Package ‘igcop’

October 13, 2021

Title Computational Tools for the IG and IGL Copula Families

Version 1.0.1

Description Compute distributional quantities for an Integrated Gamma (IG) or Integrated Gamma Limit (IGL) copula, such as a cdf and density. Compute corresponding conditional quantities such as the cdf and quantiles. Generate data from an IG or IGL copula.

License MIT + file LICENSE

Suggests testthat, knitr, rmarkdown, tibble, covr, ggplot2

Encoding UTF-8

RoxygenNote 7.1.2

VignetteBuilder knitr

Imports stats, vctrs, Rcpp, rlang

LinkingTo Rcpp

NeedsCompilation yes

Author Vincenzo Coia [aut, cre], Harry Joe [aut]

Maintainer Vincenzo Coia <vincenzo.coia@gmail.com>

Repository CRAN

Date/Publication 2021-10-13 10:20:06 UTC

R topics documented:

.onUnload .................................................. 2
.u ............................................................. 2
check_alpha .................................................. 3
dig_vec ...................................................... 3
formals_to ................................................... 4
igcop ......................................................... 4
igl_gen_vec .................................................. 5
interp_gen_inv ............................................. 6
pcondig21 .................................................... 7
qcondigl ..................................................... 8
Index

.onUnload  Clean up DLL

Description
As recommended in the "Compiled Code" chapter of the book "R packages" (Version 2) by Hadley Wickham and Jenny Bryan.

Usage
.onUnload(libpath)

Arguments

libpath  Argument

.u  Test data

Description
Internal data used in the test scripts. .u and .v are vectors of matching length containing values between 0 and 1 and reasonably cover the unit square. .cpar is a list of IG copula parameter pairs \( c(\theta, \alpha) \), and .\theta and .\alpha are the corresponding (unique) individual values.

Usage
.u
.v
.cpar
\.theta
\.alpha

Format
Everything is a numeric vector, except .cpar, which is a list of bivariate numeric vectors. .u and .v are of matching length; the rest are not intended to have matching lengths.
An object of class numeric of length 17.
An object of class list of length 25.
An object of class numeric of length 14.
An object of class numeric of length 11.
check_alpha

Check validity of copula parameters

Description
Ensures input values are non-negative.

Usage
check_alpha(alpha)

check_theta(theta)

Arguments
alpha Values of alpha to check.
theta Values of theta to check.

Value
An error if any theta or alpha is negative; an invisible value otherwise. NA values do not throw an error.

dig_vec
Select IG copula quantities: matching inputs

Description
The density function, 1|2 conditional cdf, and 1|2 conditional quantile function of the IG copula family. Inputs need to be vectors of the same length. These functions are called by the R functions of the same name, without the _vec suffix.

Usage
dig_vec(u, v, theta, alpha)
pcondig12_vec(u, v, theta, alpha)
qcondig12_vec(p, v, theta, alpha)

Arguments
u, v Copula arguments. Vector of values between 0 and 1.
theta, alpha IG copula parameters. Vector of positive values.
p Function inverse argument. Vector of values between 0 and 1.
Details

The qcondig12() function needs its own Newton Raphson algorithm. It also needs access to some version of pcondig12() and dig(). So, these three functions are coded up in C++, each with a scalar and vector pair of functions.

Note

If calling these functions manually, make sure each input are vectors of a common length.

See Also

dig(), pcondig12(), and qcondig12(): and igl_gen_vec() and family.

formals_to

Send arguments to a function after vectorizing

Description

When used within a (encapsulating) function, formals_to recycles the inputs of the encapsulating function so that they are vectors of the same length, and then sends these updated arguments to some specified function.

Usage

formals_to(.fn)

Arguments

.fn The function you want to send the recycled arguments to.

Value

The function .fn evaluated with the arguments given in the encapsulating function.

igcop

igcop: Computational Tools for the IG and IGL Copula Families

Description

Compute distributional quantities for an Integrated Gamma (IG) or IG Limit (IGL) copula, such as a cdf and density, along with conditional quantities such as the cdf, quantiles, and densities. Generate data from a copula.

