theta <- 10
R <- qua.regressCOP(cop=PLACKETTcop, para=theta) # 50th percentile regression

plot(R$U,R$V, type="l", lwd=6, xlim=c(0,1), ylim=c(0,1), col=8)
lines(R$U,(1+(theta-1)*R$U)/(theta+1), col=4, lwd=1) # theoretical for Plackett, see
# (Nelsen, 2006, p. 218)
R <- qua.regressCOP(f=0.90, cop=PLACKETTcop, para=theta) # 90th-percentile regression
lines(R$U,R$V, col=2, lwd=2)
R <- qua.regressCOP(f=0.10, cop=PLACKETTcop, para=theta) # 10th-percentile regression
lines(R$U,R$V, col=3, lty=2)

mtext("Quantile Regression V wrt U for Plackett copula")#
## End(Not run)

## Not run:
# Use a composite copula with two Placketts with compositing parameters alpha and beta.
para <- list(cop1=PLACKETTcop, cop2=PLACKETTcop,
            para1=0.04, para2=5, alpha=0.9, beta=0.6)
plot(c(0,1),c(0,1), type="n", lwd=3,
## qua.regressCOP.draw

Draw Quantile Regressions using a Copula by Numerical Derivative Method for V with respect to U or U with respect to V
Description

Draw a suite of lines for specified nonexceedance probabilities representing the *quantile regression* (Nelsen, 2006, pp. 217–218) of either $V$ with respect to $U$ or $U$ with respect to $V$ depending upon an argument setting.

Usage

```r
qua.regressCOP.draw(f=seq(0.1, 0.9, by=0.1), fs=0.5, cop=NULL, para=NULL, ploton=TRUE, wrtV=FALSE, col=c(4,2), lwd=c(1,2), lty=1,...)
```

Arguments

- **f**: Nonexceedance probability $F$ to perform quantile regression at and defaults to a 10-percent-interval sequence. This vectorization of $f$ for this function differs from that in `qua.regressCOP` and `qua.regressCOP2`;
- **fs**: A special value of nonexceedance probability to draw with second values to arguments `col` and `lwd` and defaults to the median ($F = 1/2$);
- **cop**: A copula function;
- **para**: Vector of parameters or other data structure, if needed, to pass to the copula;
- **ploton**: A logical to toggle on the plot;
- **wrtV**: If `wrtV=FALSE` call `qua.regressCOP` and perform quantile regression of $V$ with respect to $U$ and if `wrtV=TRUE` call `qua.regressCOP2` and perform regression of $U$ with respect to $V$;
- **col**: A vector of two values for the color of the line to draw, where the first value is used for the $f$ probabilities and the second value is used for the $fs$ probability;
- **lwd**: A vector of two values for the line width of the line to draw, where the first value is used for the $f$ probabilities and the second value is used for the $fs$ probability;
- **lty**: The line type to draw; and
- **...**: Additional arguments to pass.

Value

No values are returned, this function is used for its side effects.

Author(s)

W.H. Asquith

References


See Also

`qua.regressCOP, qua.regressCOP2`
Examples

# See example in qua.regressCOP documentation

Description

Perform quantile regression (Nelsen, 2006, pp. 217–218) using a copula by numerical derivatives of the copula (derCOPinv2). If \( X \) and \( Y \) are random variables having quantile functions \( x(F) \) and \( y(G) \) and letting \( x = \tilde{x}(y) \) denote a solution to \( \Pr[X \leq x \mid Y = y] = F \), where \( F \) is a nonexceedance probability. Then the curve \( x = \tilde{x}(y) \) is the quantile regression curve of \( U \) or \( X \) with respect to \( V \) or \( Y \), respectively. If \( F = 1/2 \), then median regression is performed (med.regressCOP2). Using copulas, the quantile regression is expressed as

\[
\Pr[X \leq x \mid Y = y] = \Pr[U \leq F(x) \mid V = F] = \Pr[U \leq u \mid V = F] = \frac{\delta C(u, v)}{\delta v},
\]

where \( v = G(y) \) and \( u = F(x) \). The general algorithm is

1. Set \( \delta C(u, v)/\delta v = F \),
2. Solve the regression curve \( u = \tilde{u}(v) \) (provided by derCOPinv2), and
3. Replace \( u \) by \( x(u) \) and \( v \) by \( y(v) \).

The last step is optional as step two produces the regression in probability space, which might be desired, and step 3 actually transforms the probability regressions into the quantiles of the respective random variables.

Usage

qua.regressCOP2(f=0.5, v=seq(0.01,0.99, by=0.01), cop=NULL, para=NULL, ...)

Arguments

- \( f \) A single value of nonexceedance probability \( F \) to perform regression at and defaults to median regression \( F = 1/2; \)
- \( v \) Nonexceedance probability \( v \) in the \( Y \) direction;
- \( \text{cop} \) A copula function;
- \( \text{para} \) Vector of parameters or other data structure, if needed, to pass to the copula; and
- \( \ldots \) Additional arguments to pass.

Value

An R data.frame of the regressed probabilities of \( U \) and \( V = v \) is returned.
ReineckeWell266

Author(s)
W.H. Asquith

References

See Also
med.regressCOP2, derCOPinv2

Examples
```r
## Not run:
# Use a positively associated Plackett copula and perform quantile regression
theta <- 0.10
R <- qua.regressCOP2(cop=PLACKETTcop, para=theta) # 50th percentile regression
plot(R$U,R$V, type="l", lwd=6, xlim=c(0,1), ylim=c(0,1), col=8)
lines((1+(theta-1)*R$V)/(theta+1),R$V, col=4, lwd=1) # theoretical for Plackett,
# compare the theoretical form to that in qua.regressCOP---just switch terms around
# because of symmetry
R <- qua.regressCOP2(f=0.90, cop=PLACKETTcop, para=theta) # 90th-percentile regression
lines(R$U,R$V, col=2, lwd=2)
R <- qua.regressCOP2(f=0.10, cop=PLACKETTcop, para=theta) # 10th-percentile regression
lines(R$U,R$V, col=2, lty=2)
mtext("Quantile Regression U wrt V for Plackett copula")
## End(Not run)
```

Description

These data represent porosity and permeability data from laboratory analysis for Well-266 Reinecke Oil Field, Horseshoe Atoll, Texas as used for the outstanding article by Saller and Dickson (2011). Dr. A.H. Saller shared a CSV file with the author of the copBasic package sometime in 2011. These data are included in this package because of the instruction potential of the bivariate relation between the geologic properties of permeability and porosity.

Usage
data(ReineckeWell266)
ReineckeWells

Format
An R data.frame with

- **WELLNO**: The number of the well, no. 266;
- **DEPTH**: The depth in feet to the center of the incremental spacings of the data;
- **FRACDOLOMITE**: The fraction of the core sample that is dolomite, 0 is 100 percent limestone;
- **Kmax**: The maximum permeability without respect to orientation in millidarcies;
- **POROSITY**: The porosity of the core sample; and
- **DOLOMITE**: Is the interval treated as dolomite (1) or limestone (0).

References

Examples
```r
## Not run:
data(ReineckeWell266)
summary(ReineckeWell266) # show summary statistics
## End(Not run)
```

ReineckeWells

**Porosity and Permeability Data for the Reinecke Oil Field, Horseshoe Atoll, Texas**

Description
These data represent porosity and permeability data from laboratory analysis for the Reinecke Oil Field, Horseshoe Atoll, Texas as used for the outstanding article by Saller and Dickson (2011). Dr. A.H. Art Saller shared a CSV file with the author of the copBasic package sometime in 2011. These data are included in this package because of the instruction potential of the bivariate relation between the geologic properties of permeability and porosity.

Usage
data(ReineckeWells)

Format
An R data.frame with

- **DOLOMITE**: The fraction of the core sample that is dolomite, 0 is 100 percent limestone;
- **Kmax**: The maximum permeability without respect to orientation in millidarcies;
- **K90**: The horizontal (with respect to 90 degrees of the borehole) permeability in millidarcies;
- **Kvert**: The vertical permeability in millidarcies; and
- **POROSITY**: The porosity of the core sample.
References

Examples
```r
## Not run:
data(ReineckeWells)
summary(ReineckeWells) # show summary statistics
## End(Not run)
```

### RFcop

**Description**

The *Raftery copula* (Nelsen, 2006, p. 172) is

\[
C_\Theta(u, v) = RF(u, v) = M(u, v) + \frac{1 - \Theta}{1 + \Theta} \left( uv \right)^{1/(1-\Theta)} \left[ 1 - \left( \max\{u, v\} \right)^{(1+\Theta)/(1-\Theta)} \right],
\]

where \( \Theta \in (0, 1) \). The copula, as \( \Theta \to 0^+ \) limits, to the *independence copula* \( (P(u, v); P) \), and as \( \Theta \to 1^- \), limits to the *comonotonicity copula* \( (M(u, v); M) \). The parameter \( \Theta \) is readily computed from *Spearman Rho* \( (\rho_{COP}) \) by \( \rho_C = \Theta(4 - 3\Theta)/(2 - \Theta)^2 \) or from *Kendall Tau* \( (\tau_{COP}) \) by \( \tau_C = 2\Theta/(3 - \Theta) \).

**Usage**

```r
RFcop(u, v, para=NULL, rho=NULL, tau=NULL, fit=c("rho", "tau"), ...)
```

**Arguments**

- **u** Nonexceedance probability \( u \) in the \( X \) direction;
- **v** Nonexceedance probability \( v \) in the \( Y \) direction;
- **para** A vector (single element) of parameters—the \( \Theta \) parameter of the copula;
- **rho** Optional Spearman Rho from which the parameter will be estimated and presence of \( \rho \) trumps \( \tau \);
- **tau** Optional Kendall Tau from which the parameter will be estimated;
- **fit** If \( \text{para, rho, and tau} \) are all \( \text{NULL} \), the the \( u \) and \( v \) represent the sample. The measure of association by the \( \text{fit} \) declaration will be computed and the parameter estimated subsequently. The \( \text{fit} \) has not other utility than to trigger which measure of association is computed internally by the \( \text{cor} \) function in \( \text{R} \); and
- **...** Additional arguments to pass.
Value

Value(s) for the copula are returned. Otherwise if either \( \rho \) or \( \tau \) is given, then the \( \Theta \) is computed and a list having

- \( \text{para} \): The parameter \( \Theta \);
- \( \rho \): Spearman Rho if the \( \rho \) is given; and
- \( \tau \): Kendall Tau if the \( \tau \) is given but also if both \( \rho \) and \( \tau \) are NULL as mentioned next.

and if \( \text{para}=\text{NULL} \) and \( \rho \) and \( \tau =\text{NULL} \), then the values within \( u \) and \( v \) are used to compute Kendall Tau and then compute the parameter, and these are returned in the aforementioned list.

Author(s)

W.H. Asquith

References


See Also

M, P

Examples

```r
# Lower tail dependency of Theta = 0.5 --> 2*(0.5)/(1+0.5) = 2/3 (Nelsen, 2006, p. 214)
taildepCOP(cop=RFcop, para=0.5)$lambdaL # 0.66667

## Not run:
# Simulate for a Spearman Rho of 0.7, then extract estimated Theta that
# internally is based on Kendall Tau of U and V, then convert estimate
# to equivalent Rho.
set.seed(1)
UV <- simCOP(1000, cop=RFcop, RFcop(rho=0.7)$para)
Theta <- RFcop(UV$U, UV$V, fit="tau")$para # 0.607544
Rho <- Theta*(4-3*Theta)/(2-Theta)^2 # 0.682255 (nearly 0.7)
## End(Not run)
```

A Dependence Measure for a Bivariate Extreme Value Copula based on the Expectation of the Product of Negated Log-Transformed Random Variables \( U \) and \( V \)
**rhobevCOP**

**Description**

Compute a dependence measure based on the expectation of the product of transformed random variables $U$ and $V$, which unnamed by Joe (2014, pp. 383–384) but symbolically is $\rho_E$, having a bivariate extreme value copula $C_{BEV}(u,v)$ by

$$\rho_E = E[(-\log U) \times (-\log V)] - 1 = \int_0^1 [B(w)]^{-2} \, dw - 1,$$

where $B(w) = A(w, 1-w)$, $B(0) = B(1) = 1$, $B(w) \geq 1/2$, and $0 \leq w \leq 1$, and where only bivariate extreme value copulas can be written as

$$C_{BEV}(u,v) = \exp[-A(-\log u, -\log v)],$$

and thus in terms of the coupla

$$B(w) = -\log[C_{BEV}(\exp[-w], \exp[w - 1])].$$

Joe (2014, p. 383) states that $\rho_E$ is the correlation of the “survival function of a bivariate min-stable exponential distribution,” which can be assembled as a function of $B(w)$. Joe (2014, p. 383) also shows the following expression for Spearman Rho

$$\rho_S = 12 \int_0^1 [1 + B(w)]^{-2} \, dw - 3,$$

in terms of $B(w)$. This expression, in conjunction with rhoCOP, was used to confirm the prior expression shown here for $B(w)$ in terms of $C_{BEV}(u,v)$. Lastly, for independence ($uv = \Pi$; $P$), $\rho_E = 0$ and for the Fréchet–Hoeffding upper-bound copula (perfect positive association), $\rho_E = 1$.

**Usage**

```r
rhobevCOP(cop=NULL, para=NULL, as.sample=FALSE, brute=FALSE, delta=0.002, ...)
```

**Arguments**

- `cop` A bivariate extreme value copula function—the function rhobevCOP makes no provision for verifying whether the copula in `cop` is actually an extreme value copula;
- `para` Vector of parameters or other data structure, if needed, to pass to the copula;
- `as.sample` A logical controlling whether an optional R data.frame in `para` is used to compute a $\hat{\rho}_E$ by `mean()` of the product of negated `log()`’s in R. The user is required to cast `para` into estimated probabilities (see Examples);
- `brute` Should brute force be used instead of two nested `integrate()` functions in R to perform the double integration;
- `delta` The $dw$ for the brute force (brute=TRUE) integration; and
- `...` Additional arguments to pass.

**Value**

The value for $\rho_E$ is returned.
Author(s)

W.H. Asquith

References


See Also

rhoCOP, tauCOP

Examples

Theta <- GHcop(tau=1/3)$para  # Gumbel-Hougaard copula with Kendall Tau = 1/3
rhoCOP(cop=GHcop, para=Theta) # 0.3689268 (Spearman Rho)

## Not run:
set.seed(394)
Theta <- GHcop(tau=1/3)$para  # Gumbel-Hougaard copula with Kendall Tau = 1/3
simUV <- simCOP(n=30000, cop=GHcop, para=Theta, graphics=FALSE) # large simulation
samUV <- simUV * 150; n <- length(samUV[,1]) # convert to fake unit system
samUV[,1] <- rank(simUV[,1]-0.5)/n; samUV[,2] <- rank(simUV[,2]-0.5)/n # hazen
rhoCOP(cop=GHcop, para=Theta, as.sample=TRUE) # 0.3708275

## End(Not run)

rhoCOP  

The Spearman Rho of a Copula

Description

Compute the measure of association known as the Spearman Rho $\rho_C$ of a copula according to Nelsen (2006, pp. 167–170, 189, 208) by

$$\rho_C = 12 \int \int_{I^2} C(u, v) \, du \, dv - 3,$$

or

$$\rho_C = 12 \int \int_{I^2} [C(u, v) - uv] \, du \, dv,$$

where the later equation is implemented by rhoCOP as the default method (method="default"). This equation, here having $p = 1$ and $k_p(1) = 12$, is generalized under hoefCOP. The absence of the 12 in the above equation makes it equal to the covariance defined by the Hoeffding Identity (Joe, 2014, p. 54):

$$\text{cov}(U, V) = \int \int_{I^2} [C(u, v) - uv] \, du \, dv \text{ or}$$

$$\text{cov}(U, V) = \int \int_{I^2} [\hat{C}(u, v) - uv] \, du \, dv,$$

which is
\[
\text{cov}(U, V) = \int\int_{I^2} [u + v - 1 + C(1-u, 1-v) - uv] \, du \, dv.
\]

Depending on copula family (Joe, 2014, pp. 56 and 267), the alternative formulation for \(\rho_C\) could be used

\[
\rho_C = 3 - 12 \int\int_{I^2} \frac{\delta C(u, v)}{\delta u} \, du \, dv = 3 - 12 \int\int_{I^2} \frac{\delta C(u, v)}{\delta v} \, du \, dv,
\]

where the first integral form corresponds to Joe (2014, eq. 248, p. 56) and is the method="joe21", and the second integral form is the method="joe12".

The integral

\[
\int\int_{I^2} C(u, v) \, du \, dv,
\]

represents the “volume under the graph of the copula and over the unit square” (Nelsen, 2006, p. 170) and therefore \(\rho_C\) is simple a rescaled volume under the copula. The second equation for \(\rho_C\) expresses the “average distance” between the joint distribution and statistical independence \(\Pi = uv\). Nelsen (2006, pp. 175–176) shows that the following relation between \(\rho_C\) and \(\tau_C\) (tauCOP) exists

\[-1 \leq 3\tau - 2\rho \leq 1.\]

**Usage**

```r
rhoCOP(cop=NULL, para=NULL, method=c("default", "joe21", "joe12"),
    as.sample=FALSE, brute=FALSE, delta=0.002, ...)
```

**Arguments**

- `cop` A copula function;
- `para` Vector of parameters or other data structure, if needed, to pass to the copula;
- `method` The form of integration used to compute (see above);
- `as.sample` A logical controlling whether an optional \texttt{R} data.frame in `para` is used to compute the \(\hat{\rho}\) by dispatch to cor() function in \texttt{R} with method = "spearman";
- `brute` Should brute force be used instead of two nested integrate() functions in \texttt{R} to perform the double integration;
- `delta` The \(du\) and \(dv\) for the brute force integration using `brute`; and
- `...` Additional arguments to pass.

**Value**

The value for \(\rho_C\) is returned.

**Note**

Technically, Nelsen (2006) also shows that these definitions are a form of call to a concordance function \(Q(C_1, C_2)\) of two copulas that involve \(C_1=C(u, v)\) and \(C_2=\Pi\). As such in order to keep rhoCOP a small function when `brute=TRUE`, \(\rho_C\) is computed by a special call to tauCOP, which by itself and although titled for computation of Kendall Tau, does support the concordance function \(Q(C_1, C_2)\) [see Nelsen (2006, pp. 158–159)] when given two different copulas and respective parameters as arguments. The well-known Pearson correlation coefficient equals Spearman rho value if random variables \(X\) and \(Y\) are both uniformly distributed on \([0, 1]\).
Author(s)
W.H. Asquith

References

See Also
blomCOP, footCOP, giniCOP, hoefCOP, tauCOP, wolfCOP, joeskewCOP, uvlmoms

Examples
rhoCOP(cop=PSP)    # 0.4784176
## Not run:
rhoCOP(cop=PSP, brute=TRUE) # 0.4684063
# CPU heavy example showing that the dual-integration (fast) results in
# a Spearman Rho that mimics a sample version
do_rho <- function(n) {
  uv <- simCOP(n=n, cop=PSP, ploton=FALSE, points=FALSE)
  return(cor(uv$U, uv$V, method="spearman"))
}
rhos <- replicate(100, do_rho(1000))
rho_sample <- mean(rhos); print(rho_sample) # 0.472661
## End(Not run)
## Not run:
para <- list(cop1=PLACKETTcop, cop2=PLACKETTcop,
             para1=0.00395, para2=4.67, alpha=0.9392, beta=0.5699)
rhoCOP(cop=composite2COP, para=para) # -0.5924796
para <- list(cop1=PLACKETTcop, cop2=PLACKETTcop,
             para1=0.14147, para2=20.96, alpha=0.0411, beta=0.6873)
rhoCOP(cop=composite2COP, para=para) # 0.2818874
para <- list(cop1=PLACKETTcop, cop2=PLACKETTcop,
             para1=0.10137, para2=4492.87, alpha=0.0063, beta=0.0167)
rhoCOP(cop=composite2COP, para=para) # 0.9812919
rhoCOP(cop=composite2COP, para=para, brute=TRUE) # 0.9752155
## End(Not run)
## Not run:
# This is the same composited copula used in a highly asymmetric multi-modal
# plotting example under densityCOPplot(). Let us use that copula as a means to
# check on the Spearman Rho from the alternative formulations from Joe (2014).
para <- list(alpha=0.15, beta=0.90, kappa=0.06, gamma=0.96,
             cop1=GHcop, cop2=PLACKETTcop, para1=5.5, para2=0.07)
"rhoCOPbyJoe21" <- function(cop=NULL, para=NULL, ...) { # Joe (2014, eq. 2.48)
  myint <- NULL
  try(myint <- integrate(function(u){
The Root Mean Square Error between a Fitted Copula and an Empirical Copula

Description

Compute the root mean square error RMSE$_C$ (Chen and Guo, 2019, p. 29), which is computed using mean square error MSE as

$$ \text{MSE}_C = \frac{1}{n} \sum_{i=1}^{n} \left( C_n(u_i, v_i) - C_{\Theta_m}(u_i, v_i) \right)^2 $$

and

$$ \text{RMSE}_C = \sqrt{\text{MSE}_C}, $$

where $C_n(u_i, v_i)$ is the empirical copula (empirical joint probability) for the $i$th observation, $C_{\Theta_m}(u_i, v_i)$ is the fitted copula having $m$ parameters in $\Theta$. The $C_n(u_i, v_i)$ comes from EMPIRCop. The RMSE$_C$ is in effect saying that the best copula will have its joint probabilities plotting on a 1:1 line with the empirical joint probabilities, which is an RMSE$_C = 0$. From the MSE$_C$ shown above, the Akaike information criterion (AIC) aicCOP and Bayesian information criterion (BIC) bicCOP can be computed, which add a penalty for $m$ parameters. These goodness-of-fits can assist in deciding one copula favorability over another, and another goodness-of-fit using the absolute differences between $C_n(u, v)$ and $C_{\Theta_m}(u, v)$ is found under statTn.

Usage

rmseCOP(u, v, cop=NULL, para=NULL, ...)

Arguments

u    Nonexceedance probability $u$ in the $X$ direction;

v    Nonexceedance probability $v$ in the $Y$ direction;

cop  A copula function;
para Vector of parameters or other data structure, if needed, to pass to the copula; and

... Additional arguments to pass to either copula (likely most commonly to the empirical copula).

Value

The value for $\text{RMSE}_C$ is returned.

