Package ‘PBC’

February 19, 2015

Type Package
Title Product of Bivariate Copulas (PBC)
Version 1.2
Date 2014-04-30
Author Van Trung Pham and Gildas Mazo, with contributions from R Core team, Nash, Zhu, Byrd, Lu-Chen and Nocedal
Maintainer Van Trung Pham <trungpv88@gmail.com>
Depends R (>= 3.0.0), igraph
Imports Rcpp (>= 0.10.3), copula, methods
Description This package provides tools for modeling copulas with the PBC model, a class of multivariate copulas based on products of bivariate copulas (G. Mazo G, S. Girard and F. Forbes, 2013). The likelihood is computed thanks to a message-passing algorithm on graphs (J. C. Huang and N. Jojic, 2010).
License GPL (>= 2)
LinkingTo Rcpp
Repository CRAN
NeedsCompilation yes
Date/Publication 2014-04-30 08:23:27

R topics documented:

PBC-package .................................................. 2
mpAlgo .................................................. 3
PBC .................................................. 4
PBC-class .................................................. 5
pbcModels .................................................. 7
pbcOptim .................................................. 8

Index 11
Description

This package provides tools for modeling copulas with the PBC model, a class of multivariate copulas based on products of bivariate copulas (G. Mazo G, S. Girard and F. Forbes). The likelihood is computed thanks to a message-passing algorithm on graphs (J. C. Huang and N. Jojic, 2010).

Details

Package: PBC
Type: Package
License: GPL (>= 3)
Depends: Rcpp (>= 0.10.3), igraph, copula, methods

This package provides

1. **pbcMclass**: function to create a PBC model.
2. **pbc**: random number generation, distribution function and density for the PBC model.
3. **mpAlgo**: message-passing algorithm to compute the likelihood and its gradient (with respect to parameter vector) for the PBC model.
4. **pbcOptim**: maximum likelihood estimation for a PBC copula.

Author(s)

Van Trung Pham <trungpv88@gmail.com>
Gildas Mazo <gildas.mazo@inria.fr>
Please let the authors know about bugs or suggestions!

References


Examples

```r
## Set the underlying graphical structure for the PBC model
g <- graph.formula(X1~X2, X2~X3, X3~X4, X4~X5, simplify = FALSE)

## Create the PBC object with Gumbel linking family
```
mpAlgo

myPBC <- pbc(g, model="gumbel") # or:
myPBC <- pbcGumbel(g)

## Plot PBC graph
pbcPlot(myPBC)

## Generate n observations from the model
theta <- 1:4
n <- 100
data <- rPBC(100, theta, myPBC)

## Estimate the parameter vector
init <- rep(5, 4) # the 'par' argument of \code{\link{optim}}.
# it's better if you can provide an estimate based on pairwise likelihood to
# increase the chances to get a good minimizer.

## Use \code{\link{pbcoptim}}
fitPBC <- pbcoptim(init, data, myPBC, method="BFGS")
fitPBC # estimate

## You may use \code{\link{optim}} instead
fn <- function(theta) -sum(log((dPBC(data, theta, myPBC)))) # -log likelihood
gr.temp <- function(u, theta) mpAlgo(myPBC, u, theta)@gradient # gradient of likelihood
gr <- function(theta)( # gradient of -log likelihood
  ap <- t(apply(data, 1, gr.temp, theta=theta))
  ap2 <- dPBC(data, theta, myPBC)
  apply(-ap*ap2, 2, sum)
)
fitPBC2 <- optim(par=init, fn=fn, gr=gr, method="BFGS")
fitPBC2

---

**Description**

The function mpAlgo implements a message-passing algorithm to compute the likelihood and its gradient (with respect to parameter vector) for the PBC model.

**Usage**

mpAlgo(pbcObj, u, theta, output="both", ...)

**Arguments**

- **pbcObj**
  - an object of class PBC.
- **u**
  - point at which is evaluated the density and the gradient.
- **theta**
  - parameter vector.
- **output**
  - desired output: "density" or "gradient" or "both". The default is "both".
- **...**
  - currently nothing.
Details

`mpalgo` implements the algorithm of Table 1 in J. C. Huang and N. Jojic, 2010, for the special case of a PBC model.

Value

An object of class `PBC`. The likelihood is found in the slot `@density` and its gradient in the slot `@gradient` (see example below).

References


See Also

`PBC-class`

Examples

```r
## Example with Gumbel linking family
G <- graph.formula(X1-X3,X2-X3,X3-X4,X4-X5,X4-X6,X6-X7,X6-X8,simplify = FALSE)
pbcGumbel <- pbc(g, model="gumbel")
u <- runif(8)
theta <- 1/runif(7)
pbcOut <- mpalgo(pbcGumbel, u, theta)
pbcOut@density
pbcOut@gradient

## Example with a user defined distribution (must contain 'x' and 'y')
f <- expression(exp(-(log(x))^(theta)*(-log(y))^(theta)*(1/theta))) # Gumbel
pbcUser <- pbc(g, model=f)
pbcOut2 <- mpalgo(pbcUser, u, theta)
pbcOut2@density
pbcOut2@gradient
```

---

**PBC**

*Random number generation, distribution function and density for the PBC model*

**Description**

Random generation (rPBC), distribution function (pPBC) and density (dPBC) for the PBC model.