Usage

Access copula quantities by starting with the p, d, q, or r prefixes, followed by the copula name – either ig or igl, or their conditional versions, condig or condigl.
igl_gen_vec

IG/IGL Generators and Related Functions: matching inputs

Description
These are the psi, H, and kappa functions of the IG and IGL copula families, but with inputs needing to be vectors of the same length. These functions are called by the R functions of the same name, without the _vec suffix.

Usage

\texttt{igl_gen_vec(x, alpha)}
\texttt{igl_gen_D_vec(x, alpha)}
\texttt{igl_gen_inv_vec(p, alpha)}
\texttt{igl_kappa_vec(x, alpha)}
\texttt{igl_kappa_D_vec(x, alpha)}
\texttt{igl_kappa_inv_vec(p, alpha)}
\texttt{interp_gen_vec(x, eta, alpha)}
\texttt{interp_gen_inv_vec(p, eta, alpha)}
\texttt{interp_kappa_vec(x, eta, alpha)}
\texttt{interp_kappa_inv_vec(p, eta, alpha)}

Arguments

\begin{itemize}
\item \texttt{x} Function argument. Vector of non-negative values.
\item \texttt{p} Function inverse argument. Vector of values between 0 and 1.
\item \texttt{eta, alpha} Function parameters. Vector of positive values.
\end{itemize}

Note
If calling this function manually, make sure each input are vectors of a common length.

See Also
\texttt{igl_gen()} and family; \texttt{dig_vec()}, \texttt{pcondig12_vec()}, and \texttt{qcondig12_vec()}. 
Description

These are the psi, H, and kappa functions of the IG and IGL copula families.

Usage

interp_gen_inv(p, eta, alpha)
interp_kappa(x, eta, alpha)
interp_kappa_inv(p, eta, alpha)
interp_gen(x, eta, alpha)
igl_kappa(x, alpha)
igl_kappa_D(x, alpha)
igl_kappa_inv(p, alpha)
igl_gen(x, alpha)
igl_gen_D(x, alpha)
igl_gen_inv(p, alpha)

Arguments

p Function inverse argument. Vector of values between 0 and 1.
eta, alpha Function parameters. Vector of positive values.
x Function argument. Vector of non-negative values.

Details

Kappa function and its relatives have prefix igl_kappa; Psi function and its relatives have prefix igl_gen; Interpolating function H with either kappa or psi has igl prefix replaced with interp. Relatives of these functions: suffix inv indicates inverse; suffix D represents function derivative, and D1 derivative with respect to the first argument. Suffix _vec indicates that the entries must be vectors of the same length; _single means entries must be scalars.

Value

The function values, as a vector.
Note

Inputs must be recyclable via \texttt{vctrs::vec\_recycle\_common()}.

\textbf{IG Copula Family Functions}

\textbf{Description}

Functions related to the IG copula family, denoted by 'ig'.

\textbf{Usage}

\begin{verbatim}
pcondig21(v, u, theta, alpha)
qucondig21(p, u, theta, alpha)
qucondig(p, u, theta, alpha)
pcondig(v, u, theta, alpha)
pcondig12(u, v, theta, alpha)
qucondig12(p, v, theta, alpha)
dig(u, v, theta, alpha)
logdig(u, v, theta, alpha)
pig(u, v, theta, alpha)
rig(n, theta, alpha)
\end{verbatim}

\textbf{Arguments}

- \textit{v, u} Vectors of values between 0 and 1 representing values of the first and second copula variables.
- \textit{theta} Parameter of the IG copula family. Vectorized; >0.
- \textit{alpha} Parameter of the IG copula family. Vectorized; >0.
- \textit{p} Vector of quantile levels between 0 and 1 to evaluate a quantile function at.
- \textit{n} Positive integer. Number of observations to randomly draw.

\textbf{Value}

Numeric vector of length equal to the length of the input vector(s).
Note

Inputting two vectors greater than length 1 is allowed, if they’re the same length. Also, qcondig21 and pcondig21 are the same as qcondig and pcondig – they’re the distributions of variable 2 given 1.