Author(s)

W.H. Asquith

References


See Also

`EMPIRcop, aicCOP, bicCOP`

Examples

```r
## Not run:
S <- simCOP(80, cop=GHcop, para=5) # Simulate some probabilities, but we
# must then treat these as data and recompute empirical probabilities.
U <- lmomco::pp(S$U, sort=FALSE); V <- lmomco::pp(S$V, sort=FALSE)
# The parent distribution is Gumbel-Hougaard extreme value copula.
# But in practical application we do not know that but say we speculate that
# perhaps the Galambos extreme value might be the parent. Then maximum
# likelihood is used to fit the single parameter.
pGL <- mleCOP(U,V, cop=GLcop, interval=c(0,20))$par

rmse <- c(rmseCOP(U,V, cop=GLcop, para=pGL),
          rmseCOP(U,V, cop=P),
          rmseCOP(U,V, cop=PSP))
names(rmse) <- c("GLcop", "P", "PSP")
print(rmse) # We will see that the first RMSE is the smallest as the
# Galambos has the nearest overall behavior than the P and PSP copulas.
## End(Not run)
```
Description

Compute the copula sections or the (partial) derivatives of copula sections of a copula (Nelsen, 2006, pp. 12–14). The horizontal section at \( V = a \) (a constant) is

\[
t \mapsto C(t, a),
\]

and

the vertical section at \( U = a \) (a constant, with respect to \( V \) or \( \text{wrt} V = \text{TRUE} \)) is

\[
t \mapsto C(a, t).
\]

The partial derivatives of the copula sections are conditional cumulative distribution functions (see \texttt{derCOP} and \texttt{derCOP2}). The derivatives are constrained as

\[
0 \leq \frac{\delta}{\delta u} C(u, v) \leq 1, \quad \text{and}
\]

\[
0 \leq \frac{\delta}{\delta v} C(u, v) \leq 1.
\]

Usage

\texttt{sectionCOP(f, cop=NULL, para=NULL, wrtV=FALSE, dercop=FALSE, delt=0.005, ploton=TRUE, lines=TRUE, xlab=“NONEXCEEDANCE PROBABILITY”, \ldots)}

Arguments

\texttt{f} \quad A single value of nonexceedance probability \( u \) or \( v \) along the horizontal \( U \) axis or vertical \( V \) axis of the unit square \( I^2 \);

\texttt{cop} \quad A copula function;

\texttt{para} \quad Vector of parameters, if needed, to pass to the copula;

\texttt{wrtV} \quad A logical to toggle between with respect to \( v \) or \( u \) (default). The default provides the vertical section whereas the horizontal comes from \( \text{wrt} V = \text{TRUE} \);

\texttt{dercop} \quad A logical that triggers the derivative of the section;

\texttt{delt} \quad The increment of the level curves to plot, defaults to 5-percent intervals;

\texttt{ploton} \quad A logical to toggle on the plot;

\texttt{lines} \quad Draw the lines of diagonal to the current device;

\texttt{xlab} \quad A label for the x-axis title passed to \texttt{plot()} in \texttt{R}; and

\texttt{\ldots} \quad Additional arguments to pass to the \texttt{plot()} and \texttt{lines()} functions in \texttt{R}.

Value

An \texttt{R} list is returned.

\texttt{t} \quad The nonexceedance probability along the section. The nomenclature \( t \) mimics Nelsen (2006) and is \textit{not} the same as the \( u \) or \( v \);

\texttt{seccop} \quad The section of the copula or its derivative;

\texttt{wrt} \quad A text string declaring what the setting for \texttt{wrtV} was;

\texttt{fvalue} \quad The provided value of nonexceedance probability; and

\texttt{isderivative} \quad A logical stating whether the derivative of the section is \texttt{seccop}. 
Author(s)

W.H. Asquith

References


See Also

COP, diagCOP

Examples

```r
## Not run:
# EXAMPLE 1, plot the v=0.55 section and then u=0.55 section, which will overlay
# the other because the PSP is a symmetrical copula
tmp <- sectionCOP(0.55, cop=PSP, ylab="COPULA SECTIONS", lwd=5, col=2)
tmp <- sectionCOP(0.55, cop=PSP, wrtV=TRUE, ploton=FALSE, lwd=2, col=3)
# now add the v=0.85 section and the u=0.85, again overlay each other
tmp <- sectionCOP(0.85, cop=PSP, ploton=FALSE, lwd=5, col=2, lty=2)
tmp <- sectionCOP(0.85, cop=PSP, wrtV=TRUE, ploton=FALSE, lwd=2, col=3, lty=2)
## End(Not run)

## Not run:
# EXAMPLE 2, v=0.35 section and derivative (the conditional distribution) function
tmp <- sectionCOP(0.35, cop=PSP, ylab="COPULA SECTIONS OR DERIV.", lwd=5, col=3)
tmp <- sectionCOP(0.35, cop=PSP, dercop=TRUE, ploton=FALSE, col=3)
# The thin green line represents the cumulative distribution function conditional
# on u = 0.35 from the derCOP function. Then see Example 3
## End(Not run)

## Not run:
# EXAMPLE 3 (random selection commented out)
#para <- list(cop1=PLACKETTcop, cop2=PLACKETTcop, alpha=runif(1), beta=runif(1),
# para1=10^runif(1,min=-4, max=0), para2=10^runif(1,min = 0, max=4))
para <- list(cop1=PLACKETTcop, cop2=PLACKETTcop, alpha=0.7, beta=0.22,
para1=0.0155, para2=214.4)
txts <- c("Alpha=", round(para$alpha, digits=4),
"; Beta="; round(para$beta, digits=4),
"; Theta1="; round(para$para1[1], digits=5),
"; Theta2="; round(para$para2[1], digits=2))
layout(matrix(1:2,byrow=TRUE))
D <- simCOP(n=1000, cop=composite2COP, para=para, cex=0.5, col=rgb(0,0,0,0.2), pch=16)
mtext(paste(txts,collapse=""), xlab="X", ylab="Y")
##f <- c(runif(1),runif(1))
f <- c(0.2,0.9) # RED is the horizontal section and BLACK is the vertical section
segments(f[1],0,f[1],1, col=2, lwd=2); segments(0,f[2],1,f[2], lwd=2)
ftxt <- c("Sections (thick) and derivatives (thin) at ", f, " nonexceed. prob.")
tmp <- sectionCOP(f[1],cop=composite2COP,para=para, col=2, lwd=4)
tmp <- sectionCOP(f[1],cop=composite2COP,para=para, dercop=TRUE, ploton=FALSE, col=2)
tmp <- sectionCOP(f[2],cop=composite2COP,para=para, wrtV=TRUE,ploton=FALSE,lwd=4)
tmp <- sectionCOP(f[2],cop=composite2COP,para=para, wrtV=TRUE,ploton=FALSE,dercop=TRUE)
```
semicorCOP

Lower and Upper Semi-Correlations of a Copula

Description

Compute the lower semi-correlations (bottom-left)

$$\rho_{C}^{\pm}(u, v; a) = \rho_{N}^{\pm}(a)$$

and

compute the upper semi-correlations (top-right)

$$\rho_{C}^{\pm+}(u, v; a) = \rho_{N}^{++}(a)$$

of a copula $C(u, v)$ (Joe, 2014, p. 73) using numerical simulation. The semi-correlations are defined as

$$\rho_{N}^{\pm}(a) = \text{cor}[Z_1, Z_2|Z_1 < -a, Z_2 < -a],$$

$$\rho_{N}^{++}(a) = \text{cor}[Z_1, Z_2|Z_1 > +a, Z_2 > +a],$$

and

$$\rho_{N}(a > -\infty) = \text{cor}[Z_1, Z_2],$$

where $\text{cor}[Z_1, Z_2]$ is the familiar Pearson correlation function, which is in R the syntax $\text{cor}(\ldots, \text{method}="\text{pearson}"")$, parameter $a \geq 0$ is a truncation point that identifies truncated tail regions (Joe, 2014, p. 73), and lastly $(Z_1, Z_2) \sim C(\Phi, \Phi)$ and thus from the standard normal distribution $(Z_1, Z_2) = (\Phi^{-1}(u), \Phi^{-1}(v))$ where the random variables $(U, V) \sim C$.

The semi-correlations are extended for the copBasic package into bottom right and top left versions as well by

$$\rho_{N}^{+\pm}(a) = \text{cor}[Z_1, Z_2|Z_1 > +a, Z_2 < -a],$$

and

$$\rho_{N}^{-\pm}(a) = \text{cor}[Z_1, Z_2|Z_1 < -a, Z_2 > +a].$$

As a result, the notations $-\, -, \, +\, +$, $+\, -$, and $-\, +$ can be used to represent each of the respective corners bottom-left, top-right, bottom-right, and top-left of the $(u, v)$ domain with the respective truncation. These words are used in the variable names of the returned list from the function.

Usage

```r
semicorCOP(cop=NULL, para=NULL, truncation=0, n=0, as.sample=FALSE, ...)
```
Arguments

- **cop**: A copula function;
- **para**: Vector of parameters or other data structure, if needed, to pass to the copula;
- **truncation**: The truncation value for \( a \), which is in standard normal variates, and the default of zero is the origin (medians);
- **n**: The sample size \( n \) for simulation estimates of the \( \rho_N \);
- **as.sample**: A logical controlling whether an optional data.frame in para is used to compute the \( \hat{\rho}_N \) (see Note); and
- **...**: Additional arguments to pass to the copula.

Value

The value(s) for \( \rho_N, \rho_N^-, \rho_N^+, \rho_N^{++}, \) and \( \rho_N^{-+} \) are returned.

Note

The sample semi-correlations can be computed from a two-column table that is passed into the function using the para argument. Although the truncation point \( a \geq 0 \), as \( a \) increases and focus is increasingly made into one or the other truncated tail regions, the sample version with data becomes decreasing well estimated because the available sample size diminishes. The para argument can contain probabilities or raw data because internally the function computes the Hazen plotting positions (e.g. \( u_i = (i - 0.5)/n \) for rank \( i \) and sample size \( n \)) because Joe (2014, pp. 9, 17, 245, 247–248) repeatedly emphasizes this form of plotting position when normal scores are involved.

Author(s)

W.H. Asquith

References


See Also

giniCOP, rhoCOP, tauCOP, COP

Examples

```r
## Not run:
# Gumbel-Hougaard copula with Pearson rhoN = 0.4 (by definition)
run <- sapply(1:50, function(i) semicorCOP(cop=GHcop, para=1.350, n=600))
mean(unlist(run[1,])) # cor.normal.scores
mean(unlist(run[2,])) # botleft.semicor
mean(unlist(run[3,])) # topright.semicor
sd( unlist(run[1,])) # cor.normal.scores (These are our sampling variations
sd( unlist(run[2,])) # botleft.semicor for the n=600 used as a Monte
sd( unlist(run[3,])) # topright.semicor Carlo simulation.)
# The function returns: rhoN = 0.392112, rhoN-- = 0.117674, rhoN++ = 0.404733
# standard deviations (0.038883) (0.073392) (0.073942)
```
Joe (2014, p. 72) shows: \( \rho_N = 0.4 \), \( \rho_N^- = 0.132 \), \( \rho_N^+ = 0.415 \).

We see alignment with the results of Joe with his \( n = 600 \).

---

```r
## Not run:
p <- 0.5 # Reasonable strong positive association for the Raftery copula,
# but then we are going to be reflecting and rotating the copula.
# See similar Example under COP() function.

"RFcop1" <- function(u,v, para) COP(u,v, cop=RFcop, para=para, reflect="1")
"RFcop2" <- function(u,v, para) COP(u,v, cop=RFcop, para=para, reflect="2")
"RFcop3" <- function(u,v, para) COP(u,v, cop=RFcop, para=para, reflect="3")
"RFcop4" <- function(u,v, para) COP(u,v, cop=RFcop, para=para, reflect="4")
RF <- NULL; n <- 10000
RF <- rbind(RF, as.data.frame(semicorCOP(cop=RFcop1, para=p, n=n))[,1:5])
RF <- rbind(RF, as.data.frame(semicorCOP(cop=RFcop2, para=p, n=n))[,1:5])
RF <- rbind(RF, as.data.frame(semicorCOP(cop=RFcop3, para=p, n=n))[,1:5])
RF <- rbind(RF, as.data.frame(semicorCOP(cop=RFcop4, para=p, n=n))[,1:5])
print(RF[,1:3]) # total sample and lower and upper semi-correlations
  # cor.normal.scores botleft.semicor topright.semicor
  # 1 0.5587837 0.74124686 0.10027641
  # 2 0.5567889 0.10302772 0.73729702
  # 3 -0.5807201 -0.04683536 -0.01714573 # see near zeros --,++
  # 4 -0.5698139 0.03040520 0.05125916 # see near zeros --,++
print(RF[,2:5]) # now look at all four corners
  # botleft.semicor topright.semicor topleft.semicor botright.semicor
  # 1 0.74124686 0.10027641 0.01529508 0.02046530
  # 2 0.10382772 0.73729702 -0.05195628 0.01747874
  # 3 -0.04683536 -0.01714573 -0.11106842 -0.74077321
  # 4 0.03040520 0.05125916 -0.74516061 -0.07636233
# Notice how the tight tail of the copula, being reflected and rotated
# into each of the four corners, shows semi-correlation magnitude of 0.74.
# See the copula density plots of COP() Examples section.
## End(Not run)
```

---

### simcomposite3COP

**Compute the L-comoments of a Four-Value Composited Copula by Simulation**

**Description**

Simulate copula parameters and compute \( L\)-comoments and provision for plotting features for a composited copula using four compositing parameters (see `composite3COP`). The compositing parameters are each independent and uniformly distributed:

\[
\alpha \sim U[0, 1]; \ \beta \sim U[0, 1]; \ \kappa \sim U[0, 1]; \ \gamma \sim U[0, 1].
\]

L-comoment estimation is provided by the `lcomCOP`. 

---

**Simulate copula parameters and compute L-comoments and provision for plotting features for a composited copula using four compositing parameters (see composite3COP).** The compositing parameters are each independent and uniformly distributed:

\[
\alpha \sim U[0, 1]; \ \beta \sim U[0, 1]; \ \kappa \sim U[0, 1]; \ \gamma \sim U[0, 1].
\]

L-comoment estimation is provided by the `lcomCOP`.
Usage

simcomposite3COP(nsim=100, compositor=composite3COP,
   parents=NULL, ploton=FALSE, points=FALSE,
   showpar=FALSE, showresults=FALSE, digits=6, ...)

Arguments

nsim   Number of simulations to perform;
compositor   The compositing function that could be either composite1COP, composite2COP,
   and composite3COP;
parents   A special parameter list (see Note);
ploton   A logical to toggle on intermediate plotting;
points   A logical to actually draw the simulations on the ploton by the points() func-
   tion in R;
showpar   Print the simulated parameter set with each iteration;
showresults   Print the results (useful if harvest results from a batch operation in R);
digits   The number digits to pass to round if showresults is true; and
...      Additional arguments to pass.

Value

An R matrix of results is returned. Each row represents a single simulation run. The first four
columns are the \(\alpha, \beta, \kappa,\) and \(\gamma\) compositing parameters and are labeled as such. The next two
columns are the opposing diagonals, by first row and then second, of the L-comoment corre-
lation. The following two columns are the opposing diagonals, by row and then second, of the
L-coskew. The following two columns are the opposing diagonals, by row and then second, of
the L-cokurtosis. The L-comoment columns are labeled to reflect the L-comoment matrix: \(T_{2.21}\)
means the L-comoment correlation row 2 column 1 and \(T_{3.12}\) mean the L-coskew row 1 column 2.
The remaining columns represent the \(\Theta_n\) parameters for copula 1, the \(\Theta_m\) parameters for copula 2.
The columns are labeled Cop1Thetas or Cop2Thetas.

Note

The following descriptions list in detail the parents argument structure and content of the para
argument:

cop1    — Function of the first copula;
cop2    — Function of the second copula;
para1gen    — Function to generate random parameters for the first copula; and
para2gen    — Function to generate random parameters for the second copula.

The para argument of this function are passed to the function contained in compositor and are
therefore subject to further constraints in items should such constraints exist.

Author(s)

W.H. Asquith
References

See Also
lcomCOP, simcompositeCOP

Examples
## Not run:
# EXAMPLE 1: Make a single simulation result.
mainpara <- list(cop1=PLACKETTcop, cop2=PLACKETTcop,
                 para1gen=function() { return(c(10^runif(1, min=-5, max=0))) },
                 para2gen=function() { return(c(10^runif(1, min= 0, max=5))) })
v <- simcompositeCOP(nsim=1, parent=mainpara, showresults=TRUE)
print(v)

# EXAMPLE 2: Make 1000 "results" and plot two columns.
mainpara <- list(cop1=PLACKETTcop, cop2=N4212cop,
                 para1gen=function() { return(c(10^runif(1, min=-5, max=5))) },
                 para2gen=function() { return(c(10^runif(1, min= 0, max=2))) })
v <- simcompositeCOP(nsim=100, parent=mainpara); labs <- colnames(v)
plot(v[,5], v[,7], # open circles are 1 with respect to 2
     xlab=paste(c(labs[5], "and", labs[6]), collapse=" "),
     ylab=paste(c(labs[6], "and", labs[8]), collapse=" "))
points(v[,6], v[,8], pch=16) # black dots are 2 with respect to 1
## End(Not run)

simcompositeCOP  Compute the L-comoments of a Two-Value Composited Copula by Simulation

Description
Simulate copula parameters and compute L-comoments and provision for plotting features for a composited copula using using two compositing parameters (see composite1COP as well as composite2COP). The compositing parameters are each independent and uniformly distributed:

\[ \alpha \sim U[0, 1]; \beta \sim U[0, 1]. \]

L-comoment estimation is provided by the lcomCOP.

Usage
simcompositeCOP(nsim=100, compositor=composite2COP,
                parents=NULL, ploton=FALSE, points=FALSE,
                showpar=FALSE, showresults=FALSE, digits=6, ...)

## Not run:
# EXAMPLE 1: Make a single simulation result.
mainpara <- list(cop1=PLACKETTcop, cop2=PLACKETTcop,
                 para1gen=function() { return(c(10^runif(1, min=-5, max=0))) },
                 para2gen=function() { return(c(10^runif(1, min= 0, max=5))) })
v <- simcompositeCOP(nsim=1, parent=mainpara, showresults=TRUE)
print(v)

# EXAMPLE 2: Make 1000 "results" and plot two columns.
mainpara <- list(cop1=PLACKETTcop, cop2=N4212cop,
                 para1gen=function() { return(c(10^runif(1, min=-5, max=5))) },
                 para2gen=function() { return(c(10^runif(1, min= 0, max=2))) })
v <- simcompositeCOP(nsim=100, parent=mainpara); labs <- colnames(v)
plot(v[,5], v[,7], # open circles are 1 with respect to 2
     xlab=paste(c(labs[5], "and", labs[6]), collapse=" "),
     ylab=paste(c(labs[6], "and", labs[8]), collapse=" "))
points(v[,6], v[,8], pch=16) # black dots are 2 with respect to 1
## End(Not run)
Arguments

nsim  Number of simulations to perform;
compositor  The compositing function, could be either composite1COP or composite2COP. Each of these is acceptable because two compositing parameters are used;
parents  A special parameter list (see Note);
ploton  A logical to toggle on intermediate plotting;
points  A logical to actually draw the simulations on the ploton by points() function in R;
showpar  Print the simulated parameter set with each iteration;
showresults  Print the results (useful if harvest results from a batch operation in R);
digits  The number digits to pass to round if showresults is true; and
...
  Additional arguments to pass.

Value

An R matrix of results is returned. Each row represents a single simulation run. The first two columns are the \( \alpha \) and \( \beta \) compositing parameters and are labeled as such. The next two columns are the opposing diagonals, by first row and then second, of the L-comoment correlation. The following two columns are the opposing diagonals, by row and then second, of the L-coskew. The following two columns are the opposing diagonals, by row and then second, of the L-cokurtosis. The L-comoment columns are labeled to reflect the L-comoment matrix: T2.21 means the L-comoment correlation row 2 column 1 and T3.12 mean the L-coskew row 1 column 2. The remaining columns represent the \( \Theta_n \) parameters for copula 1, the \( \Theta_m \) parameters for copula 2. The columns are labeled Cop1Thetas or Cop2Thetas.

Note

The following descriptions list in detail the parents argument structure and content of the para argument:

cop1  — Function of the first copula;
cop2  — Function of the second copula;
para1gen  — Function to generate random parameters for the first copula; and
para2gen  — Function to generate random parameters for the second copula.

The para argument of this function are passed to the function contained in compositor and are therefore subject to further constraints in items should such constraints exist.

Author(s)

W.H. Asquith

References

See Also

lcomCOP, simcomposite3COP

Examples

```r
## Not run:
# A single simulation result.
mainpara <- list(cop1=PLACKETTcop, cop2=PLACKETTcop,
                 para1gen=function() { return(c(10^runif(1,min=-5,max=0))) },
                 para2gen=function() { return(c(10^runif(1,min=0,max=5))) })
v <- simcompositeCOP(nsim=1, parent=mainpara, showresults=TRUE)
print(v) # for review
## End(Not run)
```

```r
simCOP(n=100, cop=NULL, para=NULL, na.rm=TRUE, seed=NULL, keep=FALSE,
       graphics=TRUE, ploton=TRUE, points=TRUE, snv=FALSE,
       infsnv.rm=TRUE, trapinfsnv=.Machine$double.eps, ...)
rCOP(n, cop=NULL, para=NULL, na.rm=TRUE, seed=NULL, ...)
```
Arguments

- **n**  A sample size, default is $n = 100$;
- **cop**  A copula function;
- **para**  Vector of parameters, if needed, to pass to the copula;
- **na.rm**  A logical to toggle the removal of NA entries should they form on the returned data.frame. A well implemented copula should accommodate and not return NA but because this package relies on numerical derivation, it was decided to have a mechanism to handle this;
- **seed**  The integer seed to pass immediately to set.seed();
- **keep.t**  Keep the $t$ uniform random variable for the simulation as the last column in the returned data.frame;
- **graphics**  A logical that will disable graphics by setting ploton and points to FALSE and overriding whatever their settings were;
- **ploton**  A logical to toggle on the plot (see Examples in vuongCOP);
- **points**  A logical to actually draw the simulations by the points() function in R;
- **snv**  A logical to convert the $\{u,v\}$ to standard normal scores (variates) both for the optional graphics and the returned R data.frame. Curiously, Joe (2014) advocates extensively for use of normal scores, which is in contrast to Nelsen (2006) who does not;
- **infsnv.rm**  A logical that will quietly strip out any occurrences of $u = \{0,1\}$ or $v = \{0,1\}$ from the simulations because these are infinity in magnitude when converted to standard normal variates has been selected. Thus, this logical only impacts logic flow when snv is TRUE. The infsnv.rm is mutually exclusive from trapinfsnv;
- **trapinfsnv**  If TRUE and presumably small, the numerical value of this argument ($\eta$) is used to replace $u = \{0,1\}$ and $v = \{0,1\}$ with $u(0) = v(0) = \eta$ or $u(1) = v(1) = 1 - \eta$ as appropriate when conversion to standard normal variates has been selected. The setting of trapinfsnv only is used if snv is TRUE and infsnv.rm is FALSE; and
- ...  Additional arguments to pass to the points() function in R.

Value

An R data.frame of the simulated values is returned.

Note

Function rCOP is a light-weight implementation for bivariate copula random variates that dispatches to simCOPmicro and bypasses the graphical and other features of simCOP. Finally, an experimental parallel for the empirical copula is EMPIRsim.

Author(s)

W.H. Asquith
**simCOPmicro**

**Simulate V from U through a Copula by Numerical Derivative Method**

**Description**

Perform bivariate simulation of random but coupled variables \( V \) from \( U \) through a copula (Nelsen, 2006, p. 32) by inversion of the numerical derivatives of the copula (**derCOPinv**, **derCOPinv2**). The method is more broadly known as *conditional simulation method*. An elaborate implementation is available in **simCOP**, which unlike **simCOPmicro**, has provisions (default) for graphical support. The **simCOPmicro** function is intended to be a minimalist version for copula simulation, and such a version is useful for pedagogic purposes including *conditional distributions*, *conditional quantile functions*, and *copula reflection* (see **Note** and **COP**). An extended educational discussion of simulation using the conditional method is available in the **Note** section of **derCOPinv**.

**References**


**See Also**

**derCOPinv**, **simCOPmicro**

**Examples**

```r
simCOP(n=5, cop=PARETOcop, para=2.4)

## Not run:
# The simCOP function is oft used in other Examples sections through this package.
simCOP(n=10, cop=W)  # Frechet lower-bound copula
simCOP(n=10, cop=P)  # Independence copula
simCOP(n=10, cop=M, col=2)  # Frechet upper-bound copula
simCOP(n=10, cop=PSP)  # The PSP copula
## End(Not run)

## Not run:
# Now simulate the PSP copula, add the level curves of the copula, and demonstrate
# the uniform distribution of marginals on the correct axes (U [top] and V [left]).
D <- simCOP(n=400, cop=PSP)  # store simulated values in D
level.curvesCOP(cop=PSP, ploton=FALSE)
rug(D$U, side=3, col=2); rug(D$V, side=4, col=2)

# Now let us get more complicated and mix two Plackett copulas together using the
# composite2COP as a "compositor." The parameter argument becomes more complex, but
# is passed as shown into composite2COP.
para <- list(cop1=PLACKETTcop,cop2=PLACKETTcop, alpha=0.3,beta=0.5, para1=0.1,para2=50)
D <- simCOP(n=950, cop=composite2COP, para=para, col=rgb(0,0,0.2), pch=16, snv=TRUE)
## End(Not run)
```
Some definitions are needed. The copula of \((1 - U, 1 - V)\) is the *survival copula* (*surCOP*) and is defined as

\[
\hat{C}(u, v) = u + v - 1 + C(1 - u, 1 - v),
\]

whereas, following the notation of Joe (2014, pp. 271–272), the copula of \((1 - U, V)\) is defined as

\[
\acute{C}(u, v) = v - C(1 - u, v),
\]

and the copula of \((U, 1 - V)\) is defined as

\[
\grave{C}(u, v) = u - C(u, 1 - v).
\]

Careful consideration of the nomenclature is necessary as confusion with the occurrences of \(1 - u\) and \(1 - v\) easily conflate meaning. The nomenclature for the *survival copula* is more elaborately shown under *surCOP*. The difficulty is that the bivariate arguments to the *survival copula* are exceedance probabilities.

For simulation, again following the nomenclature of Joe (2014, p. 272), the conditional distribution functions (numerical derivatives; \(\text{derCOP} \equiv C_2|1\)(v|u) and \(\text{derCOP}^2 \equiv C_{1|2}(u|v)\)) can be written in terms of \(C(u|v) = C_2|1\)(v|u) as

\[
\hat{C}_{2|1}(v|u) = 1 - C_{2|1}(1 - v|1 - u),
\]

\[
\acute{C}_{2|1}(v|u) = C_{2|1}(v|1 - u),
\]

\[
\grave{C}_{2|1}(v|u) = 1 - C_{2|1}(1 - v|u),
\]

where the respective "surv", "acute", and "grave" are inverses (conditional quantile functions; inverses of numerical derivatives; \(\text{derCOP}^{-1} \equiv C_{2|1}^{-1}(1|v)\) and \(\text{derCOP}^{-2} \equiv C_{1|2}^{-1}(u|v)\)) are

\[
\hat{C}_{2|1}^{-1}(u|v) = 1 - C_{2|1}^{-1}(1 - t|1 - u) \rightarrow "sur",
\]

\[
\acute{C}_{2|1}^{-1}(u|v) = C_{2|1}^{-1}(t|1 - u) \rightarrow "acute",\text{ and}
\]

\[
\grave{C}_{2|1}^{-1}(u|v) = 1 - C_{2|1}^{-1}(1 - t|u) \rightarrow "grave",\n\]

where \(t\) is a uniformly distributed variable.

To clarify the seemingly clunky nomenclature—Joe (2014) does not provide “names” for \(\hat{C}(u, v)\) or \(\acute{C}(u, v)\)—the following guidance is informative:

1. "surv" or \(\hat{C}(u, v)\) is a reflection of \(U\) and \(V\) on the horizontal and vertical axes, respectively
2. "acute" or \(\acute{C}(u, v)\) is a reflection of \(U\) on the horizontal axis, and
3. "grave" or \(\grave{C}(u, v)\) is a reflection of \(V\) on the vertical axis.

The names “acute” and "grave" match those used in the Rd-format math typesetting instructions.

**Usage**

\[
simCOPmicro(u, \text{cop=NULL, para=NULL, seed=NULL, reflect=c("cop", "surv", "acute", "grave", "1", "2", "3", "4"), ...)}
\]

\[
simCOPv(u, \text{cop=NULL, para=NULL, reflect=c("cop", "surv", "acute", "grave", "1", "2", "3", "4"), ...})
\]
Arguments

u Nonexceedance probability $u$ in the $X$ direction. The `runif()` function in R can be used to drive conditional simulation using the `simCOPmicro` function (see Examples);

cop A copula function;

para Vector of parameters, if needed, to pass to the copula;

seed The integer seed to pass immediately to `set.seed()` and setting it for the `simCOPv` version will dispatch through the triple dots down to `simCOPmicro`;

reflect The reflection of the copula (see above) and the default "cop" or "1" is the usual copula definition. The numbered values correspond, respectively, to the named values; and

... Additional arguments to pass should they be needed.

Value

Simulated value(s) of nonexceedance probability $v$ are returned based on the nonexceedance probabilities $u$ in argument $u$.

Note

The advanced features of `simCOPmicro` permit simulation of the three permutations of variable reflection. The first code simply produce four different “copulas” based on the Gumbel–Hougaard copula $GH(u, v; \Theta)$; $GHcop$, which has substantial upper tail dependency but no lower tail dependency for $\Theta = 2.512$ as quantified by the `taildepCOP` function call.

```r
U <- runif(1500); G <- 2.512; u <- 0.1; up <- 1-u; v <- 0.2; vp <- 1-v
UV <- data.frame(U, simCOPmicro(U, cop=GHcop, para=G ))
sUsV <- data.frame(U, simCOPmicro(U, cop=GHcop, para=G, reflect="surv" ))
sUV <- data.frame(U, simCOPmicro(U, cop=GHcop, para=G, reflect="acute") )
UsV <- data.frame(U, simCOPmicro(U, cop=GHcop, para=G, reflect="grave") )
taildepCOP(cop=GHcop, para=G) # lambdaL = 2e-05; lambdaU = 0.68224
```

The following code example will verify that the simulations produce values of $U$ and $V$ that are consistent with the empirical copula (EMPIRcop) results as well as consistent with the variable reflections provided through the COP interface. Notice the combinations of nonexceedance and exceedance probabilities blended so that the two returned values for the four different copulas are numerically congruent.

```r
c(EMPIRcop(u, v, para=UV ), COP(u, v, cop=GHcop, para=G, reflect="cop" ))
c(EMPIRcop(up,vp, para=sUsV), COP(up, vp, cop=GHcop, para=G, reflect="surv" ))
c(EMPIRcop(up,v, para=sUV ), COP(up, v, cop=GHcop, para=G, reflect="acute") )
c(EMPIRcop(u, vp, para=UsV ), COP(u, vp, cop=GHcop, para=G, reflect="grave") )
```

The user can verify the reflections graphically using code such as this

```r
xlab <- "PROBABILITY IN U"; ylab <- "PROBABILITY IN V"
layout(matrix(c(1,2,3,4), 2, 2, byrow = TRUE));
```
in which inspection of tails exhibiting the dependency is readily seen on the four plots: upper right
tail dependency (no reflection), upper left tail dependency (horizontal reflection), lower right tail
dependency (vertical reflection), and lower left tail dependency (double reflection). It is important
to stress that these descriptions and graphical depictions of single tail dependency are specific to the
GH(u, v; Θ) copula chosen for the demonstration.

Author(s)
W.H. Asquith

References

See Also
simCOP

Examples

```
simCOPmicro(runif(1), cop=W ) # Frechet lower-bound copula
simCOPmicro(runif(1), cop=P ) # Independence copula
simCOPmicro(runif(1), cop=M ) # Frechet upper-bound copula
simCOPmicro(runif(1), cop=PSP) # The PSP copula
```

## Not run:
# Now let us get more complicated and mix two Plackett copulas together using the
# composite2COP as a "compositor." The parameter argument becomes more complex, but is
# passed as shown into composite2COP.
para <- list(cop1=PLACKETTcop,cop2=PLACKETTcop, alpha=0.3,beta=0.5, para1=0.1,para2=50)
simCOPmicro(runif(5), cop=composite2COP, para=para) #
## End(Not run)

## Not run:
# Now let us implement "our" own version of features of simCOP() but using
# the micro version to manually create just the simulation vector of V.
U <- runif(1500)
UV <- data.frame(U, simCOPmicro(U, cop=N4212cop, para=4))
plot(UV, xlab="PROBABILITY IN U", ylab="PROBABILITY IN V", pch=3, col=2) #
## End(Not run)
Description

Kiriliouk et al. (2016, pp. 360–364) describe estimation of the spectral measure of bivariate data. Standardize the bivariate data as $X^*$ and $Y^*$ as in \texttt{psenolar} and select a “large” value for the pseudo-polar radius $S_f$ for nonexceedance probability $f$. Estimate the spectral measure $H(w)$, which is the limiting distribution of the pseudo-polar angle component $W$ given that the corresponding radial component $S$ is large:

$$\Pr[W \in \cdot | S > S_f] \to H(w) \text{ as } S_f \to \infty.$$ 

So, $H(w)$ is the cumulative distribution function of the spectral measure for angle $w \in (0, 1)$. The $S_f$ can be specified by a nonexceedance probability $F$ for $S_f(F)$.

The estimation proceeds as follows:

**Step 1:** Convert the bivariate data $(X_i, Y_i)$ into $(\hat{S}_i, \hat{W}_i)$ by \texttt{psenolar} and set the threshold $S_f$ according to “$n/k$” (this part involving $k$ does not make sense in Kiriliouk et al. (2016)) where for present implementation in \texttt{copBasic} the $S_f$ given the $f$ by the user is based on the empirical distribution of the $\hat{S}_i$. The empirical distribution is estimated by the Bernstein empirical distribution function from the \texttt{lmomco} package.

**Step 2:** Let $I_n$ denote the set of indices that correspond to the observations when $\hat{S}_i \geq S_f$ and compute $N_n$ as the cardinality of $N_n = |I_n|$, which simply means the length of the vector $I_n$.

**Step 3:** Use the maximum Euclidean likelihood estimator, which is the third of three methods mentioned by Kiriliouk et al. (2016):

$$\hat{H}_3(w) = \sum_{i \in I_n} \hat{p}_{3,i} \times 1[\hat{W}_i \leq w],$$

where $1[\cdot]$ is an indicator function that is only triggered if \texttt{smooth=FALSE}, and following the notation of Kiriliouk et al. (2016), the “3” represents maximum Euclidean likelihood estimation. The $\hat{p}_{3,i}$ are the weights

$$\hat{p}_{3,i} = \frac{1}{N_n} \left[ 1 - \left( \frac{W - \hat{W}_i}{1 - \hat{W}_i} \right)^2 S_W^{-2} \left( \hat{W}_i - W \right) \right],$$

where $W$ is the sample mean and $S_W^2$ is the sample variance of $\hat{W}_i$

$$W = \frac{1}{N_n} \sum_{i \in I_n} \hat{W}_i \quad \text{and} \quad S_W^2 = \frac{1}{N_n - 1} \sum_{i \in I_n} (\hat{W}_i - W)^2,$$

where Kiriliouk et al. (2016, p. 363) do not show the $N_n - 1$ in the denominator for the variance but \texttt{copBasic} uses it because those authors state that the sample variance is used.

**Step 4:** A smoothed version of $\hat{H}_3(w)$ is optionally available by

$$\tilde{H}_3(w) = \sum_{i \in I_n} \hat{p}_{3,i} \times \mathcal{B}(w; \hat{W}_i \nu, (1 - \hat{W}_i) \nu),$$
where $B(x; p, q)$ is the cumulative distribution function of the Beta distribution for $p, q > 0$ and where $\nu > 0$ is a smoothing parameter that can be optimized by cross validation.

**Step 5:** The spectral density lastly can be computed optionally as

$$\hat{h}_3(w) = \sum_{i \in I_n} \hat{p}_{3,i} \times \beta(w; \hat{W}_i \nu, (1 - \hat{W}_i) \nu)$$

where $\beta(x; p, q)$ is the probability density function (pdf) of the Beta distribution. Readers are alerted to the absence of the $1[\cdot]$ indicator function in the definitions of $\hat{H}_3(w)$ and $\hat{h}_3(w)$. This is correct and matches Kiriliouk et al. (2016, eqs. 17.21 and 17.22) though this author was confused for a day or so by the indicator function in what is purported to be the core definition of $\hat{H}_l(w)$ where $l = 3$ in Kiriliouk et al. (2016, eq. 17.21 and 17.17).

**Usage**

```r
spectralmeas(u, v=NULL, w=NULL, f=0.90, snv=FALSE, smooth=FALSE, nu=100, pdf=FALSE, ...)
```

**Arguments**

- **u** Nonexceedance probability $u$ in the $X$ direction (actually the ranks are used so this can be a real-value argument as well);
- **v** Nonexceedance probability $v$ in the $Y$ direction (actually the ranks are used so this can be a real-value argument as well) and if NULL then $u$ is treated as a two column R data.frame;
- **w** A vector of polar angle values $W \in [0, 1]$ on which to compute the $H(w)$;
- **f** The nonexceedance probability $F(S_f)$ of the pseudo-polar radius in `psepolar`;
- **snv** Return the standard normal variate of the $H$ by the well-known transform through the quantile function of the standard normal, `qnorm()`;
- **smooth** A logical to return $\hat{H}_3(w)$ instead of $H_3(w)$;
- **nu** The $\nu > 0$ smoothing parameter;
- **pdf** A logical to return the smoothed probability density $\tilde{h}_3(w)$. If pdf=TRUE, then internally smooth=TRUE will be set; and
- **...** Additional arguments to pass to the `psepolar` function.

**Value**

An R vector of $H_3(w)$, $\hat{H}_3(w)$, or $\tilde{h}_3(w)$ is returned.

**Note**

The purpose of this section is to describe a CPU-intensive study of goodness-of-fit between a Gumbel–Hougaard copula ($\text{GHcop}$, $\text{GH}(u, v; \Theta_1)$) parent and a fitted Hüsler–Reiss copula ($\text{HRcop}$, $\text{HR}(u, v; \Theta_2)$). Both of these copulas are extreme values and are somewhat similar to each other, so sample sizes necessary for detection of differences should be large. A two-sided Kolmogorov–Smirnov tests (KS test, `ks.test()`) is used to measure significance in the differences between the estimated spectral measure distributions at $f = 0.90$ (the 90th percentile, $F(S_f)$) into the right tail.
The true copula will be the \( \text{GH}(\Theta_1) \) having parameter \( \Theta_1 = 3.3 \). The number of simulations per sample size \( n \in \text{seq}(50,1000,\,\text{by}=25) \) is \( \text{nsim} = 500 \). For each sample size, a sample from the true parent is drawn, and a \( \text{HR}(\Theta_2) \) fit by maximum likelihood (\text{mleCOP}). The two spectral measure distributions \( (\hat{H}_{\text{GH}}(w), \text{Htru}) \) and \( (\hat{H}_{\text{HR}}(w), \text{Hfit}) \) are estimated for a uniform variate of the angle \( W \) having length equal to the applicable sample size. The Kolmogorov–Smirnov (KS) test is made between \( \text{Htru} \) and \( \text{Hfit} \), and number of p-values less than the \( \beta = 0.05 \) (Type II error because alternative hypothesis is rigged as true) and simulation count are returned and written in file \text{Results.txt}. The sample sizes initially are small and traps of NaN (abandonment of a simulation run) are made. These traps are needed when the empirical distribution of \( \text{Htru} \) or \( \text{Hfit} \) degenerates.

\[
\text{Results} \leftarrow \text{NULL}
\text{true.par} \leftarrow 3.3; \quad \text{true.cop} \leftarrow \text{GHcop}; \quad \text{fit.cop} \leftarrow \text{HRcop}; \quad \text{search} \leftarrow \text{c}(0,100)
\text{nsim} \leftarrow 20000; \quad \text{first.time} \leftarrow \text{TRUE}; \quad \text{f} \leftarrow 0.90; \quad \beta \leftarrow 0.05
\text{ns} \leftarrow \text{c(seq(100,1000, by=50), 1250, 1500, 1750, 2000)}
\text{for}(n \text{ in } \text{ns}) {
\quad W \leftarrow \text{sort}(&\text{runif}(n)); \quad \text{PV} \leftarrow \text{vector}(\text{mode}="\text{numeric})
\quad \text{for}(i \text{ in } 1:(\text{nsim}/(n/2))) {
\quad \quad \text{UV} \leftarrow \text{simCOP}(n=n, \text{cop=}\text{true.cop}, \text{para=}\text{true.par}, \text{graphics}=&\text{FALSE})
\quad \quad \text{fit.par} \leftarrow \text{mleCOP}(\text{UV}, \text{cop}=&\text{fit.cop}, \text{interval}=&\text{search})\\\text{spectralmeas}(\text{UV}, \quad \text{w}=&W, \text{bound.type}="\text{Carv}'', \text{f}=&f)
\quad \quad \text{Htru} \leftarrow \quad \text{spectralmeas}(\text{UVfit}, \quad \text{w}=&W, \text{bound.type}="\text{Carv}'', \text{f}=&f)
\quad \quad \text{if}(\text{length}(\text{Htru}[!\text{is.nan}(\text{Htru})]) \neq \text{length}(\text{Hfit}[!\text{is.nan}(\text{Hfit})]) \quad \text{length}(\text{Htru}[!\text{is.nan}(\text{Htru})]) \neq 0 \quad \text{length}(\text{Hfit}[!\text{is.nan}(\text{Hfit})]) \neq 0) {
\quad \quad \quad \quad \text{PV}[i] \leftarrow \text{NA}; \quad \text{next}
\quad \quad \quad \quad \text{# suppressWarnings()} \text{to silence ties warnings from \text{ks.test}()}
\quad \quad \text{KS} \leftarrow \text{suppressWarnings( stats::ks.test(\text{Htru}, \text{Hfit})\text{p.value} )}
\quad \quad \text{#plot(\text{FF}, \text{H}, \text{type}="\text{l}''); \quad \text{lines}(\text{FF}, \text{Hfit}, \text{col}=2); \quad \text{mtext}(\text{KS})}
\quad \quad \text{message("","i, \text{appendLF}=&FALSE)}
\quad \quad \text{PV}[i] \leftarrow \text{KS}
\quad \quad \}\text{message("","n)}
\quad \text{zz} \leftarrow \text{data.frame}(\text{SampleSize}=&n, \text{NumPVle0p05}=&\text{sum}(\text{PV}[!\text{is.na}(\text{PV})] \leq \beta),
\quad \quad \text{SimulationCount}=&\text{length}(\text{PV}[!\text{is.na}(\text{PV})])
\quad \text{if}(\text{first.time}) \quad \text{\{ Results} \leftarrow \text{zz}; \quad \text{first.time} \leftarrow \text{FALSE}; \quad \text{next }\}
\quad \text{Results} \leftarrow \text{rbind}(\text{Results}, \text{zz})
\quad \}
\quad \text{plot(Results$SampleSize, 100*Results$NumPVle0p05/Results$SimulationCount,}
\quad \quad \text{type}="\text{b}'', \quad \text{cex}=1.1, \quad \text{xlab}="\text{Sample size}'',
\quad \quad \text{ylab}="\text{Percent simulations with p-value < 0.05}'")
\]

The \text{Results} show a general increase in the counts of p-value \( \leq 0.05 \) as sample size increases. There is variation of course and increasing \( \text{nsim} \) would smooth that considerably. The results show for \( n \approx 1,000 \) that the detection of statistically significant differences for extremal \( F(S_f) = 0.90 \) dependency between the \( \text{GH}(\Theta_1=3.3) \) and \( \text{HR}(\Theta_2) \) are detected at the error rate implied by the specified \( \beta = 0.05 \).

This range in sample size can be compared to the Kullback–Leibler sample size \( (n_{fg}) \):
UV <- simCOP(n=10000, cop=true.cop, para=true.par, graphics=FALSE)
fit.par <- mleCOP(UV, cop=fit.cop, interval=search)$para
kullCOP(cop1=true.cop, para1=true.par,
        cop2=fit.cop, para2=fit.par)$KL.sample.size
# The Kullback-Leibler (integer) sample size for detection of differences at
# alpha=0.05 are n_fg = (742, 809, 815, 826, 915, 973, 1203) for seven runs
# Do more to see variation.

where the Kullback-Leibler approach is to measure density departures across the whole $I^2$ domain
as opposed to extremal dependency in the right tail as does the spectral measure.

Different runs of the above code will result in different $n_{fg}$ in part because of simulation differences
internal to kullCOP but also because the $\Theta_2$ has its own slight variation in its fit to the large sample
simulation ($n = 10,000$) of the parent. However, it seems that $n_{fg} \approx 900$ will be on the order of
the $n$ for which the KS test on the spectral measure determines statistical significance with similar
error rate.

Now if the aforementioned simulation run is repeated for $F(S_f) = 0.95$ or $f=0.95$, the $n_{fg}$ ob-
viously remains unchanged at about 900 but the $n$ for which the error rate is about $\beta = 0.05$ is
$n \approx 600$. This sample size is clearly smaller than before and smaller than $n_{fg}$, therefore, the analy-
sis of the empirical spectral measure deeper into the tail $F(S_f) = 0.95$ requires a smaller sample
size to distinguish between the two copula. Though the analysis does not address the question as
to whether one or both copula are adequate for the problem at hand. For a final comparison, if the
aforementioned simulation run is repeated for $F(S_f) = 0.80$ or $f=0.80$, then the $n$ for which the
error rate is about $\beta = 0.05$ is $n \approx 1,700$. Thus as analysis is made further away from the tail
into the center of the distribution, the sample size to distinguish between these two similar copula
increases substantially.

Author(s)
William Asquith <william.asquith@ttu.edu>

References
Kiriliouk, Anna, Segers, Johan, Warchol, Michal, 2016, Nonparameteric estimation of extremal
dependence: in Extreme Value Modeling and Risk Analysis, D.K. Dey and Jun Yan eds., Boca

See Also
psepolar, stabtaildepf

Examples
## Not run:
UV <- simCOP(n=500, cop=HRcop, para=1.3, graphics=FALSE)
W <- seq(0,1,by=0.005)
Hu <- spectralmeas(UV, w=W)
Hs <- spectralmeas(UV, w=W, smooth=TRUE, nu=100)
plot(W,Hu, type="l", ylab="Spectral Measure H", xlab="Angle")
lines(W, Hs, col=2)
### End(Not run)

```
#' @name GAUScop
#' @description A copula function for estimating stable tail dependence.
#' @details
#' Kiriliouk et al. (2016, pp. 364–366) describe a technique for estimation of an empirical stable tail dependence function for a random sample. The function is defined as
#' \[
#' \hat{l}(x,y) = \frac{1}{k} \sum_{i=1}^{n} 1[R_{i,x,n} > n + 1 - k x \text{ or } R_{i,y,n} > n + 1 - k y],
#' \]
#' where \(1[\cdot]\) is an indicator function, \(R\) denotes the rank() of the elements and \(k \in [1, \ldots, n]\) and \(k\) is intended to be “large enough” that \(\hat{l}(x, y)\) has converged to a limit.
```
The “Capéraà–Fougères smooth” of the empirical stable tail dependence function is defined for a coordinate pair $(x, y)$ as

$$\hat{l}_{CP}(x, y) = 2 \sum_{i \in I_n} \hat{p}_{3,i} \times \max[\hat{W}_ix, (1 - \hat{W}_i)y],$$

where $\hat{p}_{3,i}$ are the weights for the maximum Euclidean likelihood estimator (see spectralmeas) and $\hat{W}_i$ are the pseudo-polar angles (see spectralmeas) for the index set $I_n$ defined by $I_n = \{i = 1, \ldots, n : \hat{S}_i > \hat{S}_{(k+1)}\}$, where $\hat{S}_{(k+1)}$ denotes the $(k+1)$-th largest observation of the pseudo-polar radii $\hat{S}_i$ where the cardinality of $I_n$ is exactly $k$ elements long. (Tentatively, then this definition of $I_n$ is ever so slightly different than in spectralmeas.) Lastly, note the multiplier of 2 on the smooth form, and this multiplier is missing in Kiriliouk et al. (2016, p. 365) but shown in Kiriliouk et al. (2016, eq. 17.14, p. 360). Numerical experiments indicate that the 2 is needed for $\hat{l}_{CP}(x, y)$ but evidently not in $\hat{l}(x, y)$.