Examples

```r
u <- runif(10)
v <- runif(10)
pig(u, v, theta = 5, alpha = 1)
dig(u, v, theta = 2, alpha = 2)
logdig(u, v, theta = 2, alpha = 2)
pcondig21(v, u, theta = 3, alpha = 6)
qcondig21(v, u, theta = 3, alpha = 6)
pcondig12(u, v, theta = 3, alpha = 6)
qcondig12(u, v, theta = 3, alpha = 6)
rig(10, theta = 3, alpha = 3)

# log density available for extra precision
log(dig(0.1, 0.1, 2.5, 12.3)) == logdig(0.1, 0.1, 2.5, 12.3)
```

---

**qcondigl**

*IGL Copula Family Functions*

**Description**

Functions related to the IGL copula family, denoted by ‘igl’.

**Usage**

- `qcondigl(p, u, alpha)`
- `pcondigl(v, u, alpha)`
- `qcondigl21(p, u, alpha)`
- `pcondigl21(v, u, alpha)`
- `pcondigl12(u, v, alpha)`
- `qcondigl12(p, v, alpha)`
- `digl(u, v, alpha)`
- `pigl(u, v, alpha)`
- `rigl(n, alpha)`
- `logdigl(u, v, alpha)`
**Arguments**

- **p** Vector of quantile levels between 0 and 1 to evaluate a quantile function at.
- **u, v** Vectors of values between 0 and 1 representing values of the first and second copula variables.
- **alpha** Single numeric >0; corresponds to parameter $\alpha$ in the IGL copula family.
- **n** Positive integer. Number of observations to randomly draw.

**Value**

Numeric vector of length equal to the length of the input vector(s).

**Note**

Inputting two vectors greater than length 1 is allowed, if they’re the same length. Also, `qcondigl21` and `pcondigl21` are the same as `qcondigl1` and `pcondigl1` – they are the distributions of variable 2 given 1.

**Examples**

```r
set.seed(1)
u <- runif(10)
v <- runif(10)
pigl(u, v, alpha = 1)
digl(u, v, alpha = 2)
logdigl(u, v, alpha = 0.4)
pcondigl21(v, u, alpha = 6)
qundigl21(v, u, alpha = 6)
pcondigl22(u, v, alpha = 6)
qundigl22(u, v, alpha = 6)
rigl(10, alpha = 3)
```
Index

* datasets
  .u, 2
  .alpha (.u), 2
  .cpar (.u), 2
  .onUnload, 2
  .theta (.u), 2
  .u, 2
  .v (.u), 2

check_alpha, 3
check_theta (check_alpha), 3

dig (pcondig21), 7
dig_vec, 3
digl (qcondig1), 8

formals_to, 4

igcop, 4
igl_gen (interp_gen_inv), 6
igl_gen_D (interp_gen_inv), 6
igl_gen_D_vec (igl_gen_vec), 5
igl_gen_inv (interp_gen_inv), 6
igl_gen_inv_vec (igl_gen_vec), 5
igl_gen_vec, 5
igl_kappa (interp_gen_inv), 6
igl_kappa_D (interp_gen_inv), 6
igl_kappa_D_vec (igl_gen_vec), 5
igl_kappa_inv (interp_gen_inv), 6
igl_kappa_inv_vec (igl_gen_vec), 5
igl_kappa_vec, 5
interp_gen (interp_gen_inv), 6
interp_gen_inv (igl_gen_vec), 5
interp_gen_inv_vec (igl_gen_vec), 5
interp_gen_vec (igl_gen_vec), 5
interp_kappa (interp_gen_inv), 6
interp_kappa_inv (interp_gen_inv), 6
interp_kappa_inv_vec (igl_gen_vec), 5
interp_kappa_vec (igl_gen_vec), 5
logdig (pcondig21), 7

logdigl (qcondig1), 8
pcondig (pcondig21), 7
pcondig12 (pcondig21), 7
pcondig12_vec (dig_vec), 3
pcondig21, 7
pcondigl (qcondig1), 8
pcondigl12 (qcondig1), 8
pcondigl21 (qcondig1), 8
pigl (pcondig21), 7
pigl (qcondig1), 8
qcondig (pcondig21), 7
qcondig12 (pcondig21), 7
qcondig12_vec (dig_vec), 3
qcondig21 (pcondig21), 7
qcondig1, 8
qcondig12 (qcondig1), 8
qcondigl21 (qcondig1), 8
rill (pcondig21), 7
rill (qcondig1), 8