The visualization of $l(x, y)$ commences by setting a constant ($c > 0$) as $c_i \in 0.2, 0.4, 0.6, 0.8$ (say). The $y$ are solved for $x \in [0, \ldots, c_i]$ through the $l(x, y)$ for each of the $c_i$. Each solution set constitutes a level set for the stable tail dependence function. If the bivariate data have asymptotic independence (to the right), then a level set or the level sets for all the $c$ are equal to the lines $x + y = c$. Conversely, if the bivariate data have asymptotic dependence (to the right), then the level sets will make 90-degree bends for $\max(x, y) = c$.

Usage

```r
stabtaildepf(uv=NULL, xy=NULL, k=function(n) as.integer(0.5*n), levelset=TRUE, ploton=TRUE, title=TRUE, delu=0.01, smooth=FALSE, ...)
```

Arguments

- **uv**: An R data.frame of $u$ and $v$ nonexceedance probabilities in the respective $X$ (horizontal) and $Y$ (vertical) directions. Note that `rank()`s are called on these so strictly speaking this need not be as nonexceedance probabilities. This is not an optional argument;

- **xy**: A vector of the scalar coordinates $(x, y)$, which are “the relative distances to the upper endpoints of [these respective] variables” (Kiriliouk et al., 2016, p. 356). This is a major point of nomenclature confusion. If these are in probability units, they are exceedance probabilities. Though tested for NULL and a warning issued, these can be NULL only if levelset=TRUE but can be set to `xy=NA` if levelset=FALSE and smooth=TRUE (see discussion in Note);

- **k**: The $k$ for both the $l(x, y)$ and $\hat{l}_{CP}$, though the effect of $k$ might not quite be the same for each. The default seems to work fairly well;

- **levelset**: A logical triggering the construction of the level sets for $c = \text{seq}(0.1, 1, by=0.1)$;

- **ploton**: A logical to call the `plot()` function;

- **title**: A logical to trigger a title for the plot if ploton=TRUE;

- **delu**: The $\Delta x$ for a sequence of $x = \text{seq}(0, c, by=\text{delu})$;

- **smooth**: A logical controlling whether $l(x, y)$ or the Capéraà–Fougères smooth function $\hat{l}_{CP}(x, y)$ is used; and

- **...**: Additional arguments to pass.
Value

Varies according to argument settings. In particular, the levelset=TRUE will cause an R list to be returned with the elements having the character string of the respective $c$ values and the each holding a data.frame of the $(x,y)$ coordinates.

Note

This function is also called in a secondary recursion mode. The default levelset=TRUE makes a secondary call with levelset=FALSE to compute the $\hat{l}(x,y)$ for the benefit of the looping on the one-dimensional root to solve for a single $y$ in $\hat{l}(x,y) = c$ given a single $x$. If levelset=TRUE and smooth=TRUE, then a secondary call with smooth=TRUE and levelset=FALSE is made to internally return an R list containing scalar $N_n$ and vectors $\hat{W}_n$ and $\hat{p}_{3,i}$ for similar looping and one-dimensional rooting for $\hat{l}_{CF}(x,y)$.

If levelset=FALSE, then $xy$ is required to hold the $(x,y)$ coordinate pair of interest. A demonstration follows and shows the limiting behavior of a random sample from the N4212cop copula.

```r
n <- 2000 # very CPU intensive this and the next code snippet
UV <- simCOP(n=n, cop=N4212cop, para=pi); k <- 1:n
lhat <- sapply(k, function(j)
  stabtaildepf(xy=c(0.1, 0.1), uv=UV, levelset=FALSE, k=j))
plot(k, lhat, xlab="k in [1,n]", cex=0.8, lwd=0.8, type="b",
ylab="Empirical Stable Tail Dependence Function")
mtext("Empirical function in the 0.10 x 0.10 Pr square (upper left corner)"

The R list that can be used to compute $\hat{l}_{CF}(x,y)$ is retrievable by

```r
x <- 0.1; y <- 0.1; k <- 1:(n-1)
lhatCF <- sapply(k, function(j) {
  Hlis <- stabtaildepf(xy=NA, uv=UV, levelset=FALSE, smooth=TRUE, k=j)
  2*sum(Hlis$p3 * sapply(1:Hlis$Nn, function(i) {
    max(c(Hlis$Wn[i]*x, (1-Hlis$Wn[i])*y)) ))
})
lines(k, lhatCF, col="red")
```

The smooth line (red) of $lhatCF$ is somewhat closer to the limiting behavior of $lhat$, but it is problematic to determine computational consistency. Mathematical consistency with Kiriliouk et al. (2016) appears to be achieved. The Examples section TODO.

Author(s)

William Asquith <william.asquith@ttu.edu>

References


See Also

psepolar,spectralmeas

Examples

```r
## Not run:
UV <- simCOP(n=1200, cop=GLcop, para=2.1) # Galambos copula
tmp1 <- stabtaildepf(UV) # the lines are curves (strong tail dependence)
tmp2 <- stabtaildepf(UV, smooth=TRUE, ploton=FALSE, col="red") #
## End(Not run)
```

statTn

The Tn Statistic of a Fitted Copula to an Empirical Copula

Description

Compute the $T_n(p)$ statistic of Genest et al. (2011) that is defined as

$$ T_n(p) = \sum_{i=1}^{n} |C_n(u_i, v_i) - C_{\Theta_n}(u_i, v_i)|^p, $$

where $C_n(u, v)$ is the empirical copula, $C_{\Theta_n}(u, v)$ is the fitted copula with estimated parameters $\Theta_n$ from the sample of size $n$. The $T_n$ for $p = 2$ is reported by those authors to be of general purpose and overall performance in large scale simulation studies. The extension here for arbitrary exponent $p$ is made for flexibility. Alternatively the definition could be associated with the statistic $T_n(p)^{1/p}$ in terms of a root $1/p$ of the summation as shown above.

The $T_n$ statistic is obviously a form of deviation between the empirical (nonparametric) and parametric fitted copula. The distribution of this statistic through Monte Carlo simulation could be used for inference. The inference is based on that a chosen parametric model is suitably close to the empirical copula. The $T_n(p)$ statistic has an advantage of being relatively straightforward to understand and explain to stakeholders and decision makers, is attractive for being suitable in a wide variety of circumstances, but intuitively might have limited statistical power in some situations for it looks at whole copula structure and not say at tail dependency. Finally, other goodness-of-fits using the squared differences between $C_n(u, v)$ and $C_{\Theta_m}(u, v)$ are aicCOP, bicCOP, and rmseCOP.

Usage

```r
statTn(uv, cop=NULL, para=NULL, p=2, proot=FALSE, ...)
```
Arguments

uv  An \( \mathbb{R} \) two column matrix or data.frame of a sample of nonexceedance probabilities \( u \) and \( v \);

cop  A copula function;

para  Vector of parameters or other data structure, if needed, to pass to the copula;

p  The value for \( p \), and the default follows that of Genest et al. (2011);

proot  A logical controlling whether the \( T_n \) returned be rooted by \( 1/p \), and the default follows that of Genest et al. (2011); and

...  Additional arguments to pass to the copula function and (or) the empirical copula.

Value

The value for \( T_n \) is returned dependent on the specification of \( p \) and whether rooting of the result is desired.

Note

The Examples section shows a simple computation of the \( \hat{T}_n \) statistic for a sample and a fitted copula to that sample. Ideally \( \text{statTn} \) would be wrapped in a Monte Carlo process of fitting the apparent “parent” distribution from the sample data, then for some large replication count, generate \( N \) samples of size \( n \) from the parent and from these samples compute the empirical copula and also fit parameter(s) of the chosen copula and repeatedly solve for \( T_n \). Given a total of \( N \) values of \( T_n \), then the sample \( T_n \) or \( \hat{T}_n \) can be compared to the distribution, and if \( \hat{T}_n \) is greater than say the 95th percentile, then the assumed form of the copula could be rejected.

The \( \text{distTn} \) defined below and is dependent on the \( \text{copBasic.fitpara.beta} \) function can be used to demonstrate concepts. (The process is complex enough that user-level implementation of \( \text{distTn} \) in \( \text{copBasic} \) is not presently (2019) thought appropriate.)

```
"distTn" <- function(n, N=1000, statf=NULL, 
cop=NULL, para=para, interval=NULL, ...) {
  opts <- options(warn=-1)
  message("Estimating Tn distribution: ", appendLF=FALSE)
  Tn <- vector(mode="numeric", N)
  for(i in 1:N) {
    showi <- as.logical(length(grep("0+$", i, perl=TRUE)))
    if(showi) message(i, "-", appendLF=FALSE)
    ruv <- simCOP(n=n, cop=cop, para=para, graphics=FALSE, ...)
    rpara <- copBasic.fitpara.beta(ruv, statf=statf, 
                                 interval=interval, cop=cop)
    Tn[i] <- ifelse(is.na(rpara), NA, statTn(ruv, cop=cop, para=rpara))
  }
  numNA <- length(Tn[is.na(Tn)])
  message("done: Number of failed parameter estimates=", numNA)
  options(opts)
  return(Tn[! is.na(Tn)])
}
```
Let us imagine an $n = 400$ sample size of a Galambos copula ($GL(u, v); GL_{cop}$) and then treat the Plackett copula ($PL(u, v); PL_{cop}$) as the proper (chosen) model. The estimated parameter by the sample Blomqvist Beta of $\hat{\beta}_C = 0.64$ using the blomCOP function called from within copBasic.fitpara.beta is then placed in variable para. The $\hat{\beta}_C$ is not the most efficient estimator but for purposes here, but it is fast. The parameter for the given seed is estimated as about $PL(\Theta=20.75)$.

```r
n <- 400 # sample size
correctCopula <- GLcop; set.seed(1596)
sampleUV <- simCOP(n=n, cop=correctCopula, para=1.9) # a random sample
para.correctCopula <- copBasic.fitpara.beta(uv=sampleUV, statf=blomCOP, interval=c(1,5), cop=correctCopula)
chosenCopula <- PLACKETTcop
para <- copBasic.fitpara.beta(uv=sampleUV, statf=blomCOP, interval=c(.001,200), cop=chosenCopula )
```

Next, compute the sample $\hat{T}_n = 0.063$ from sampleUV. The distribution of the $T_n$ is estimated using the distTn function, and an estimate of the $\hat{T}_n$ p-value is in turn estimated. A large simulation run $N = 1,000$ for a sample of size of $n = 400$ is selected. The distTn function internally will simulated for $N$-replicates from the assumed parent and estimate the parameter. A computation run yields a p-value of approximately 0.01 (depending upon the seed) and is statistically significant at an alpha of 0.05, and therefore, the $PL(\Theta=20.75)$ should be rejected for fitting to these data.

```r
sampleTn <- statTn(sampleUV, cop=chosenCopula, para=para)
Tns <- distTn(n=n, cop=chosenCopula, para=para, interval=c(.001,100), statf=blomCOP)
Tns_pvalue <- 1 - sum(Tns <= sampleTn)/length(Tns) # estimate p-value
```

The demonstration is furthered with a check on the Kullback–Leibler sample size $n_{fg}$ at the 5-percent significance level (alpha = 0.05) by the kullCOP function, which yields 100. Given the parent copula as $GL(\Theta=1.9)$, therefore, it would take approximately 100 samples to distinguish between that copula and a $PL(\Theta=20.75)$ where in this case the fit was through the $\hat{\beta}_C = 0.64$.

```r
ekl <- kullCOP(cop1=correctCopula, para1=1.9, cop2=chosenCopula, para2=para)$KL.sample.size # KL sample size = 100
vuongCOP(sampleUV, cop1=correctCopula, para1=para.correctCopula, cop2=chosenCopula, para2=para)$message
# [1] "Copula 1 has better fit than Copula 2 at 100x(1-alpha) level"
```

The available sample size $n = 400$ is then about four times larger than $n_{fg}$ so the sample size $n$ should be sufficient to judge goodness-of-fit. This is a large value but with the sample variability of $\hat{\beta}_C$, it seems that other measures of association such as Spearman Rho ($rho_{COP}$) or Kendall Tau ($tau_{COP}$) and others cross-referenced therein might be preferable.

The prior conclusion is supported by the p-value of the $\hat{T}_n$ being about 0.01, which suggests that the $PL(u, v)$ is not a good model of the available sample data in sampleUV. Lastly, these judgments are consistent with the Vuoug Procedure performed by the vuongCOP function, which reports at the 5-percent significance level that "copula number 1"—in this case, the $GL(u, v)$—has the better fit, and this is obviously consistent with the problem setup because the random sample for investigation was drawn from the Galambos coupla (the parent form).
surCOP

Author(s)
W.H. Asquith

References

See Also
aicCOP, bicCOP, rmseCOP, vuongCOP, kullCOP

Examples
## Not run:
# Example here is just for Tn. For the example below, the PSP copula is quite different
# from the Gumbel-Hougaard copula and thus, the hatTn would be expected to be different
# from those of the Gumbel-Hougaard and certainly not too near to zero.
sampleUV <- simCOP(n=60, cop=PSP, graphics=FALSE) # random sample
hatTau <- cor(sampleUV$U, sampleUV$V, method="kendall") # Kendall Tau
hatTn <- statTn(sampleUV, cop=GHcop, para=GHcop(tau=hatTau)$para,
                ctype="bernstein", bernprogress=TRUE)
# approximate range 0.0141 -- 0.030
# hatTn in this case is by itself is somewhat uninformative and requires
# Monte Carlo to put an individual value into context.
## End(Not run)

surCOP

The Survival Copula

Description
Compute the survival copula from a copula (Nelsen, 2006, pp. 32–34), which is defined as

\[
\hat{C}(1 - u, 1 - v) = \hat{C}(u', v') = \Pr[U > u, V > v] = u' + v' - 1 + \hat{C}(1 - u', 1 - v'),
\]

where \( u' \) and \( v' \) are exceedance probabilities and \( C(u, v) \) is the copula (COP). The survival copula is a reflection of both \( U \) and \( V \).

The survival copula is an expression of the joint probability that both \( U > v \) and \( U > v \) when the arguments \( a \) and \( b \) to \( C(a, b) \) are exceedance probabilities as shown. This is unlike a copula that has \( U \leq u \) and \( V \leq v \) for nonexceedance probabilities \( u \) and \( v \). Alternatively, the joint probability that both \( U > u \) and \( V > v \) can be solved using just the copula \( 1 - u - v + C(u, v) \), as shown below where the arguments to \( C(u, v) \) are nonexceedance probabilities. The later formula is the joint survival function \( C(u, v) \) (surfuncCOP) defined for a copula (Nelsen, 2006, p. 33) as

\[
C(u, v) = \Pr[U > u, V > v] = 1 - u - v + C(u, v).
\]

Users are directed to the collective documentation in COP and simCOPmicro for more details on copula reflection.
Usage

\texttt{surCOP(u, v, cop=NULL, para=NULL, exceedance=TRUE, \ldots)}

Arguments

\begin{itemize}
  \item \texttt{u}  
    \quad Exceedance probability \( u' = 1 - u \) (\textit{u} nonexceedance based on exceedance) in the \( X \) direction;
  \item \texttt{v}  
    \quad Exceedance probability \( v' = 1 - v \) (\textit{v} nonexceedance based on exceedance) in the \( Y \) direction;
  \item \texttt{cop}  
    \quad A copula function;
  \item \texttt{para}  
    \quad Vector of parameters or other data structure, if needed, to pass to the copula;
  \item \texttt{exceedance}  
    \quad A logical affirming whether \textit{u} and \textit{v} are really in exceedance probability or not? If FALSE, then the complements of the two are made internally and the nonexceedances can thus be passed; and
  \item \ldots  
    \quad Additional arguments to pass (such as parameters, if needed, for the copula in the form of an \texttt{R} list).
\end{itemize}

Value

Value(s) for the survival copula are returned.

Note

The author (Asquith) finds the use of exceedance probabilities delicate in regards to Nelsen’s notation. This function and \texttt{coCOP} have the exceedance argument to serve as a reminder that the survival copula as usually defined uses exceedance probabilities as its arguments.

Author(s)

W.H. Asquith

References


See Also

\texttt{COP, coCOP, duCOP, surfuncCOP, simCOPmicro}

Examples

\begin{verbatim}
  u <- 0.26; v <- 0.55  # nonexceedance probabilities
  up <- 1 - u; vp <- 1 - v  # exceedance probabilities
  surCOP(up, vp, cop=PSP, exceedance=TRUE)  # 0.4043928
  surCOP(u, v, cop=PSP, exceedance=FALSE)  # 0.4043928 (same because of symmetry)
  surfuncCOP(u, v, cop=PSP)  # 0.4043928
  # All three examples show joint prob. that U > u and V > v.

  ## Not run:
\end{verbatim}
# A survival copula is a copula so it increases to the upper right with increasing exceedance probabilities. Let us show that by hacking the surCOP function into a copula for feeding back into the algorithmic framework of copBasic.

```r
UsersCop <- function(u,v, para=NULL) {
  afunc <- function(u,v, theta=para) { surCOP(u, v, cop=N4212cop, para=theta)}
  return(asCOP(u,v, f=afunc)) }
```

```r
image(gridCOP(cop=UsersCop, para=1.15), col=terrain.colors(20),
      xlab="U, EXCEEDANCE PROBABILITY", ylab="V, EXCEEDANCE PROBABILITY") #
## End(Not run)
```

The Joint Survival Function

**Description**

Compute the joint survival function for a copula (Nelsen, 2006, p. 33), which is defined as

\[
C(u,v) = \Pr[U > u, V > v] = 1 - u - v + C(u,v) = \hat{C}(1 - u, 1 - v),
\]

where \( \hat{C}(u', v') \) is the survival copula (surCOP), which is defined by

\[
\hat{C}(u', v') = \Pr[U > u, V > v] = u' + v' - 1 + C(1 - u', 1 - v').
\]

Although the joint survival function is an expression of the probability that both \( U > v \) and \( U > v \), \( \bar{C}(u, v) \) is not a copula.

**Usage**

```r
surfuncCOP(u, v, cop=NULL, para=NULL, ...)
```

**Arguments**

- **u**: Nonexceedance probability \( u \) in the \( X \) direction;
- **v**: Nonexceedance probability \( v \) in the \( Y \) direction;
- **cop**: A copula function;
- **para**: Vector of parameters or other data structure, if needed, to pass to the copula; and
- **...**: Additional arguments to pass (such as parameters, if needed, for the copula in the form of a list).

**Value**

Value(s) for the joint survival function are returned.

**Author(s)**

W.H. Asquith
tailconCOP

The Tail Concentration Function of a Copula

Description

Compute the tail concentration function \( q_{C} \) of a copula \( C(u, v) \) (COP) or diagonal (diagCOP) of a copula \( \delta_{C}(t) = C(t, t) \) according to Durante and Semp (2015, p. 74):

\[
q_{C}(t) = \frac{C(t,t)}{t} \cdot 1_{[0,0.5]} + \frac{1 - 2t + C(t,t)}{1 - t} \cdot 1_{[0.5,1]}
\]

or

References


See Also

surCOP

Examples

```r
"MOcop.formula" <- function(u,v, par=para, ...) {
    alpha <- para[1]; beta <- para[2]; return(min(v*u^(1-alpha), u*v^(1-beta)))
}
"MOcop" <- function(u,v, ...) { asCOP(u,v, f=MOcop.formula, ...) }

u <- 0.2; v <- 0.75; ab <- c(1.5, 0.3)
# U **and** V are less than or equal to a threshold +
# U **or** V are less than or equal to a threshold
surCOP(1-u,1-v, cop=MOcop, para=ab) + duCOP(u,v, cop=MOcop, para=ab) # UNITY
surfuncCOP(u,v, cop=MOcop, para=ab) + duCOP(u,v, cop=MOcop, para=ab) # UNITY

## Not run:
# The joint survival function is not a copula. So, it does not increases to the upper right with increasing exceedance probabilities. Let us show that by hacking the surCOP function into a copula for feeding back into the algorithmic framework of copBasic.

UsersCop <- function(u,v, para=NULL) {
    afunc <- function(u,v, theta=para) { surfuncCOP(u, v, cop=N4212cop, para=theta) }
    return(asCOP(u,v, f=afunc))
}

image(gridCOP(cop=UsersCop, para=1.15), col=terrain.colors(20),
      xlab="U, NONEXCEEDANCE PROBABILITY", ylab="V, NONEXCEEDANCE PROBABILITY")

## End(Not run)

## Not run:
# Conditional return period (Salvadori et al., 2007, p. 159)
UV <- simCOP(n=100000, cop=PLACKETTcop, para=5, graphics=FALSE)

u <- 0.5; v <- 0.99; cd <- UV$V[UV$U > u]
by.counting <- length(cd[cd > v]) / length(cd) # 0.0172
by.theo <- surfuncCOP(u,v, cop=PLACKETTcop, para=5) / (1-u) # 0.0166
by.ec <- surfuncCOP(u,v, cop=EMPIRcop, para=UV) / (1-u) # 0.0189
print(1/by.theo) # conditional return period for V > 0.99 given U > 0.5

## End(Not run)
```
\[ q_C(t) = \frac{\delta_C(t)}{t} \cdot 1_{[0,0.5]} + \frac{1 - 2t + \delta_C(t)}{1 - t} \cdot 1_{[0.5,1]}, \]

where \( t \) is a nonexceedance probability on the margins and \( 1(.) \) is an \textit{indicator function} scoring 1 if condition is true otherwise zero on what interval \( t \) resides: \( t \in [0,0.5) \) or \( t \in [0.5,1] \). The \( q_C(t; M) = 1 \) for all \( t \) for the \( M \) copula and \( q_C(t; W) = 0 \) for all \( t \) for the \( W \) copula. Lastly, the function is related to the \textit{Blomqvist Beta} \((\beta_C; blomCOP)\) by

\[ q_C(0.5) = (1 + \beta_C)/2, \]

where \( \beta_C = 4C(0.5,0.5) - 1 \). Lastly, the \( q_C(t) \) for \( 0, 1 = t \) is \( \text{NaN} \) and no provision for alternative return is made. Readers are asked to note some of the mathematical similarity in this function to Blomqvist Betas in \texttt{blomCOPss} in regards to tail dependency.

**Usage**

\[ \text{tailconCOP}(t, \text{cop=NULL, para=NULL, ...}) \]

**Arguments**

- \( t \) Nonexceedance probabilities \( t; \)
- \( \text{cop} \) A copula function;
- \( \text{para} \) Vector of parameters or other data structure, if needed, to pass to the copula; and
- \( \ldots \) Additional arguments to pass to the copula function.

**Value**

Value(s) for \( q_C \) are returned.

**Author(s)**

W.H. Asquith

**References**


**See Also**

\texttt{taildepCOP, tailordCOP}

**Examples**

\[ \text{tailconCOP}(0.5, \text{cop=PSP}) == (1 + \text{blomCOP(cop=PSP)}) / 2 \text{ # TRUE} \]
The Lower- and Upper-Tail Dependency Parameters of a Copula

Description

Compute the lower- and upper-tail dependency parameters (if they exist), respectively, of a copula according to Nelsen (2006, pp. 214–215). Graphical confirmation of the computations is important, and therefore, the function can also generate a plot. The dependency parameters are expressions of conditional probability that $Y$ is greater than the $100 \times t$-th percentile of its distribution $G$ given that $X$ is greater than the $100 \times t$-th percentile of its distribution $F$ as $t$ approaches unity. Specifics in terms of quantile functions $G\left( -1 \right)(t) = y(t)$ and $F\left( -1 \right)(t) = x(t)$ follow.

The lower-tail dependence parameter $\lambda_L$ is defined as

$$\lambda_L = \lim_{t \to 0^+} \Pr\{Y \leq y(t) \mid X \leq x(t)\},$$

and the upper-tail dependence parameter $\lambda_U$ with reversed inequalities is defined as

$$\lambda_U = \lim_{t \to 1^-} \Pr\{Y > y(t) \mid X > x(t)\}.$$

Nelsen (2006, p. 214) also notes that both $\lambda_L$ and $\lambda_U$ are nonparametric and depend only on the copula of $X$ and $Y$, and Nelsen shows that each can be computed if the above limits exist as follows:

$$\lambda_L = \lim_{t \to 0^+} \frac{C(t,t)}{t} = \delta'_C(0^+)$$

and

$$\lambda_U = \lim_{t \to 1^-} \frac{1 - 2t - C(t,t)}{1 - t} = 2 - \lim_{t \to 1^-} \frac{1 - C(t,t)}{1 - t} = 2 - \delta'_C(1^-),$$

where $\delta'_C(t)$ is the derivative of the diagonal of the copula. Multiple presentations are shown because algebraic variants are shown across the literature.

If $\lambda_L \in (0, 1]$, then $C$ has lower-tail dependence but if $\lambda_L = 0$, then $C$ has no lower-tail dependence. Likewise, if $\lambda_U \in (0, 1]$, then $C$ has upper-tail dependence but if $\lambda_U = 0$, then $C$ has no upper-tail dependence.

Usage

```
taildepCOP(cop=NULL, para=NULL, tol=1e-6, divisor=2, plot=FALSE, ylim=NULL, verbose=FALSE, ...)```

Arguments

- `cop`: A copula function;
- `para`: Vector of parameters or other data structure, if needed, to pass to the copula;
- `tol`: A tolerance on convergence;
- `divisor`: The divisor on the incremental reductions towards $0^+$ and $0^-$ by the algorithm;
plot A logical plotting a diagnostic plot of the diagonal derivatives and label the limits;

ylim Optional vertical limits if the plot is turned on. Although the dependence parameters are bounded as described above, numerical stability can be a problem. Stability is especially a problem if an empirical copula is being used; therefore, the bounds of the plot are left open unless the user locks them down with this argument;

verbose Show incremental progress; and

... Additional arguments to pass to the copula function.

Value

An R list is returned.

lambdaL The rounded value of \( \lambda_C^L \);

lambdaU The rounded value of \( \lambda_C^U \);

source An attribute identifying the computational source: “taildepCOP”.

Note

IMPLEMENTED ALGORITHM—The algorithm implemented for taildepCOP is based on halves (or alternatives by the setting of divisor argument) and uses the copula function (not an analytical or even numeric derivative of the diagonal, \( \delta_C(t) \)). Starting from the median or \( t = 0.5 \), each limit is respectively computed by successive halving (or the setting of argument divisor) of the distance towards \( 0^+ \) and \( 1^- \) and checking the change in computed value against the tolerance tol argument. After the change becomes less than the tolerance, convergence is assumed. Other tests are made for NaN to aid in breaking the successive halvings. The rounding for the numerical results for \( \lambda_C^L \) and \( \lambda_C^U \) is an order of magnitude larger than the tolerance.

Users are encouraged to plot the results and further verify whether the convergence makes sense. The plot produced when plot=TRUE shows the probability \( t \) transformed into standard normal variates by the qnorm() function in R so that the distal reaches of each tail and thus limit are readily seen. The terminal points of each limit computation are shown by a small dot and the letter “L” and “U” also are plotted at the terminal points.

Joe (2014, p. 63) reports that “the empirical measure of tail dependence \( \hat{\lambda}_C^L \) or \( \hat{\lambda}_C^U \) for data does not really exist because of the limit.” Joe (2014) suggests that other sources in the literature are the “best that can be done” (Dobrić and Schmid, 2005; Frahm et al., 2005). Another source of discussion is by Schmidt and Stadtmüller (2006). The results therein are not yet followed up for the copBasic package. Picking up the simulation dealt with extensively in the Note section of vuongCOP, a user might try this:

```r
cop <- simCOP(cop=PSP, n=n, col=8, pch=16, graphics=FALSE)
taildepCOP(cop=EMPIRcop, para=UV, plot=TRUE, divisor=8, ylim=c(0,1))
taildepCOP(cop=PSP) # lower=0.5, upper=0
```

The returned tail dependency parameters are numerically of little importance and in strict terms likely misleading. What should be of interest are the plotted trajectories of the lower and upper
Note: the lower tail wobbles but seems to show stability towards \( \hat{\lambda}_L^C = 0.5 \) and upper-tail line wobbles downward towards \( \hat{\lambda}_U^C = 0 \). These values are, respectively, the tail dependencies of the PSP copula (PSP). A user might try increasing the sample size by an order of magnitude and rerunning the above code. Lastly, Salvadori et al. (2006, pp. 173–175) caution on the difficulties of nonparametric tail dependency estimation. Given objectives of the copBasic package, estimation of \( \hat{\lambda}_L^C \) and \( \hat{\lambda}_U^C \), therefore, is an open development opportunity.

**DEMONSTRATION (Tail Dependence)**—The following example shows a comparison between early code examples by Charpentier (2012) concerning copulas and tail dependence using real-world data. Consider the lossalae data set and the following code requiring the evd package:

```r
library(evd); X <- lossalae # Charpentier (2012)
library(copBasic)
fakeU <- lmomco::pp(X[,1],sort=FALSE,a=0) # Weibull plotting position i/(n+1)
fakeV <- lmomco::pp(X[,2],sort=FALSE,a=0) # Weibull plotting position i/(n+1)
uv <- data.frame(U=fakeU, V=fakeV) # parameter "object" for Empirical copula plot(uv)
TD <- taildepCOP(cop=EMPIRcop, para=uv, divisor=25,
plot=TRUE, ylim=c(0, 7/10))
U <- rank(X[,1])/(nrow(X)+1); V <- rank(X[,2])/(nrow(X)+1)# Charpentier(2012)
Lemp <- function(z) sum((U<=z) & (V<=z)) / sum(U<=z )# Charpentier(2012)
Remp <- function(z) sum((U>=1-z) & (V>=1-z)) / sum(U>=1-z)# Charpentier(2012)
u <- seq(0.001, 0.5, by=.001) # Charpentier(2012)
L <- Vectorize(Lemp)(u); R <- Vectorize(Remp)(rev(u)) # Charpentier(2012)
lines(qnorm(c(u, u + 0.5 - u[1])), c(L,R)) # modified after Charpentier(2012)

The figure that will have been generated shows considerably similarity to that from the algorithms of Charpentier. Now, let us extend the discussion by using the Blomqvist (Schmid–Schmidt) Betas (blomCOPss) that have a formulation permitting lower- and upper-tail dependency parameters in a different manner than the definitions of this documentation for taildepCOP.

```r
dec <- 30 * 1 / (1+nrow(X)) # as few as 30 samples into the tails
psl <- pnorm(seq(0, qnorm( dec), by=-0.005))
psu <- pnorm(seq(0, qnorm(1-dec), by= 0.005))
lines(qnorm(psl),
sapply(psl, function(p) { blomCOPss(as.sample=TRUE, para=uv, ctype="checkerboard", uu=rep(p, 2), vv=c(1,1)) }),
col="darkgreen", lty=1, lwd=2)
lines(qnorm(psu),
sapply(psu, function(p) { blomCOPss(as.sample=TRUE, para=uv, ctype="checkerboard", uu=c(0,0), vv=rep(p, 2)) }),
col="darkgreen", lty=1, lwd=2)
points(0, blomCOP(as.sample=TRUE, para=uv), pch=16, col="magenta", cex=2)
legend("topleft", c("Tail dependency by Blomqvist (Schmid-Schmidt) Betas", "Blomqvist Beta C(1/2, 1/2)"),
"Lower-tail dependency by taildepCOP(), upper-tail dependency by taildepCOP()", "Charpentier (2012)", bty="n", cex=0.9,
lwd=1, lty=1, col=c("red", "blue", "black"))
```
The thick green lines show that the dependency parameters to the left and right are approached along a different trajectory using the definitions in `blomCOPss`. It seems at some stage in analysis that the practitioner will need to decide how deep into the tail the sample will permit for a reliable estimate of the dependency parameters. Blomqvist Beta ($\hat{\beta}_C$) (`blomCOP` is shown as the magenta dot at the median and henceforth from do the trajectories of $\hat{\lambda}_L^C$ and $\hat{\lambda}_U^C$ extend as their $\lim p \to 0^+$. Ultimately, for the data shown in the figure, perhaps the $\hat{\lambda}_L^C = 0.1$ and the $\hat{\lambda}_U^C = 0.3$ and a parametric copula fitted in part to such values.

**Author(s)**

W.H. Asquith

**References**


**See Also**

`COP, tailconCOP, tailordCOP, blomCOPss`

**Examples**

```r
# Plot the tail dependencies by nonexceedance probability for a
# for a positive association Plackett copula and see that both are zero.
taildepCOP(cop=PLACKETTcop, para=3, plot=TRUE)
# So, Plackett has no tail dependency, as Nelsen (2006, p. 215) shows.

## Not run:
"MOcop" <- function(u,v, para=NULL) { # Marshall-Olkin copula
  alpha <- para[1]; beta <- para[2]; return(min(v*u^(1-alpha), u*v^(1-beta)))
} # The results that follow match those reported by Nelsen (2006, p. 215).
taildepCOP(cop=MOcop, para=c(0.4, 0.9)) # LambL = 0, LambU = 0.4 [min(alpha,beta)]
## End(Not run)
```
## Not run:

# Analytical solution to Gumbel-Hougaard copula from the copula package:
copula::lambda(copula::gumbelCopula(3))
# lower upper
# 0.000000 0.740079
# Numerical approximation (see copBasic::GHcop for analytical formula):
as.data.frame(taildepCOP(GHcop, para=3))
# lambdaL lambdaU source
# 1 0.00012 0.74008 taildepCOP
## End(Not run)

## Not run:

# Plot the tail dependencies by nonexceedance probability
# for the PSP copula, which has lower but no upper-tail dependence.
taildepCOP(cop=PSP, para=NULL, plot=TRUE) # LambL=0.5, LambU=0
# which is readily confirmed by simCOP(1000, cop=PSP)
# Nelsen (2006, p. 216) reports that this copula has LambL=1/2 and LambU=0,
# and we get the same results here.

# How about some composited Plackett-Plackett copulas?
# Each has upper- and lower-tail dependence parameters equal to zero.
para <- list( cop1=PLACKETTcop, cop2=PLACKETTcop, alpha=0.9392,
               para1=0.00395, para2=4.67, beta=0.5699)
taildepCOP(cop=composite2COP, para=para, plot=TRUE, verbose=TRUE) #
## End(Not run)

## Not run:

# This next Plackett-Plackett is interesting because at its core it looks
# like it should be both tail dependent like M() but the shapes of the curves
# are quite different from those of M(). This example shows numerical
# instability for the upper tail but not the lower tail. So, we extend the
# example to shown the tail dependency trajectories by blomCOPss(). And again
# it is seen that the lower tail as a stable solution but the upper tail
# has instability at 6 standard deviations into the upper tail.
para <- list( cop1=PLACKETTcop, cop2=PLACKETTcop, alpha=0.0063,
               para1=0.101, para2=4493, beta=0.0167)
taildepCOP(cop=composite2COP, para=para, plot=TRUE)
lsu <- pnorm(seq(-7, 0, by=.01))
psu <- pnorm(seq( 0, 7, by=.01))
lines(qnorm(lsu), sapply(lsu, function(p) {
   blomCOPss(cop=composite2COP, para=para, vv=c(1,1), uu=rep(p, 2) )
}), col="darkgreen", lty=2, lwd=1)
lines(qnorm(psu), sapply(psu, function(p) {
   blomCOPss(cop=composite2COP, para=para, uu=c(0,0), vv=rep(p, 2) )
}), col="darkgreen", lty=2, lwd=1) #
## End(Not run)
Description

Compute the lower- and upper-tail orders (if they exist), respectively, of a copula \( C(u, v) \) according to Joe (2014, pp. 67–70). The tail order is a concept for the strength of dependence in the joint tails of a multivariate distribution. The opposing tails can be compared to assess tail order or reflection symmetry (term by Joe (2014) for Nelsen’s (2006, p. 36) term radial symmetry). Joe (2014) provides extensively analytical details but sufficient for the copBasic package, the tail orders can be numerically explored.

The lower-tail order maybe numerically approximated by

\[
\kappa_C^L = \frac{\log[C(t, t)]}{\log(t)},
\]

for some small positive values of \( t \), and similarly the upper-tail order maybe numerically approximated by

\[
\kappa_C^U = \frac{\log[\hat{C}(t, t)]}{\log(t)},
\]

where \( \hat{C}(u, v) \) is the survival copula (surCOP). Joe (2014) has potentially(?) conflicting notation in the context of the upper-tail order; the term “reflection” is used (p. 67) and “lower tail order of the reflected copula is the same as the upper tail order of the original copula” (p. 69), but Joe (2014, p. 67) only uses the joint survival function (surfuncCOP) in the definition of \( \kappa_C^U \).

As a note, the author of this package was not able to get tailordCOP to function properly for the upper-tail order using the joint survival function as implied on the bottom of Joe (2014, p. 67) and fortunately the fact that “reflection” is used in other contexts and used in analytical examples, the tailordCOP function uses the lower-tail order of the reflection (survival copula). Joe (2014) also defines tail order parameter \( \Psi \) but that seems to be a result of analytics and not implemented in this package. Lastly, the tail orders are extendable into \( d \) dimensions, but only a bivariate (\( d = 2 \)) is provided in copBasic. The tail orders have various classifications for \( \kappa = \kappa_L = \kappa_U \):

- **Intermediate tail dependence** for \( 1 < \kappa < d \) or \( \kappa = 1, \Psi = 0 \);
- **Strong tail dependence** for \( \kappa = 1 \) with \( \Psi > 0 \); and
- **Tail orthant independence or tail quadrant independence** for \( \kappa = d \).

Joe (2014) provides additional properties:

- \( \kappa_L = \kappa_U = d \) for the \( d \)-dimensional independence copula (P; e.g. tailordCOP(cop=P));
- It is not possible for \( \kappa_L < 1 \) or \( \kappa_U < 1 \) but each can be > 1 for a \( C(u, v) \) having some negative dependence (e.g. tailordCOP(cop=PLACKETTcop, para=0.2); see PLACKETTcop); and
- For the bivariate Fréchet–Hoeffding lower-bound copula (W; countermonotonicity copula) the \( \kappa_L = \kappa_U \) and can be considered \(+\infty\). (A special trap in the tailordCOP provides consistency on W but does not test that the copula is actually that function itself.)

Usage

\[
\text{tailordCOP(cop=NULL, para=NULL, tol=1e-6, plot=FALSE, verbose=FALSE, ...)}
\]
Arguments

cop A copula function;
para Vector of parameters or other data structure, if needed, to pass to the copula;
tol A tolerance on convergence;
plot A logical plotting a diagnostic plot of the diagonal derivatives and label the limits;
verbose Show incremental progress; and
... Additional arguments to pass to the copula function.

Value

An \texttt{R} list is returned.

kappaL The rounded value of $\kappa_L^C$;
kappaU The rounded value of $\kappa_U^C$;
source An attribute identifying the computational source: “tailordCOP”.

Note

The algorithm implemented for \texttt{tailordCOP} is based on halves (or alternatives by the setting of the divisor argument) and uses the copula function (not an analytical or even numerical derivative of the diagonal, $\delta_C(t)$). Starting from the median or $t = 0.5$, each limit is respectively computed by successive halving of the distance towards $0^+$ and checking the change in computed value against the tolerance \texttt{tol} argument. After the change becomes less than the the tolerance, convergence is assumed. Other tests are made for \texttt{NaN} to aid in breaking the successive halvings. The rounding for the numerical results for $\kappa_U^C$ and $\kappa_L^C$ is an order of magnitude larger than the tolerance.

Users are encouraged to plot the results and further verify whether the convergence makes sense. The plot produced when \texttt{plot=TRUE} shows the probability $t$ transformed into standard normal variates by the \texttt{qnorm()} function in \texttt{R} so that the distal reaches of each tail and thus limit are readily seen. The terminal points of each limit computation are shown by a small dot, and the letter “L” and “U” also are plotted at the terminal points.

Author(s)

W.H. Asquith

References


See Also

\texttt{COP, tailconCOP, taildepCOP}
Examples

```r
# Not run:
# Joe (2014, p. 5) names MTCJ = Mardia-Takahasi-Cook-Johnson copula
"MTCJ" <- function(u,v, para) { (u^(-para) + v^(-para) - 1)^(-1/para) }
# The results that follow match those reported by Joe (2014, p. 69) who
# analytically derives KappaL = 1 and KappaU = 2.
# TAIL ORDER:
tailordCOP(cop=MTCJ, para=3, plot=TRUE) # kappaL = 1.00667, kappaU = 1.96296
# TAIL DEPENDENCY:
taildepCOP(cop=MTCJ, para=3, plot=TRUE) # lambdaL = 0, lambdaU = 0.7937
# Joe (2014) reports lambdaL = 2^(-1/para) = 2^(-1/3) = 0.7937005
## End(Not run)
```

---

**tauCOP**  
*The Kendall Tau and Concordance Function of a Copula*

---

**Description**

Compute the measure of association known as the *Kendall Tau* \( \tau_C \) of a copula \( \tau_C \) according to Nelsen (2006, sec. 5.1.1 and p. 161) by

\[
\tau_C = Q(C, C) = 4 \int \int_{I^2} C(u, v) dC(u, v) - 1,
\]

where \( Q(C, C) \) is a *concordance function* (concordCOP) of a copula with itself. Nelsen (2006, p. 164) reports however that this form is often not amenable to computation when there is a singular component to the copula and that the expression

\[
\tau_C = 1 - 4 \int \int_{I^2} \frac{\delta C(u, v)}{\delta u} \frac{\delta C(u, v)}{\delta v} dudv
\]

is to be preferred. Such an expression hence relies on the partial numerical derivatives of the copula provided by derCOP and derCOP2. The Nelsen (2006) preferred expression is used by the tauCOP function. Nelsen (2006, pp. 175–176) reports that the relation between \( \tau_C \) and \( \rho_C \) (rhoCOP) is \(-1 \leq 3\tau - 2\rho \leq 1\) (see rhoCOP for more details).

Nelsen (2006, pp. 160–161) lists some special identities involving \( Q(C_1, C_2) \):

\[
Q(M, M) = 4 \int_0^1 u du - 1 = 1,
\]

\[
Q(M, \Pi) = 4 \int_0^1 u^2 du - 1 = 1/3,
\]

\[
Q(M, W) = 4 \int_{1/2}^1 (2u - 1) du - 1 = 0,
\]

\[
Q(W, \Pi) = 4 \int_0^1 u(1-u) du - 1 = -1/3,
\]
\[ Q(W, W) = 4 \int_0^1 0 \, du - 1 = -1, \text{ and} \]

\[ Q(\Pi, \Pi) = 4 \int_{I^2} uv \, du \, dv - 1 = 0. \]

Kendall Tau also can be expressed in terms of the *Kendall Function* \( F_K(z); \text{kfuncCOP} \):

\[ \tau_C = 3 - 4 \int_0^1 F_K(t) \, dt, \]

which is readily verified by code shown in *Examples*. This definition might be useful if integration errors are encountered for some arbitrary copula and arbitrary parameter set. In fact, should two attempts (see source code) at dual integration of the partial derivatives occur, the implementation switches over to integration of the Kendall Function (*e.g.* \( \text{tauCOP(cop=N4212cop, para=2)} \)). Note that Durante and Sempi have erroneously dropped the multiplication by “4” as shown above in their definition of \( \tau_C \) as a function of \( F_K(t) \) (Durante and Sempi, 2015, eq. 3.9.4, p. 121).

**Usage**

```r
tauCOP( cop=NULL, para=NULL, 
      cop2=NULL, para2=NULL, as.sample=FALSE, brute=FALSE, delta=0.002, ...)
```

```r
concordCOP(cop=NULL, para=NULL, cop2=NULL, para2=NULL, ...)
```

**Arguments**

- **cop** A copula function;
- **para** Vector of parameters or other data structure, if needed, to pass to the copula;
- **cop2** A second copula function;
- **para2** Vector of parameters or other data structure, if needed, to pass to the second copula;
- **as.sample** A logical controlling whether an optional \( \mathbf{R} \) data.frame in para is used to compute the \( \hat{r} \) by dispatch to \( \text{cor()} \) function in \( \mathbf{R} \) with method = "kendall";
- **brute** Should brute force be used instead of two nested \( \text{integrate()} \) functions in \( \mathbf{R} \) to perform the double integration;
- **delta** The \( du \) and \( dv \) for the brute force integration using brute; and
- **...** Additional arguments to pass on to \( \text{derCOP} \) and \( \text{derCOP2} \).

**Value**

The value for \( \tau_C \) is returned.
**Note**

Although titled for computation of the Kendall Tau, the tauCOP function also is the implementation of the concordance function \( Q(C_1, C_2) \) (see Nelsen (2006, pp. 158–159) when given two different copulas and respective parameters as arguments. The function concordCOP just dispatches to tauCOP. A useful relation is

\[
\int \int_{T^2} C_1(u,v) \, dC_2(u,v) = \frac{1}{2} - \int \int_{T^2} \frac{\delta}{\delta u} C_1(u,v) \frac{\delta}{\delta v} C_2(u,v) \, du \, dv,
\]

where \( C_1(u,v) \) is the first copula and \( C_2(u,v) \) is the second copula.

Nelsen et al. (2001, p. 281) lists several measures of association defined by the concordance function:

1. \( \tau_C = Q(C, C) \) : (Kendall Tau; tauCOP);
2. \( \rho_C = 3 \cdot Q(C, \Pi) \) : (Spearman Rho; rhoCOP);
3. \( \gamma_C = 2 \cdot Q(C, [M + W] / 2) \) : (Gini Gamma; giniCOP); and
4. \( \psi_C = \frac{3}{2} \cdot Q(C, M) - \frac{1}{2} \) : (Spearman Footrule; footCOP).

**Author(s)**

W.H. Asquith

**References**


**See Also**

blomCOP, footCOP, giniCOP, hoefCOP, rhoCOP, wolfCOP, joeskewCOP, uvlmoms, derCOP, derCOP2, kfuncCOP

**Examples**

```r
## Not run:
tauCOP(cop=PSP) # 1/3
# Now compute Kendall Tau via integration of the Kendall Function.
# 3 - 4*integrate(function(t) kfuncCOP(t, cop=PSP), 0, 1)$value # 0.3333314
## End(Not run)

## Not run:
tauCOP(cop=PSP, brute=TRUE) # 0.3306625
# CPU heavy example showing that the dual-integration (fast) results in
# a Kendall Tau that matches a sample version
dotau <- function(n) {
  uv <- simCOP(n=n, cop=PSP, ploton=FALSE, points=FALSE)
  return(cor(uv$U, uv$V, method="kendall"))
}
```
set.seed(817600)
taus <- replicate(100, dotau(100))
tau.sample <- mean(taus); print(tau.sample) # 0.3342034
## End(Not run)
## Not run:
# Nelsen (2006, pp. 160-161, numeric results shown thereine)
# The rational values or integers may be derived analytically.
tauCOP(cop=M, cop2=M) # 1, correct
tauCOP(cop=M, cop2=P) # 1/3, correct
tauCOP(cop=P, cop2=M) # 1/3, correct
tauCOP(cop=M, cop2=W) # 0, correct
tauCOP(cop=W, cop2=M) # throws warning, swaps copulas, == tauCOP(M,W)
tauCOP(cop=W, cop2=P) # throws warning, swaps copulas, approx. -1/3
tauCOP(cop=P, cop2=W) # -1/3, correct
tauCOP(cop=P, cop2=P) # 0, correct
tauCOP(cop=M, cop2=W, brute=TRUE) # 0, correct
## End(Not run)
## Not run:
para <- list(cop1=PLACKETTcop, cop2=PLACKETTcop,
             para1=0.00395, para2=4.67, alpha=0.9392, beta=0.5699)
tauCOP(cop=composite2COP, para=para) # -0.4671213
para <- list(cop1=PLACKETTcop, cop2=PLACKETTcop,
             para1=0.14147, para2=20.96, alpha=0.8411, beta=0.6873)
tauCOP(cop=composite2COP, para=para) # +0.1950727
para <- list(cop1=PLACKETTcop, cop2=PLACKETTcop,
             para1=0.10137, para2=4492.87, alpha=0.0063, beta=0.0167)
# Theoretical attempt fails because para2 is large and thus a singularity
# is emerging and internal copula swapping does not help.
tauCOP(cop=composite2COP, para=para) # fails (0.94+-0.01)
tauCOP(cop=composite2COP, para=para, brute=TRUE) # about 0.94+-0.01
## End(Not run)

tEVcop

The t-EV (Extreme Value) Copula

Description

The t-EV copula (Joe, 2014, p. 189) is a limiting form of the t-copula (multivariate t-distribution):

\[ C_{\rho, \nu}(u, v) = \text{tEV}(u, v; \rho, \nu) = \exp\left(-\left(x + y\right) \times B\left(x/(x + y); \rho, \nu\right)\right), \]

where \( x = -\log(u) \), \( y = -\log(v) \), and letting \( \eta = \sqrt{(\nu + 1)/(1 - \rho^2)} \) define

\[ B(w; \rho, \nu) = w T_{\nu+1}(\eta((1 - w)/\nu)^{1/\nu} - \rho) + (1 - w) T_{\nu+1}(\eta((1 - w)/w)^{1/\nu} - \rho), \]

where \( T_{\nu+1} \) is the cumulative distribution function of the univariate t-distribution with \( \nu - 1 \) degrees of freedom. As \( \nu \to \infty \), the copula weakly converges to the Hüsler–Reiss copula (HRcop) because the t-distribution converges to the normal (see Examples for a study of this copula).
The \( tEV(u, v; \rho, \nu) \) copula is a two-parameter option when working with extreme-value copula. There is a caveat though. Demarta and McNeil (2004) conclude that “the parameter of the Gumbel \([GHcop]\) or Galambos \([GLcop]\) A-functions [the \textit{Pickend dependence function} and B-function by association] can always be chosen so that the curve is extremely close to that of the \( tEV \) \( A \)-function for any values of \( \nu \) and \( \rho \). The implication is that in all situations where the \( tEV \) copula might be deemed an appropriate model then the practitioner can work instead with the simpler Gumbel or Galambos copulas.”

Usage

\[
tEVcop(u, v, para=NULL, ...)\]

Arguments

- \( u \): Nonexceedance probability \( u \) in the \( X \) direction;
- \( v \): Nonexceedance probability \( v \) in the \( Y \) direction;
- \( para \): A vector (two element) of parameters in \( \rho \) and \( \nu \) order; and
- ...: Additional arguments to pass.

Value

Value(s) for the copula are returned.

Note

Note that Joe (2014) shows \( x = \log(u) \) (note absence of the minus sign). This is not correct.

Author(s)

W.H. Asquith

References


See Also

\( GHcop, GLcop, HRcop \)

Examples

```r
## Not run:
tau <- 1/3 # Example from copula::evCopula.Rd
tev.cop <- copula::tevCopula(copula::iTau(copula::tevCopula(), tau))
copula::pCopula(c(0.1, 0.5), copula=tev.cop) # 0.07811367
tEVcop(0.1, 0.5, para=slot(tev.cop, "parameters")) # 0.07811367
## End(Not run)
```
## Not run:

nsim <- 2000; pargh <- c(5, 0.5, 0.5)
UV <- simCOP(nsim, cop=GHcop, para=pargh)
U <- lmomco::pp(UV[,1], sort=FALSE)
V <- lmomco::pp(UV[,2], sort=FALSE)
RT <- mleCOP(u=U, v=V, cop=tEVcop, init.para=c(0.5,log(4)),
    parafn=function(k) return(c(k[1], exp(k[2]))))
partev <- RT$para
FT <- simCOP(nsim, cop=tEVcop, para=RT$para)

tauCOP(cop=GHcop, para=pargh)
tauCOP(cop=tEVcop, para=partev)

tauCOP(cop=GHcop, para=pargh) # [1] 0.3003678
tauCOP(cop=tEVcop, para=partev) # [1] 0.3178904

densityCOPplot(cop=GHcop, para=pargh)
densityCOPplot(cop=tEVcop, para=partev, ploton=FALSE, contour.col="red") #

## End(Not run)

## Not run:

# A demonstration Joe (2014, p. 190) for which tEvocop() has
# upper tail dependence parameter as
para <- c(0.8, 10)
"tEvocop.copula" <- function(u,v, para=NULL, ...)

    if(length(u)==1) u<-rep(u,length(v)); if(length(v)==1) v<-rep(v,length(u))
    return(copula::pCopula(matrix(c(u,v), ncol=2),
        tevCopula(param=para[1], df=para[2])))

}
lamU.copBasic <- taildepCOP(cop=tEvocop, para)$lambdaU
lamU.copula <- taildepCOP(cop=tEvocop.copula, para)$lambdaU
print(c(lamU, lamU.copBasic, lamU.copula))
[[1]] 0.2925185 0.2925200 0.2925200 # So, we see that they all match.

## End(Not run)

## Not run:

# Convergence of tEvocop to HRcop as nu goes to infinity.
nu <- 10^(seq(-4, 2, by=0.1)) # nu right end point rho dependent
rho <- 0.7 # otherwise, expect to see 'zeros' errors on the plot()
# Compute Blomqvist Beta (fast computation is reason for choice)
bEv <- sapply(nu, function(n) blomCOP(tEvocop, para=c(rho, n)))
lim.thetas <- sqrt(2 / (nu*(1-rho))) # for nu --> infinity HRcop
thetas <- sapply(bEv, function(b)

    uniroot(function(l, blom=NA) { blom - blomCOP(HRcop, para=l) },
        interval=c(0,10), blom=b)$root )
plot(limit.thetas, thetas, log="xy", type="b",
    xlab="Theta of HRcop via limit nu --> infinity",
    ylab="Theta from Blomqvist Beta equivalent HRcop to tEvocop")
abline(0,1)
mtext(paste0("Notice the 'weak' convergence to lower left, and \
",
"convergence increasing with rho"))
uvlmoms

Bivariate Skewness after Joe (2014) or the Univariate L-moments of Combined U and V

Description

Joe (2014, pp. 65–66) suggests two quantile-based measures of bivariate skewness defined for uniform random variables $U$ and $V$ combined as either $\psi_{u+v-1} = u + v - 1$ or $\psi_{u-v} = u - v$ for which the $E[u] = E[v] = 0$. The bivariate skewness is the quantity $\eta$:

$$\eta(p; \psi) = \frac{x(1-p) - 2x(\frac{1}{2}) + x(p)}{x(1-p) - x(p)},$$

where $0 < p < \frac{1}{2}$, $x(F)$ is the quantile function for nonexceedance probability $F$ for either the quantities $X = \psi_{u+v-1}$ or $X = \psi_{u-v}$ using either the empirical quantile function or a fitted distribution. Joe (2014, p. 66) reports that $p = 0.05$ to “achieve some sensitivity to the tails.” How these might be related (intuitively) to L-coskew (see function lcomoms2() of the lmomco package) of the L-comoments or bivariate L-moments (bilmoms) is unknown, but see the Examples section of joeskewCOP.

Structurally the above definition for $\eta$ based on quantiles is oft shown in comparative literature concerning L-moments. But why stop there? Why not compute the L-moments themselves to arbitrary order for $\eta$ by either definition (the uvlmoms variation)? Why not fit a distribution to the computed L-moments for estimation of $x(F)$? Or simply compute “skewness” according to the definition above (the uvskew variation).

Usage

uvlmoms(u,v=NULL, umv=TRUE, p=NA, type="gno", getlmoms=TRUE, ...)

uvskew( u,v=NULL, umv=TRUE, p=0.05, type=6, getlmoms=FALSE, ...)

Arguments

u
Nonexceedance probability $u$ in the $X$ direction;

v
Nonexceedance probability $v$ in the $Y$ direction and if NULL then $u$ is treated as a two column R data.frame;

umv
A logical controlling the computation of $\psi$: $\psi = u - v$ (umv = TRUE) or $\psi = u + v - 1$ (umv = FALSE). The “m” is to read “minus”;

p
A suggested $p$ value is $p = 0.05$. If is.na(NA), then getlmoms is set to TRUE (see below);
type

The type argument is mutable, and is a syntax match to the canonical use in package `lmomco`. Variation from that package however is permitted. Either type is an integer between 1 and 9 selecting one of the nine quantile algorithms described for the quantile function in R. The default 6 uses the Weibull plotting positions and differs from the R default of 7. Otherwise type must be a valid distribution abbreviation for the `lmomco` package as in the abbreviation list `dist.list` function of that package. The gno shown as a default for the generalized normal distribution (see distribution type "gno" in package `lmomco`);

getlmoms

A logical triggering whether the L-moments of either $\psi_{u+v-1}$ or $\psi_{u-v}$ are returned instead computing the above definition of "skewness;" and

Value

An R list of the univariate L-moments of $\eta$ is returned (see documentation for `lmoms` in the `lmomco` package). Or the skewness of $\eta$ can be either (1) based on the empirical distribution based on plotting positions by the quantile function in R using the type as described, or (2) based on the fitted quantile function for the parameters of a distribution for the `lmomco` package.

Author(s)

W.H. Asquith

References


See Also

COP

Examples

```r
## Not run:
set.seed(234)
UV <- simCOP(n=100, cop=GHcop, para=1.5, graphics=FALSE)
lmr <- uvlmoms(UV); print(lmr) # L-kurtosis = 0.16568268
uvskew(UV, p=0.10) # -0.1271723
uvskew(UV, p=0.10, type="gno") # -0.1467011
## End(Not run)

## Not run:
pss <- seq(0.01,0.49, by=0.01)
ETA <- sapply(1:length(pss), function(i) uvskew(UV, p=pss[i], type=5, uvmi=FALSE) )
plot(pss, ETA, type="l", xlab="P FACTOR", ylab="BIVARIATE SKEWNESS") #
## End(Not run)
```
The Vuong Procedure for Parametric Copula Comparison

Description

Perform the Vuong Procedure following Joe (2014, pp. 257–258). Consider two copula densities $f_1 = c_1(u, v; \Theta_1)$ and $f_2 = c_2(u, v; \Theta_2)$ for two different bivariate copulas $C_1(\Theta_1)$ and $C_2(\Theta_2)$ having respective parameters $\Theta_1$ and $\Theta_2$ that provide the “closest” Kullback–Leibler Divergence from the true copula density $g(u, v)$.

The difference of the Kullback–Leibler Divergence (kullCOP) of the two copulas from the true copula density can be measured for a sample of size $n$ and bivariate sample realizations $\{u_i, v_i\}$ by

$$\hat{D}_{12} = n^{-1} \sum_{i=1}^{n} D_i,$$

where $\hat{D}_{12}$ is referred to in the copBasic package as the “Vuong $D$” and $D_i$ is defined as

$$D_i = \log \left[ \frac{f_1(u_i, v_i; \Theta_2)}{f_2(u_i, v_i; \Theta_1)} \right].$$

The variance of $\hat{D}_{12}$ can be estimated by

$$\hat{\sigma}_{12}^2 = (n - 1)^{-1} \sum_{i=1}^{n} (D_i - \hat{D}_{12})^2.$$

The sample estimate and variance are readily turned into the $100 \times (1 - \alpha)$ confidence interval by

$$\hat{D}_{12}^{(lo)} < \hat{D}_{12} < \hat{D}_{12}^{(hi)},$$

where, using the quantile (inverse) function of the t-distribution $\sim T^{-1}(F; df=(n-2))$ for nonexceedance probability $F$ and $n-2$ degrees of freedom for $n$ being the sample size, the confidence interval is

$$\hat{D}_{12} - T^{-1}(1-\alpha/2) \times \hat{\sigma}_{12}/\sqrt{n} < \hat{D}_{12} < \hat{D}_{12} + T^{-1}(1-\alpha/2) \times \hat{\sigma}_{12}/\sqrt{n}.$$

Joe (2014, p. 258) reports other interval forms based (1) on the Akaike (AIC) correction and (2) on the Schwarz (BIC) correction, which are defined for AIC as

$$AIC = \hat{D}_{12} - (2n)^{-1} \log(n) \left[ \dim(\Theta_2) - \dim(\Theta_1) \right] \pm T^{-1}(1-\alpha/2) \times \hat{\sigma}_{12}/\sqrt{n},$$

and for BIC as

$$BIC = \hat{D}_{12} - (2n)^{-1} \log(n) \left[ \dim(\Theta_2) - \dim(\Theta_1) \right] \pm T^{-1}(1-\alpha/2) \times \hat{\sigma}_{12}/\sqrt{n}.$$

The AIC and BIC corrections incorporate the number of parameters in the copula and for all else being equal the copula with the fewer parameters is preferable. If the two copulas being compared
have equal number of parameters than the AIC and BIC equate to \( \hat{D}_{12} \) and the same confidence interval because the difference \( \dim(\Theta_2) - \dim(\Theta_1) \) is zero.

Joe (2014, p. 258) reports that these three intervals can be used for diagnostic inference as follows. If an interval contains 0 (zero), then copulas \( C_1(\Theta_1) \) and \( C_2(\Theta_2) \) are not considered significantly different. If the interval does not contain 0, then copula \( C_1(\Theta_1) \) or \( C_2(\Theta_2) \) is the better fit depending on whether the interval is completely below 0 (thus \( C_1(\Theta_1) \) better fit) or above 0 (thus \( C_2(\Theta_2) \) better fit), respectively. Joe (2014) goes on to emphasize that “the procedure compares different copulas and assesses whether they provide similar fits to the data. [The procedure] does not assess whether [either copula] is a good enough fit.”

**Usage**

\[
\text{vuongCOP}(u, v=NULL, \text{cop1=NULL, cop2=NULL, para1=NULL, para2=NULL,}
\text{alpha=0.05, method=c("D12", "AIC", "BIC"),}
\text{the.zero=.Machine$double.eps^0.25, ...})
\]

**Arguments**

- **u**: Nonexceedance probability \( u \) in the \( X \) direction;
- **v**: Nonexceedance probability \( v \) in the \( Y \) direction and if NULL then \( u \) is treated as a two-column \( \mathbb{R} \) data.frame;
- **cop1**: A copula function corresponding to copula \( f_1 \) in the Vuong Procedure;
- **para1**: Vector of parameters or other data structure, if needed, to pass to the copula \( f_1 \);
- **cop2**: A copula function corresponding to copula \( f_2 \) in the the Vuong Procedure;
- **para2**: Vector of parameters or other data structure, if needed, to pass to the copula \( f_2 \);
- **alpha**: The \( \alpha \) in the Vuong Procedure, which results in the \( 100 \times (1 - \alpha) \) confidence interval (two sided);
- **method**: The interval to evaluate as to position of the respective statistic form the comparison of the two copulas;
- **the.zero**: The value for “the zero” of the copula density function. This argument is the argument of the same name for \( \text{densityCOP} \). The default here is intended to suggest that a tiny nonzero value for density will trap the numerical zero densities; and
- **...**: Additional arguments to pass to the \( \text{densityCOP} \) function.

**Value**

An \( \mathbb{R} \) list is returned having the following components:

- **title**: A descriptive title of the procedure;
- **method**: A textual description of the method setting;
- **result.text**: A textual description of the result of the Vuong Procedure;
- **result**: A value 1 if \( C_1(\Theta_1) \) is better fit, 2 if copula \( C_2(\Theta_2) \) is better fit, and \( \emptyset \) if neither is better \( (\hat{D}_{12} = 0) \), and \( \text{NA} \) including the likely(?!) erroneous situation of \( C_1(\Theta_1) \equiv C_2(\Theta_2) \);
The two-sided p-values of the Vuong Procedure inclusive of AIC and BIC;

A named vector of the lower and upper bounds of Vuong $D$ at the respective confidence interval percentage along with $D_{12}$ and $\sigma_{12}^2$;

A named vector of the lower and upper bounds of Vuong AIC at the respective confidence interval percentage;

A named vector of the lower and upper bounds of Vuong BIC at the respective confidence interval percentage; and

A named vector of the alpha, sample size, value for the t-distribution quantile $qt(1-\alpha/2, df=n)$, and $\hat{\sigma}_{12}$.

The `vuongCOP` function along with `kullCOP` and features of function `densityCOPplot` represent collective milestones towards copula inference and diagnostics post fitting of copulas to the usual measures of association such as the Kendall Tau ($\tau_K$) and Spearman Rho ($\rho_S$) and their copula counterparts $\tau_C$ (tauCOP) and $\rho_C$ (rhoCOP).

For an example application, imagine a problem of say low springflow risk at “nearby springs” that jointly should converge in the lower tail because drought usually has a strong regional impact. First, it is necessary to form a reflection of the Gumbel–Hougaard copula ($GH(u,v; \Theta_{GH})$; GHcop) but parameter estimation using $\hat{\tau}_K$ is the same because sample $\hat{\tau}_K$ is invariant to reflection.

```r
"rGHcop" <- function(u,v, ...) { u + v - 1 + GHcop(1-u, 1-v, ...) }
```

The prior code also sets a seed on the pseudo-random number generator so that reported values here are reproducible. The reflected $GH(u,v; \Theta_{GH})$ is denoted $rGH(u,v; \Theta_{rGH})$.

Second, the PSP$(u,v)$ copula (PSP) is chosen as the parent distribution, and this copula has no parameter. The PSP has lower-tail dependency, which will be important as discussion unfolds. The following two lines of code establish a sample size to be drawn from the PSP and then simulates a sample from that copula. The color grey is used for the simulated values on the figure produced by simCOP, which forms a background example of the joint structure of the PSP copula.

```r
n <- 390
UV <- simCOP(cop=PSP, n=n, col=8, pch=16) # simulate and form the base image
```

By inspection of the so-produced graphic, it is obvious that there is contraction in the lower-left corner of the plot, which is a geometric representation of tail dependency. The lower-tail dependency thus phenomenologically says that there is joint interconnect during low springflow conditions—both springs are likely to be at low flow simultaneously. The variable UV contains the bivariate data as uniform variables (nonexceedance probabilities $u$ and $v$).

The Plackett copula ($PL(u,v; \Theta_{PL})$; PLACKETTcop) and the $rGH(u,v; \Theta_{rGH})$ copula are chosen as candidate models of the “unknown” parent. Both PL and rGH copulas use different “measures of association” for their parameter estimation. Next, sample estimates of the copula parameters using Schweizer and Wolff Sigma $\hat{\sigma}_C$. The sample value computations and parameter estimates also are set as shown in the following code:
STEP 1—Compute Kullback–Leibler sample size: The Kullback–Leibler Divergences (KL(f|g) and KL(g|f)) are computed (kullCOP) for the evaluation of the sample size as appropriate for distinguishing between the two candidate copulas 95 percent of the time. The Kullback–Leibler sample size \( n_{fg} \) also is computed as the following code illustrates and provides additional commentary.

\[
KL <- replicate(20, kullCOP(cop1=PLcop, para1=paraPL, # CPU intensive
                   cop2=rGHCop, para2=paraGH, n=1E5)$KL.sample.size)
\]

\[
print(round(mean(KL))) # n_{fg} = 221 sample size
\]

\[
print( range(KL)) # 204 <-- n_{fg} --> 252 sample size range
\]

Depending on the sample \( \hat{\sigma}_C \) coming from the simulation of the parent PSP copula, the call to kullCOP will likely report different \( n_{fg} \) values because \( n_{fg}(C_1(\Theta_1), C_1(\Theta_1)) \). These sample sizes have a range for 20 replications of about \( n_{fg} = 204-252 \). The result here is \( n_{fg} = 221 \) and thus the sample size \( n = 390 \) should be more than large enough to generally distinguish between the PL and rGH copulas at the respective sample measure of association.

STEP 2—Perform the Vuong Procedure: The Vuong Procedure can now be completed. Now watch the copula and parameter order in the next code for mistakes, the author has purposefully switched order here to draw attention to the need to make sure argument cop1 has the correct parameter(s) for copula 1 (the PL). The two calls to simCOP are made to graphically superimpose these simulations on top of the parent PSP.

\[
VD <- vuongCOP(UV, cop2=rGHCop, para2=paraGH, cop1=PLcop, para1=paraPL)
\]

\[
print(VD) # "Copula 2 better" or rGHcop (Gumbel-Hougaard is better)
\]

\[
set.seed(385) # seems harmless enough to reuse the seed to emphasize "fit"
\]

\[
TMP <- simCOP(cop=PLcop, para=paraPL, n=n, plot=FALSE, col="red", pch=16,cex=0.5)
\]

\[
set.seed(385) # seems harmless enough to reuse the seed to emphasize "fit"
\]

\[
TMP <- simCOP(cop=rGHCop, para=paraGH, n=n, plot=FALSE, col="green",pch=16,cex=0.5)
\]

\[
rm(TMP) # just cleaning up the workspace.
\]

Further discussion of the Vuong Procedure is informative. Simply speaking, the result is that the rGH (copula 2) has better fit than PL (copula 1). The 95-percent confident limits from the procedure for \( D_{12} = 0.049 \) with p-value 0.0012, \( \hat{\sigma}_{12} = 0.297 \), and \( n = 390 \) are 0.0194 < \( \hat{D}_{12} < 0.0786 \). This interval does not contain zero and is greater than zero and therefore a conclusion may be drawn that copula 2 has the better fit.

STEP 3—Comparison of lower-tail dependency parameters: What does the tail dependency do for inference? This can be checked by computing the lower-tail dependency parameters (\( \lambda_L \); taildepCOP) in the code that follows for each of the three copulas and the empirical copula, but note that true sample estimators do not quite exist. Numeric focus need only be on the lower tail, but the four graphics are informative.

\[
taildepCOP(cop=PSP, plot=TRUE)$lambdaL # = 1/2
\]

\[
taildepCOP(cop=PLcop, para=paraPL, plot=TRUE)$lambdaL # = ZERO
\]

\[
taildepCOP(cop=rGHCop, para=paraGH, plot=TRUE)$lambdaL # = 0.429
\]

\[
taildepCOP(cop=EMPIRcop, para=UV, plot=TRUE)$lambdaL # = 0.328
\]
The important aspect of the graphics by taildepCOP is that the rGH has lower-tail dependency whereas the PL does not. So, based on inspection rGH is superior given that we known PSP was the true parent. The empirical estimate of the \( \lambda_C^L = 0.328 \) through the EMP1Rcop copula indicates that its lower-tail dependency is closer to that of the rGH relative to PL and thus quantitatively by lower-tail dependency the rGH has a superior fit.

Therefore the rGH has a tail dependency more similar to the true model compared to the PL. Hence for this example, the rGH is clearly a superior fitting model in terms of the Vuong Procedure (fit alone) and the \( \lambda_C^L \) then is used as a follow up to shown that the rGH might be “good enough” an approximation to the PSP. The efficacy of reflecting the GH copula into a “new” form as rGH is demonstrated. Users are strongly encouraged to review the so-produced graphic from the simCOP call several listings back for \( n = 390 \), and lastly, this example is one for which absence of the argument snv (standard normal variate [scores]) by simCOP makes the tail dependency issue for the sample size more prominent in the graphic.

**STEP 4**—Qualitatively compare using copula density plots: Graphical depiction of copula density contours by the densityCOPplot function supports the conclusion that the rGH is the superior model relative to the PL. The so-produced graphic obviously shows that the rGH strongly mimics the shape of the parent PSP.

```r
densityCOPplot(cop=PSP, contour.col=8) # grey is the parent bivariate density
densityCOPplot(cop=PLcop, para=paraPL, contour.col="green", ploton=FALSE)
densityCOPplot(cop=rGHcop, para=paraGH, contour.col="red", ploton=FALSE)
```

**STEP 5**—Compute L-comoments of the data via simulation and estimate the sampling distributions: An open research problem is the what if any role that L-comoments might play in either copula estimation or inference. (There being very little literature on the topic?) Because a measure of association was used for parameter estimation, the L-correlation is uninformative, but a comparison is conceptually useful. The \( \hat{\sigma}_C = 0.4969 \) and Spearman Rho of the data \( \hat{\rho}_S \) and the L-correlations \( \hat{\rho}_S \approx r_2^{[12]} \approx r_2^{[21]} \approx 0.497 \) are all similar as mandated by the mathematics.

Inference using L-coskew and L-cokurtosis seems possible. The following code listing is CPU intensive. First, the L-correlation, L-coskew, and L-cokurtosis values are computed from the simulated sample by the lcomoms2() function of the lmomco package. Second and third, the respective sampling distributions of these L-comoments (lcomCOPpv) for the two copulas are estimated.

```r
UVlmr <- lmomco::lcomoms2(UV, nmom=4) # The sample L-comoments
# This execution will result in nonrejection of rGH copula.
GHlmr <- lcomCOPpv(n, UVlmr, cop=rGHcop, para=paraGH) # NONREJECTION
# LcomType n Mean Lscale Lskew Lkurt sample.est p.value signif
# Tau3[12] 390 -0.06952 0.01819 0.04505 0.12024 -0.11188 0.08795 .
# Tau3[21] 390 -0.06739 0.02084 0.04104 0.12917 -0.10673 0.14162 -
# Tau3[12:21] 390 -0.06845 0.01713 0.04930 0.11696 -0.10931 0.08161 .
# Tau4[12] 390 0.04970 0.01682 -0.01635 0.10150 0.04183 0.38996 -
# Tau4[21] 390 0.05129 0.01606 -0.06833 0.13798 0.07804 0.17470 -
# Tau4[12:21] 390 0.05049 0.01329 -0.02045 0.12001 0.05994 0.35069 -

# This execution will result in rejection of Plackett copula.
PLlmr <- lcomCOPpv(n, UVlmr, cop=PLACKETTcop, para=paraPL) # REJECT PLACKETT
# LcomType n Mean Lscale Lskew Lkurt sample.est p.value signif
# Tau3[12] 390 -0.00267 0.02133 0.01556 0.09581 -0.11188 0.00129 **
```

The important aspect of the graphics by taildepCOP is that the rGH has lower-tail dependency whereas the PL does not. So, based on inspection rGH is superior given that we known PSP was the true parent. The empirical estimate of the \( \lambda_C^L = 0.328 \) through the EMP1Rcop copula indicates that its lower-tail dependency is closer to that of the rGH relative to PL and thus quantitatively by lower-tail dependency the rGH has a superior fit.
Because each copula was fit to a measure of association, the p-values for the L-correlations are all nonsignificant (noninformative because of how the copulas were fit), and therefore p-values quite near to the 50th percentile should be produced. So here, the L-correlation is noninformative on fit even though it might have some role because it is asymmetrical unlike the statistics $\tau_K$ and $\rho_S$.

The results in variable $\text{GhLmr}$ show no statistically significant entries (all p-values $>0.05 = (\alpha = 0.1)/2$) for L-coskew and L-cokurtosis—the \textit{rGH copula is not rejected}. The results in $\text{PLlmr}$ show many p-values $<0.05 = (\alpha = 0.1)/2$ for both L-coskew and L-cokurtosis—the \textit{PL copula is rejected}. The experimental L-comoment inference shown is consistent with results with the Vuong Procedure.

The Vuong Procedure, however, does not address adequacy of fit—it just evaluates which copula fits better. The inspection of the lower tail dependency results previously shown ($\lambda_{\text{PSP}}^L = 1/2 \approx \lambda_{\text{rGH}}^U = 0.429$) along with the L-coskew and L-cokurtosis of the sample being well within the sample distribution suggests that the \textit{rGH} is a adequate mimic of the \textit{PSP} copula.

Some open research questions concern the numerical performance of the L-comoments as simulation sample size becomes large and whether or not the L-comoments should be computed on the probabilities \{u, v\}. Also should conversion to normal scores be made and if so, should adjustment by the \textit{Hazen plotting positions} ($u_i = (r_i - 0.5)/n$ for rank $r_i$) be made that Joe (2014) repeatedly advocates when standard normal variates (scores) $z_i = \Phi^{-1}(u_i)$ for quantile function of standard normal distribution $\Phi(0, 1)$? Collectively, Nelsen (2006) and Salvadori \textit{et al.} (2007) are both silent on the matter of normal score conversion, and in conclusion Nelsen (2006), Salvadori \textit{et al.} (2007), and Joe (2014) also are all silent on L-comoment applications with copulas.

Author(s)

W.H. Asquith

References


See Also

densityCOP, kullCOP, simCOP, statTn, mleCOP

Examples

# See extended code listings and discussion in the Note section
# See Examples in mleCOP() (Last example therein might suggest a problem in the
# implied 95th percentile associated with n_fg above.
The Fréchet–Hoeffding Lower-Bound Copula

**Description**

Compute the Fréchet–Hoeffding lower-bound copula (Nelsen, 2006, p. 11), which is defined as

\[ W(u, v) = \max(u + v - 1, 0). \]

This is the copula of perfect anti-association (countermonotonicity, perfectly negative dependence) between \( U \) and \( V \) and is sometimes referred to as the countermonotonicity copula. Its opposite is the \( M(u, v) \) copula (comonotonicity copula; \( M \)), and statistical independence is the \( \Pi(u, v) \) copula (\( P \)).

**Usage**

\[ W(u, v, \ldots) \]

**Arguments**

- \( u \) Nonexceedance probability \( u \) in the \( X \) direction;
- \( v \) Nonexceedance probability \( v \) in the \( Y \) direction; and
- \( \ldots \) Additional arguments to pass.

**Value**

Value(s) for the copula are returned.

**Author(s)**

W.H. Asquith

**References**


**See Also**

\( M, P \)

**Examples**

- \( W(0.4, 0.6) \)
- \( W(0, 0) \)
- \( W(1, 1) \)
Description

Compute the measure of association known as Schweizer–Wolff Sigma $\sigma_C$ of a copula according to Nelsen (2006, p. 209) by

$$\sigma_C = 12 \int \int_{x^2} |C(u, v) - uv| \, du \, dv,$$

which is $0 \leq \sigma_C \leq 1$. It is obvious that this measure of association, without the positive sign restriction, is similar to the following form of Spearman Rho ($\rho_{COP}$) of a copula:

$$\rho_C = 12 \int \int_{x^2} [C(u, v) - uv] \, du \, dv.$$

If a copula is positively quadrant dependent (PQD, see isCOP.PQD) then $\sigma_C = \rho_C$ and conversely if a copula is negatively quadrant dependent (NQD) then $\sigma_C = -\rho_C$. However, a feature making $\sigma_C$ especially attractive is that for random variables $X$ and $Y$, which are not PQD or NQD—copulas that are neither larger nor smaller than $\Pi$—is that “$\sigma_C$ is often a better measure of [dependency] than $\rho_C$” (Nelsen, 2006, p. 209).

Usage

`wolfCOP(cop=NULL, para=NULL, as.sample=FALSE, brute=FALSE, delta=0.002, ...)`

Arguments

cop A copula function;
para Vector of parameters or other data structure, if needed, to pass to the copula;
as.sample A logical controlling whether an optional R data.frame in para is used to compute the $\hat{\sigma}_C$ (see Note). If set to -1, then the message concerning CPU effort will be suppressed;
brute Should brute force be used instead of two nested integrate() functions in R to perform the double integration;
delta The $du$ and $dv$ for the brute force (brute=TRUE) integration; and
...

Value

The value for $\sigma_C$ is returned.
**Note**

A natural estimator for $\sigma_C$ is through the empirical copula (Póczos et al., 2015) and can be computed as

$$\hat{\sigma}_C = \frac{12}{n^2 - 1} \sum_{i=1}^{n} \sum_{j=1}^{n} \left| \hat{C}_n\left( \frac{i}{n}, \frac{j}{n} \right) - \frac{i}{n} \times \frac{j}{n} \right|,$$

where $\hat{C}_n$ is the simplest empirical copula of

$$\hat{C}_n\left( \frac{i}{n}, \frac{j}{n} \right) = \frac{1}{n} \{ \# \text{ of } (U_k \leq U_i, V_k \leq V_j) \}.$$

An extended example is informative. First declare a composite of two different Plackett copulas (PLcop) and simulate a few hundred values:

```r
para <- list(cop1 = PLcop, cop2 = PLcop, para1 = 0.145, para2 = 21.9, alpha = 0.81, beta = 0.22)
D <- simCOP(n = 300, cop = composite2COP, para = para, cex = 0.5, col = rgb(0, 0, 0, 0.2), pch = 16)
```

Second, show that this copula is globally PQD (isCOP.PQD), but there is a significant local NQD part of $\mathbb{I}^2$ space that clearly is NQD.

```r
PQD <- isCOP.PQD(cop = composite2COP, para = para, uv = D)
message(PQD$global.PQD) # TRUE
points(D, col = PQD$local.PQD + 2, lwd = 2)
```

This composited copula intersects, that is, passes through, the $P$ copula. Hence by the logic of Nelsen (2006), then the $\sigma_C$ should be larger than $\rho_C$ as shown below

```r
wolfCOP(cop = composite2COP, para = para) # 0.08373378 (theoretical)
rhoCOP(cop = composite2COP, para = para) # 0.02845131 (theoretical)
hoefCOP(cop = composite2COP, para = para) # 0.08563823 (theoretical)
```

In fact, the output above also shows Schweizer–Wolff Sigma to be larger than Blomqvist Beta (blomCOP), Gini Gamma (giniCOP), and Kendall Tau (tauCOP). The Schweizer–Wolff Sigma has captured the fact that although the symbols plot near randomly on the figure, the symbol coloring for PQD and NQD clearly shows local dependency differences. The sample version is triggered by

```r
wolfCOP(para = D, as.sample = TRUE) # 0.09278467 (an example sample)
```

**Author(s)**

W.H. Asquith

**References**


See Also

blomCOP, footCOP, giniCOP, hoefCOP, rhoCOP, tauCOP, joeskewCOP, uvlmoms

Examples

```r
## Not run:
wolfCOP(cop=PSP) # 0.4784176
## End(Not run)

## Not run:
n <- 1000; UV <- simCOP(n=n, cop=NA212cop, para=7.53, graphics=FALSE)
wolfCOP(cop=NA212cop, para=7.53) # 0.9884666 (theoretical)
wolfCOP(para=UV, as.sample=TRUE) # 0.9884174 (sample)
## End(Not run)

## Not run:
# Redo D from Note section above
para <- list(cop1=PLcop, cop2=PLcop,
             para1=0.145, para2=21.9, alpha=0.81, beta=0.22)
D <- simCOP(n=300, cop=composite2COP, para=para,
cex=0.5, col=rgb(0, 0, 0, 0.2), pch=16)

the.grid <- EMPIRgrid(para=D)
the.persp <- persp(the.grid$empcop, theta=-25, phi=20, shade=TRUE,
xlab="U VARIABLE", ylab="V VARIABLE", zlab="COPULA C(u,v)")
empcop <- EMPIRcopdf(para=D) # data.frame of all points
points(trans3d(empcop$u, empcop$v, empcop$empcop, the.persp), cex=0.7,
       col=rgb(0, 1-sqrt(empcop$empcop), 1, sqrt(empcop$empcop)), pch=16)
points(trans3d(empcop$u, empcop$v, empcop$empcop, the.persp),
col=PQD$local.PQD+1, pch=1)

layout(matrix(c(1,2,3,4), 2, 2, byrow = TRUE), respect = TRUE)
PQD.NQD.cop <- gridCOP(cop=composite2COP, para=para)
Pi <- gridCOP(cop=P)
RHO <- PQD.NQD.cop - Pi; SIG <- abs(RHO)

the.persp <- persp(PQD.NQD.cop, theta=-25, phi=20, shade=TRUE, cex=0.5,
xlab="U VARIABLE", ylab="V VARIABLE", zlab="COPULA C(u,v)")

mtext("The Copula that has local PQD and NQD", cex=0.5)

the.persp <- persp(Pi, theta=-25, phi=20, shade=TRUE, cex=0.5,
xlab="U VARIABLE", ylab="V VARIABLE", zlab="COPULA C(u,v)")
mtext("Independence (Pi)", cex=0.5)

the.persp <- persp(RHO, theta=-25, phi=20, shade=TRUE, cex=0.5,
xlab="U VARIABLE", ylab="V VARIABLE", zlab="COPULA C(u,v)")
mtext("Copula delta: Integrant of Spearman Rho", cex=0.5)

the.persp <- persp(SIG, theta=-25, phi=20, shade=TRUE, cex=0.5,
xlab="U VARIABLE", ylab="V VARIABLE", zlab="COPULA C(u,v)")
mtext("abs(Copula delta): Integrant of Schweizer-Wolff Sigma", cex=0.5)
## End(Not run)
```
Index

* AIC
  mleCOP, 206
  vuongCOP, 285
* API to the copula package
  GHcop, 119
  med.regressCOP, 202
  mleCOP, 206
  spectralmeas, 255
* Akaike Information Criterion
  aicCOP, 13
* Akaike
  mleCOP, 206
  vuongCOP, 285
* Ali–Mikhail–Haq copula
  AMHcop, 15
* Ali-Mikhail-Haq copula
  AMHcop, 15
* Asymmetric Gumbel–Hougaard copula
  GHcop, 119
* BIC
  mleCOP, 206
  vuongCOP, 285
* Bekrizade et al. (2012) Examples and Exercises
  FGMcop, 111
* Bernstein copula
  EMPIRcop, 83
* Bernstein empirical copula
  EMPIRcop, 83
* Blomqvist (Schmid-Schmidt) Beta
  blomCOPs, 34
  taildepCOP, 270
* Blomqvist Beta
  blommatrixCOP, 27
  blomCOP, 31
  blomCOPs, 34
* Clayton copula
  CLcop, 39
* Durante and Sempi (2015) Examples and Exercises
  FGMcop, 111
  Erdely (2017) Examples and Exercises
  glueCOP, 131
  Farlie–Gumbel–Morgenstern copula
  FGMcop, 111
  Farlie-Gumbel-Morgenstern copula
  FGMcop, 111
  Frechet Family copula
  FRECHETcop, 115
  Frechet lower bound
  W, 291
  Frechet lower-bound copula
  W, 291
  Frechet upper bounds
  M, 201
  Frechet upper-bound copula
  M, 201
  Frechet–Hoeffding lower-bound copula
  W, 291
  Frechet–Hoeffding lower-bound
  W, 291
  Frechet–Hoeffding upper bounds
  M, 201
  Frechet–Hoeffding upper-bound copula
  M, 201
  Frechet–Hoeffding lower-bound copula
  W, 291
  Frechet–Hoeffding lower-bound
  W, 291
  Frechet–Hoeffding upper bounds
  M, 201
  Frechet–Hoeffding upper-bound copula
  M, 201
  Galambos copula (lower extreme value form)
  GLcop, 129
* Galambos copula
  GLcop, 129
* Galambos extreme value copula
  GLcop, 129
* Gaussian copula
  blomCOP, 31
  med.regressCOP, 202
* Gumbel–Hougaard copula
  GHcop, 119
* Gumbel–Hougaard extreme value copula
  GHcop, 119
* Gumbel–Hougaard extreme value copula
  GHcop, 119
* Hazen copula
  EMPIRcop, 83
* Hazen empirical copula
  EMPIRcop, 83
* Hoeffding Phi-Square
  hoefCOP, 135
* Hoeffding Phi
  hoefCOP, 135
* Husler–Reiss copula
  HRcop, 139
* Husler–Reiss extreme value copula
  HRcop, 139
* Husler-Reiss extreme value copula
  HRcop, 139
* I-divergence
  kullCOP, 179
* Jeffery Divergence
  kullCOP, 179
* Jeffery’s Divergence
  kullCOP, 179
* Joe (2014) Examples and Exercises
  blomCOP, 31
  CLcop, 39
  densityCOP, 62
  densityCOPplot, 64
  derCOPinv, 71
  GHcop, 119
  GLcop, 129
  HRcop, 139
  JOcopB5, 150
  joeskewCOP, 152
  kfuncCOP, 167
  kullCOP, 179
  PLACKETTcop, 215
  rhoCOP, 236
  semicorCOP, 243
  tailordCOP, 274
  tEVcop, 280
* Joe/B5 copula
  JOcopB5, 150
* Joe/BB4 copula
  GLcop, 129
* Kendall Function (L-moments)
  kfuncCOPlmoms, 176
* Kendall Function L-moment diagram
  kfuncCOPlmoms, 176
* Kendall Function L-moment ratio diagram
  kfuncCOPlmoms, 176
* Kendall Function L-moments
  kfuncCOPlmoms, 176
* Kendall Function Lmoment diagram
  kfuncCOPlmoms, 176
* Kendall Function Lmoment ratio diagram
  kfuncCOPlmoms, 176
* Kendall Function Lmoments
  kfuncCOPlmoms, 176
* Kendall Function
  kfuncCOP, 167
* Kendall Measure
  kfuncCOP, 167
* Kendall Tau
  tauCOP, 277
* Kendall distribution function
  kfuncCOP, 167
* Kendall function
  kfuncCOP, 167
* Kendall inverse distribution function
  kfuncCOPinv, 174
* Kendall measure
  kfuncCOP, 167
* Kendall quantile function
  kfuncCOPinv, 174
* Kullback–Leibler Divergence
  kullCOP, 179
* Kullback–Leibler sample size
  kullCOP, 179
* Kullback–Leibler
  mleCOP, 206
  vuongCOP, 285
* Kullback–Leibler Divergence
  kullCOP, 179
* Kullback–Leibler sample size
  kullCOP, 179
* Kullback–Leibler
INDEX

* L-comoment copula inference
  vuongCOP, 285
* L-moments
  bilmoms, 22
  copBasic-package, 4
  lcomCOP, 184
  lcomCOPpv, 187
  lcomoms2.ABcop2parameter, 190
  lcomoms2.ABKGCop2parameter, 193
  simcomposite3COP, 245
  simcompositeCOP, 247
* L-moments of combined U and V
  uvlmoms, 283
* L-moments
  kfuncCOPlmoms, 176
  uvlmoms, 283
* L-comoment copula inference
  vuongCOP, 285
* Lcomoments
  bilmoms, 22
  lcomCOP, 184
* Linear Spearman copula
  FRECHETcop, 115
* Linear Spearman
  FRECHETcop, 115
* Lmoments
  kfuncCOPlmoms, 176
  uvlmoms, 283
* Lp distance
  hoefCOP, 135
* Markov Product
  prod2COP, 221
* Moments of combined U and V
  joeskewCOP, 152
* Nelsen (2006) Examples and Exercises
  asCOP, 18
  bilmoms, 22
  blomCOP, 31
  copBasic-package, 4
  derCOPinv, 71
  hoefCOP, 135
  isfuncCOP, 149
  prod2COP, 221
  RFcop, 233
  taildepCOP, 270
  tauCOP, 277
* Normal copula
  blomCOP, 31
  med.regressCOP, 202
* Nu-Skew and Nu-Star (skewnesses)
  joeskewCOP, 152
* Package randtoolbox
  bilmoms, 22
  kullCOP, 179
* Pareto copula
  PARETOcop, 247
* Pearson correlation coefficient
  gEVcop, 118
  P, 212
  rhoCOP, 236
* Plackett copula
  PLACKETTcop, 215
  PLACKETTpar, 217
  PLACKETTsim, 219
* Raftery copula
  RFcop, 233
* Rosenblatt transform
  simCOP, 249
* Salvadori et al. (2007) Examples and Exercises
  derCOPinv, 71
  kfuncCOP, 167
  surfCOP, 267
* Sobol sequence (Monte Carlo integration)
  bilmoms, 22
  kullCOP, 179
* Spearman Rho
  bilmoms, 22
  gEVcop, 118
  lcomCOP, 184
  rhoCOP, 236
* Table of Copulas
  copBasic-package, 4
* Table of Probabilities
  copBasic-package, 4
* Vuong Procedure
  mleCOP, 206
  vuongCOP, 285
* Vuong
  mleCOP, 206
  vuongCOP, 285
* Weibull copula
  EMPIRcop, 83
* Weibull empirical copula
INDEX

EMPIRcop, 83

* Zhang and Singh (2019) Examples and Exercises
derCOPinv, 71

* asymmetric Gumbel copula
  GHcop, 119

* asymmetric Gumbel–Hougaard copula
  GHcop, 119

* asymmetric Gumbel-Hougaard copula
  GHcop, 119

* asymmetric logistic copula
  GHcop, 119

* bivariate L-correlation
  blilmoms, 22
  lcomCOP, 184

* bivariate L-kurtosis
  blilmoms, 22
  lcomCOP, 184

* bivariate L-moments
  blilmoms, 22
  lcomCOP, 184

* bivariate L-skew
  blilmoms, 22
  lcomCOP, 184

* bivariate asymmetry (measures)
  blilmoms, 22
  lcomCOP, 184
  uvlmoms, 283

* bivariate skewness
  blilmoms, 22
  joeskewCOP, 152
  lcomCOP, 184
  uvlmoms, 283

* comprehensive copula
  FRELCHETcop, 115
  PLACKETTcop, 215

* concordance function
  tauCOP, 277

* conditional approach
  simCOP, 249

* conditional cumulative distribution function
  derCOP, 67
  derCOP2, 69

* conditional quantile function
  derCOPinv, 71
  derCOPinv2, 75

* conditional return period
  surfuncCOP, 267

* copula (characteristics)
  blomatrixCOP, 27
  blomCOP, 31
  blomCOPss, 34
  diagCOP, 76
  diagCOPatf, 78
  footCOP, 113
  giniCOP, 126
  hoefCOP, 135
  isCOP.LTD, 140
  isCOP.permSYM, 141
  isCOP.PQD, 142
  isCOP.radsym, 145
  isCOP.RTI, 147
  joeskewCOP, 152
  kfuncCOP, 167
  kfuncCOPinv, 174
  kfuncCOPloms, 176
  lcomCOPpv, 187
  rhobeCOP, 234
  rhoCOP, 236
  semicorCOP, 243
  tailconCOP, 268
  taildepCOP, 270
  tailordCOP, 274
  tauCOP, 277
  wolfCOP, 292

* copula (comprehensive)
  FRELCHETcop, 115
  PLACKETTcop, 215

* copula (conditional distribution)
  COP, 52
  derCOP, 67
  derCOP2, 69
  simCOPmicro, 251

* copula (conditional quantile function)
  COP, 52
  derCOPinv, 71
  derCOPinv2, 75
  simCOPmicro, 251

* copula (density)
  densityCOP, 62
  densityCOPplot, 64
  kullCOP, 179
  PLACKETTcop, 215
  vuongCOP, 285

* copula (derivative inverse)
derCOPinv, 71
derCOPinv2, 75
* copula (derivative)
derCOP, 67
derCOP2, 69
* copula (empirical)
EMPIRcop, 83
* copula (estimation)
PLACKETTpar, 217
* copula (extreme value)
gEVcop, 118
GHcop, 119
GLcop, 129
HRcop, 139
tEvCop, 280
* copula (fitting)
copBasic.fitpara, 56
mleCOP, 206
* copula (formulas)
AMHcop, 15
asCOP, 18
CLCOP, 39
COP, 52
FGMcop, 111
FRECHETcop, 115
gEVcop, 118
GHcop, 119
GLcop, 129
glueCOP, 131
HRcop, 139
JOCopB5, 150
M, 201
N4212cop, 211
P, 212
PARETOcop, 214
PLACKETTcop, 215
PSP, 225
RFcop, 233
tEvCop, 280
W, 291
* copula (goodness-of-fit)
aicCOP, 13
bicCOP, 20
copBasic.fitpara, 56
psepolar, 223
rmseCOP, 239
spectralmeas, 255
statTn, 262
vuongCOP, 285
* copula (inference)
aicCOP, 13
bicCOP, 20
copBasic.fitpara, 56
dullCOP, 179
lcomCOPp, 187
psepolar, 223
rmseCOP, 239
spectralmeas, 255
statTn, 262
vuongCOP, 285
* copula (inverse)
COPinv, 59
COPinv2, 61
* copula (limits/bounds)
M, 201
W, 291
* copula (properties for extreme value)
rhobevCOP, 234
* copula (properties)
blomatrixCOP, 27
blomCOP, 31
blomCOPSS, 34
footCOP, 113
giniCOP, 126
hoefCOP, 135
isCOP.LTD, 140
isCOP.permsym, 141
isCOP.pqD, 142
isCOP.ordsym, 145
isCOP.RTI, 147
isfuncCOP, 149
joeskewCOP, 152
kfuncCOP, 167
kfuncCOPinv, 174
rhobevCOP, 234
rhopCOP, 236
semicorCOP, 243
tailconCOP, 268
taildepCOP, 270
tailordCOP, 274
tauCOP, 277
wolfCOP, 292
* copula (reflection)
COP, 52
simCOPmicro, 251
surCOP, 265
* copula (simulation)
  PLACKETTsim, 219
  simcomposite3COP, 245
  simcompositeCOP, 247
  simCOP, 249
  simCOPmicro, 251

* copula (symmetry)
  isCOP.perm_sym, 141
  isCOP.rad_sym, 145

* copula (tail characteristics)
  blomCOPss, 34
  semicorCOP, 243
  tailconCOP, 268
  taildepCOP, 270
  tailordCOP, 274

* copula (tail properties)
  blomCOPss, 34
  semicorCOP, 243
  tailconCOP, 268
  taildepCOP, 270
  tailordCOP, 274

* copula (utilities)
  asCOP, 18

* copula (utility)
  COPinv, 59
  COPinv2, 61
  derCOP, 67
  derCOP2, 69
  derCOPinv, 71
  derCOPinv2, 75

* copula L-cokurtosis
  bilmom, 22
  lcomCOP, 184

* copula L-comoments
  bilmom, 22
  lcomCOP, 184

* copula L-correlation
  bilmom, 22
  lcomCOP, 184

* copula L-coskew
  bilmom, 22
  lcomCOP, 184

* copula composition (convex combination)
  convex2COP, 48
  convexCOP, 50

* copula composition (four compositing parameters)
  composite3COP, 46

* copula composition (permutation asymmetric)
  breveCOP, 37

* copula composition (two compositing parameters)
  composite1COP, 42
  composite2COP, 44

* copula composition
  breveCOP, 37
  composite1COP, 42
  composite2COP, 44
  composite3COP, 46
  convex2COP, 48
  convexCOP, 50
  glueCOP, 131
  lcomom2.ABcop2parameter, 190
  lcomom2.ABKGcop2parameter, 193
  simcomposite3COP, 245
  simcompositeCOP, 247

* copula divergence
  kullCOP, 179

* copula gluing
  glueCOP, 131

* copula multiplication
  prod2COP, 221

* copula operator
  coCOP, 40
  COP, 52
  COPinv, 59
  COPinv2, 61
  derCOP, 67
  derCOP2, 69
  derCOPinv, 71
  derCOPinv2, 75
  duCOP, 80
  prod2COP, 221
  surCOP, 265
  surfuncCOP, 267

* copula product
  prod2COP, 221

* copula section
  sectionCOP, 240

* copula skew
  joeskewCOP, 152

* copula theory
  copBasic-package, 4

* copula utility
  gridCOP, 133
* copulatic surface
  EMPIRgrid, 89
  gridCOP, 133

* copula
  AMHcop, 15
  CLcop, 39
  EMPIRcop, 83
  FGMcop, 111
  FRECHETcop, 115
  gEVcop, 118
  GHcop, 119
  GLcop, 129
  HRcop, 139
  JOcopB5, 150
  M, 201
  N4212cop, 211
  P, 212
  PARETOcop, 214
  PLACKETTcop, 215
  PSP, 225
  RFCop, 233
  tEVcop, 280
  W, 291

* datasets
  ReineckeWell266, 231
  ReineckeWells, 232

* decomposition of Kendall Tau
  kfuncCOP, 167

* dependence index
  hoefCOP, 135

* derivative
  derCOP, 67
  derCOP2, 69
  derCOPinv, 71
  derCOPinv2, 75
  EMPIRgridderr, 93
  EMPIRgridr2, 94
  EMPIRgridrinv, 95
  EMPIRgridrinv2, 98

* diagnostics
  diagCOP, 76
  diagCOPatf, 78
  level.curvesCOP, 196
  level.curvesCOP2, 198
  level.setCOP, 199
  level.setCOP2, 200
  sectionCOP, 240

* diagonal inversion
  diagCOPatf, 78

* empirical copula (derivative inverses)
  EMPIRgridrinv, 95
  EMPIRgridrinv2, 98

* empirical copula (derivatives)
  EMPIRgridr, 93
  EMPIRgridr2, 94

* empirical copula (median regression)
  EMPIRmed.regress, 99
  EMPIRmed.regress2, 100

* empirical copula (quantile regression)
  EMPIRmed.regress, 99
  EMPIRmed.regress2, 100
  EMPIRqua.regress, 100
  EMPIRqua.regress2, 106

* empirical copula (simulation)
  EMPIRsim, 107
  EMPIRsimv, 109

* empirical copula (utility)
  EMPIRcopdf, 88
  EMPIRgrid, 89

* empirical copula
  EMPIRcop, 83
  EMPIRcopdf, 88
  EMPIRgrid, 89

* diagnostics
  diagCOPatf, 78

* extreme value copula
  gEVcop, 118
  GHcop, 119
  GLcop, 129
  HRcop, 139
  tEVcop, 280

* g-EV copula
  gEVcop, 118

* generalized Farlie-Gumbel-Morgenstern copula
  FGMcop, 111
* **goodness-of-fit**
  - aicCOP, 13
  - bicCOP, 20
  - copBasic.fitpara, 56
  - psepolar, 223
  - rmseCOP, 239
  - spectralmeas, 255
  - statTn, 262
  - vuongCOP, 285

* **hypothesis testing**
  - lcomCOPpv, 187

* **independence copula**
  - P, 212

* **inference**
  - aicCOP, 13
  - bicCOP, 20
  - copBasic.fitpara, 56
  - kullCOP, 179
  - lcomCOPpv, 187
  - psepolar, 223
  - rmseCOP, 239
  - spectralmeas, 255
  - statTn, 262
  - vuongCOP, 285

* **iterated Farlie-Gumbel-Morgenstern copula**
  - FGMcop, 111

* **joint probability (exclusive or)**
  - duCOP, 80

* **joint probability**
  - coCOP, 40
  - COP, 52
  - duCOP, 80
  - joint.curvesCOP, 157
  - joint.curvesCOP2, 160
  - jointCOP, 162
  - surCOP, 265
  - surfuncCOP, 267

* **level contours**
  - level.curvesCOP, 196
  - level.curvesCOP2, 198
  - level.setCOP, 199
  - level.setCOP2, 200

* **level curve**
  - level.curvesCOP, 196
  - level.curvesCOP2, 198
  - level.setCOP, 199
  - level.setCOP2, 200

* **level sets**
  - level.curvesCOP, 196
  - level.curvesCOP2, 198
  - level.setCOP, 199
  - level.setCOP2, 200

* **literature errors and inconsistencies**
  - GHcop, 119
  - kfuncCOP, 167
  - stabtaildepf, 259
  - tauCOP, 277

* **maximum likelihood**
  - mleCOP, 206

* **medial correlation coefficient**
  - blomCOP, 31

* **medial correlation**
  - blomCOP, 31

* **median regression**
  - med.regressCOP, 202
  - med.regressCOP2, 205

* **multivariate**
  - copBasic-package, 4

* **mutually exclusive or condition**
  - duCOP, 80

* **mutually exclusive or**
  - duCOP, 80

* **package copula (comparison to)**
  - AMHcop, 15
  - gEVcop, 118
  - GHcop, 119
  - med.regressCOP, 202
  - mleCOP, 206
  - spectralmeas, 255
  - taildepCOP, 270
<table>
<thead>
<tr>
<th><strong>tEVcop, 280</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>permutation asymmetry</strong></td>
</tr>
<tr>
<td>breveCOP, 37</td>
</tr>
<tr>
<td>joeskewCOP, 152</td>
</tr>
<tr>
<td><strong>polar</strong></td>
</tr>
<tr>
<td>psepolar, 223</td>
</tr>
<tr>
<td>spectralmeas, 255</td>
</tr>
<tr>
<td><strong>product copula</strong></td>
</tr>
<tr>
<td>P, 212</td>
</tr>
<tr>
<td><strong>pseudo-polar representation</strong></td>
</tr>
<tr>
<td>psepolar, 223</td>
</tr>
<tr>
<td><strong>quantile regression</strong></td>
</tr>
<tr>
<td>med.regressCOP, 202</td>
</tr>
<tr>
<td>med.regressCOP2, 205</td>
</tr>
<tr>
<td>qua.regressCOP, 226</td>
</tr>
<tr>
<td>qua.regressCOP.drew, 228</td>
</tr>
<tr>
<td>qua.regressCOP2, 230</td>
</tr>
<tr>
<td><strong>radial asymmetry</strong></td>
</tr>
<tr>
<td>joeskewCOP, 152</td>
</tr>
<tr>
<td><strong>relative entropy</strong></td>
</tr>
<tr>
<td>kullCOP, 179</td>
</tr>
<tr>
<td><strong>return period (conditional)</strong></td>
</tr>
<tr>
<td>copBasic-package, 4</td>
</tr>
<tr>
<td><strong>return period (secondary)</strong></td>
</tr>
<tr>
<td>copBasic-package, 4</td>
</tr>
<tr>
<td>kfuncCOP, 167</td>
</tr>
<tr>
<td><strong>semi-correlation coefficient</strong></td>
</tr>
<tr>
<td>semicorCOP, 243</td>
</tr>
<tr>
<td><strong>semi-correlation</strong></td>
</tr>
<tr>
<td>semicorCOP, 243</td>
</tr>
<tr>
<td><strong>shuffle (in Examples)</strong></td>
</tr>
<tr>
<td>asCOP, 18</td>
</tr>
<tr>
<td><strong>spectral measure</strong></td>
</tr>
<tr>
<td>spectralmeas, 255</td>
</tr>
<tr>
<td><strong>stable tail dependence function</strong></td>
</tr>
<tr>
<td>stabtaildepf, 259</td>
</tr>
<tr>
<td><strong>symmetric Gumbel copula</strong></td>
</tr>
<tr>
<td>GHcop, 119</td>
</tr>
<tr>
<td><strong>symmetric Gumbel–Hougaard copula</strong></td>
</tr>
<tr>
<td>GHcop, 119</td>
</tr>
<tr>
<td><strong>symmetric Gumbel-Hougaard copula</strong></td>
</tr>
<tr>
<td>GHcop, 119</td>
</tr>
<tr>
<td><strong>t-EV copula</strong></td>
</tr>
<tr>
<td>tEVcop, 280</td>
</tr>
<tr>
<td><strong>tail dependence</strong></td>
</tr>
<tr>
<td>blomCOPss, 34</td>
</tr>
<tr>
<td>stabtaildepf, 259</td>
</tr>
<tr>
<td>taildepCOP, 270</td>
</tr>
<tr>
<td><strong>triangle (in Examples)</strong></td>
</tr>
<tr>
<td>asCOP, 18</td>
</tr>
<tr>
<td><strong>visualization</strong></td>
</tr>
<tr>
<td>densityCOP, 62</td>
</tr>
<tr>
<td>densityCOPplot, 64</td>
</tr>
<tr>
<td>diagCOP, 76</td>
</tr>
<tr>
<td>diagCOPatf, 78</td>
</tr>
<tr>
<td>level.curvesCOP, 196</td>
</tr>
<tr>
<td>level.curvesCOP2, 198</td>
</tr>
<tr>
<td>level.setCOP, 199</td>
</tr>
<tr>
<td>level.setCOP2, 200</td>
</tr>
<tr>
<td>qua.regressCOP.drew, 228</td>
</tr>
<tr>
<td>sectionCOP, 240</td>
</tr>
<tr>
<td>simcomposite3COP, 245</td>
</tr>
<tr>
<td>simcompositeCOP, 247</td>
</tr>
<tr>
<td>simCOP, 249</td>
</tr>
</tbody>
</table>

```
| aicCOP, 6, 7, 13, 21, 208, 239, 240, 262, 265 |
| bicCOP, 6, 7, 14, 20, 208, 239, 240, 262, 265 |
| blmoms, 6, 7, 22, 155, 185, 186, 188, 283 |
| blomatrixCOP, 27, 36 |
| blomatrixCOPdec, 32 |
| blomatrixCOPdec(balomatrixCOP), 27 |
| blomatrixCOPiqr, 32 |
| blomatrixCOPiqr(balomatrixCOP), 27 |
| blomCOP, 6, 7, 27, 28, 31, 35, 36, 85, 115, 128, 138, 142, 144, 153, 155, 204, 238, 264, 269, 273, 279, 293, 294 |
| blomCOPss, 28, 31, 32, 34, 269, 272, 273 |
| breveCOP, 9, 37, 43, 46, 48, 49, 51, 132 |
| CLcop, 4, 5, 39 |
| coCOP, 6, 9, 40, 54, 80, 82, 160, 266 |
| composite1COP, 9, 38, 42, 43, 45, 46, 48, 49, 51, 52, 54, 57, 132, 221, 222, 246–248 |
| composite2COP, 9, 38, 42, 43, 44, 46–49, 51, 52, 132, 134, 158, 192, 221, 222, 246–248 |
| composite3COP, 9, 38, 43, 45, 46, 49, 51, 52, 132, 195, 221, 222, 245, 246 |
| concordCOP, 114, 126, 127 |
| concordCOP(tauCOP), 277 |
| convex2COP, 9, 38, 48, 51, 52, 117, 126, 221, 222 |
```
convexCOP, 9, 38, 43, 46, 48, 49, 50, 52, 132, 221, 222
COP, 6–10, 19, 34, 38, 41–43, 46, 48, 49, 51, 52, 60, 62, 77, 82, 132, 145, 164, 189, 222, 242, 244, 251, 253, 265, 266, 268, 273, 276, 284
copBasic-package, 4
copBasic.fitpara, 56
copBasic.fitpara.beta, 263
COPinv, 7, 52, 59, 62, 168, 196, 197
COPinv2, 7, 52, 60, 61, 198, 199
densityCOP, 6, 23, 62, 64, 66, 150, 180, 181, 183, 206, 207, 210, 216, 286, 290
densityCOPplot, 6, 62, 63, 64, 287, 289
derCOP, 7, 9, 67, 69, 70, 74, 93, 121, 168, 174, 221, 241, 252, 277–279
derCOP2, 7, 9, 67, 68, 69, 75, 95, 121, 174, 221, 241, 252, 277–279
derCOPinv, 7, 68, 71, 95, 174, 202, 226, 227, 249, 251, 252
derCOPinv2, 7, 70, 75, 98, 174, 205, 206, 230, 231, 251, 252
diagCOP, 6, 76, 79, 86, 126, 242, 268
diagCOPatf, 6, 77, 78, 160, 162, 164, 166
diagCOPinv(diagCOPatf), 78
duCOP, 6, 9, 10, 40, 41, 54, 78, 80, 158, 160–164, 166, 266

EMPIRcop, 6, 14, 20, 21, 28, 32, 35, 36, 83, 88–90, 93, 95, 96, 99, 137, 144, 169, 239, 240, 253, 289
EMPIRcopdf, 88, 90, 93, 95, 96, 99, 134
EMPIRgrid, 6, 89, 93, 95, 96, 98, 99, 108–110
EMPIRgridder, 93, 95, 96
EMPIRgridder2, 93, 94, 96, 98, 99
EMPIRgridderinv, 95, 99, 101, 107–110
EMPIRgridderinv2, 9, 88, 99, 100, 107
EMPIRmed.regress, 99, 101, 107
EMPIRmed.regress2, 99, 100, 100, 101, 107
EMPIRqua.regress, 99, 100, 100, 107
EMPIRqua.regress2, 100, 101, 106
EMPIRsim, 6, 107, 109, 110, 250
EMPIRsimv, 6, 107, 109, 109

FGMcop, 5, 72, 111, 222
FGMcop (FGMcop), 111
footCOP, 6, 7, 32, 85, 113, 128, 138, 153, 155, 238, 279, 294

FRECHETcop, 5, 38, 49, 115
gEVcop, 5, 118
GHcop, 5, 7, 10, 11, 54, 57, 119, 131, 139, 152, 158, 163, 167, 171, 172, 253, 256, 281, 287
giniCOP, 6, 7, 32, 57, 58, 85, 115, 126, 138, 142–144, 153, 155, 238, 244, 279, 293, 294
GLcop, 5, 124, 129, 139, 167, 264, 281
GLEVcop (GLcop), 129
GLPMcop (GLcop), 129
glueCOP, 9, 38, 43, 46, 48, 49, 51, 52, 131, 221, 222
gridCOP, 6, 133
hoeffCOP, 6, 7, 32, 57, 85, 114, 115, 128, 135, 153, 155, 236, 238, 279, 294
HRcop, 5, 57, 58, 124, 131, 139, 167, 256, 280, 281
isCOP.LTD, 8, 140, 144, 149
isCOP.persym, 8, 22, 37, 120, 141, 146, 153, 184
isCOP.PQO, 8, 140, 141, 142, 148, 149, 292, 293
isCOP.radset, 8, 120, 142, 145, 153
isCOP.RTI, 8, 141, 144, 147
isfuncCOP, 63, 149
JOcopBS, 5, 150
JOcopBB4 (GLcop), 129
joeskewCOP, 7, 32, 85, 120, 128, 138, 152, 238, 279, 283, 294
joint.curvesCOP, 7, 78, 79, 82, 157, 162, 166, 197
joint.curvesCOP2, 7, 160, 160, 199
jointCOP, 6, 9, 78, 79, 82, 160, 162, 162
kfuncCOP, 6, 10, 167, 174, 175, 178, 278, 279
kfuncCOPinv, 6, 167, 169, 174, 174, 175
kfuncCOPmom, 6, 175
kfuncCOPmom (kfuncCOPmom), 176
kfuncCOPmoms, 6, 175, 176, 214
kmeasCOP, 6
kmeasCOP (kfuncCOP), 167
kullCOP, 8, 63, 179, 189, 210, 258, 264, 265, 285, 287, 288, 290
kullCOPint (kullCOP), 179
INDEX

lcomCOP, 6, 7, 22, 27, 184, 189, 245, 247, 249
lcomCOPv, 8, 186, 187, 289
lcomms2: ABcop2parameter, 190
lcomms2: ABKcop2parameter, 193
level.curvesCOP, 7, 59, 60, 62, 85, 86, 157, 159, 160, 166, 196, 199, 200
level.curvesCOP2, 7, 59–62, 160, 162, 197, 198, 200, 201
level.setCOP, 7, 197, 199, 201
level.setCOP2, 7, 199, 200, 200
LpCOP, 6
LpCOP (hoeCOP), 135
LpCOPpermsym (hoeCOP), 135
LpCOPradsym (hoeCOP), 135

med. regressCOP, 7, 11, 31, 202, 206, 226, 227
med. regressCOP2, 7, 205, 205, 230, 231
mleCOP, 6, 11, 63, 112, 206, 257, 290

N4212cop, 5, 170, 171, 211, 225, 226, 261
nuskeW (joeskewCOP), 6, 7, 57, 123, 136, 159
nuskeCOP (joeskewCOP), 152
nustarCOP (joeskewCOP), 152

PACop (PAREToCOP), 214
PAREToCOP, 214
PLACETCOP, 146, 170, 171, 215, 217, 219, 220, 264, 275, 287
PLACETTcop, 216, 217, 220
PLACETTpar, 216, 217, 220
PLACETTs, 216, 217, 219
PLcop, 5, 152, 217, 219, 293
PLcop (PLACETTcop), 215
PLpar, 216
PLpar (PLACETTpar), 217
prod2COP, 9, 52, 221
psepolar, 8, 223, 255, 256, 258, 262
PSP, 5, 9, 35, 81, 146, 211, 225, 272, 287
qua.regressCOP, 8, 202, 205, 226, 229
qua.regressCOP.draw, 202, 205, 206, 227, 228
qua.regressCOP2, 8, 205, 206, 229, 230
rCOP, 6
rCOP (simCOP), 249
ReineckeWells, 231
ReineckeWells, 232
RFcop, 5, 52, 233
rhobeveCOP, 120, 124, 234
rmseCOP, 6, 7, 11, 239, 265
sectionCOP, 8, 77, 240
semicorCOP, 6, 7, 243
simcomposite2COP (simcompositeCOP), 247
simcomposite3COP, 48, 245, 249
simcompositeCOP, 190, 192, 193, 195, 247, 247
simCOP, 6, 7, 9, 48, 57, 58, 63, 65, 66, 86, 107, 109, 159, 192, 195, 220, 249, 251, 254, 287–290
simCOPmicro, 6, 7, 53, 54, 109, 220, 250, 251, 251, 265, 266
simCOPv, 207
simCOPv (simCOPmicro), 251
spectralmeas, 8, 225, 255, 260, 262
stabstaildepf, 8, 225, 258, 259
statTn, 6, 7, 14, 21, 239, 262, 290
surCOP, 5, 6, 8, 9, 41, 53, 54, 80, 82, 135, 145, 221, 252, 265, 267, 285
surfCOP, 6, 9, 34, 54, 265, 266, 267, 275
tailconCOP, 8, 268, 273, 276
taildePCOP, 7, 8, 35, 36, 116, 120, 167, 171, 211, 215, 253, 269, 270, 276, 288, 289
tailordCOP, 8, 269, 273, 274
tEVcop, 5, 119, 124, 131, 139, 280
uvlmoms, 7, 27, 32, 85, 128, 136, 138, 152, 186, 238, 279, 283, 294
uvskew, 7, 8, 155
uvskew (uvlmoms), 283
vuongCOP, 8, 181, 189, 209, 250, 264, 265, 271, 285
wolfCOP, 6, 7, 32, 85, 115, 128, 138, 143, 153, 155, 238, 279, 292