**Usage**

- `rPBC(n, theta, pbcObj, ...)`
- `pPBC(u, theta, pbcObj, ...)`
- `dPBC(u, theta, pbcObj, ...)"
Arguments

pbcObj  an object of class PBC.
n  number of observations to be generated.
theta parameter vector.
u  a vector or a matrix at which the function needs to be evaluated.
... currently nothing.

Details

The density is computed via the function mpAlgo. The parameter vector has length equal to the number of variables minus one. If u is a matrix, it has dimension (n,d) where n is the number of vectors the function is evaluated at, and d is the number of variables in the model.

Value

rPBC() generates random data, pPBC() computes the distribution function and dPBC() computes the density.

See Also

mpAlgo,PBC-class.

Examples

```r
## set a parameter vector
theta <- runif(4)
## construct the graph
g <- graph.formula(X1~X4, X4~X2, X2~X3, X4~X5, simplify = FALSE)
## create the PBC object with linking family "AMH"
myPBC.AMH <- pbc(g, model="amh")
## alternatively:
## myPBC.AMH <- pbcAMH(g)
## Generate 5 random data vectors
r1 <- rPBC(5, theta, myPBC.AMH)
## Compute the distribution function
p1 <- pPBC(r1, theta, myPBC.AMH)
## Compute the density
d1 <- dPBC(r1, theta, myPBC.AMH)
```

Class "PBC" for the PBC model

Description

The class "PBC" provides a function to create PBC objects.
Usage

```
pbc(g, model, ...)
```

Arguments

- `g`: an `igraph` object.
- `model`: a family of copulas among "gumbel", "fgm", "frank", "normal", "amh", "joe".
- `...`: currently nothing.

Value

An object of class PBC.

Slots

- `graph`: a graph of class `igraph` describing the PBC where the nodes represent the variables and the edges represent the linking bivariate copulas.
- `root`: the root for the message-passing algorithm (center of the graph associated to the PBC model).
- `nIteration`: number of iterations for the message-passing algorithm.
- `binmat`: a matrix encoding the links between the variables (represented by nodes) and the bivariate copulas (represented by edges) in the graph. An element of the matrix is set to 1 if a variable and a bivariate copula are adjacent, 0 otherwise.
- `model`: a copula family to link the variable nodes. Choices include "gumbel", "fgm", "frank", "normal", "amh", "joe" (see `pbcModels` for details).
- `density, gradient`: the density and gradient (with respect to the parameter vector) obtained from the message-passing algorithm.

For more details about the linking copula families, see `pbcModels`.

Methods

- `initialize` signature("PBC"): set the slot values, construct the graph and compute the encoding matrix.
- `pbcPlot` signature("PBC"): display the graph of the PBC model.

See Also

Linking copula families are detailed in `pbcModels`.

Examples

```
## PBC class information
showClass("PBC")
## Create a PBC object with linking family "Gumbel"
g <- graph.formula(X1~X3,X2~X3,X3~X4,X4~X5,simplify = FALSE)
bcObj <- pbc(g, model="gumbel")
```
Linking copula families for the PBC model

Description
Linking copula families implemented in the PBC package.

Usage
- pbcGumbel(graph)
- pbcFGM(graph)
- pbcFrank(graph)
- pbcNormal(graph)
- pbcAMH(graph)
- pbcJoe(graph)

Arguments
- graph: the graph (of class igraph) associated to the PBC copula.

Details
A pair \((U_i, U_j)\) of the PBC model has copula
\[
C_{ij}(u, v) = u^{1-1/n_i} \ast v^{1-1/n_j} \ast D_{ij}(u^{1/n_i}, v^{1/n_j}),
\]
where \(n_i\) and \(n_j\) are the number of neighbors in the graph for the variables \(U_i\) and \(U_j\) respectively (G. Mazo G, S. Girard and F. Forbes). The copula families implemented for \(D_{ij}(u, v)\) are given below.

- pbcGumbel: family of Gumbel copulas:
  \[
  \exp(-((-\ln(u))^\theta + (-\ln(v))^\theta)^{1/\theta})
  \]
  with \(\theta \in [1, \infty)\).

- pbcFGM: family of Farlie-Gumbel-Morgenstern (FGM) copulas:
  \[
  u \ast v \ast (1 + \theta \ast (1 - u) \ast (1 - v))
  \]
  with \(\theta \in [-1, 1]\).

- pbcFrank: family of Frank copulas:
  \[
  -\ln(1 + (\exp(-\theta \ast u) - 1) \ast (\exp(-\theta \ast v) - 1)/(\exp(-\theta) - 1))/\theta
  \]
  with \(\theta \in (0, \infty)\).

- pbcNormal: family of normal copulas:
  \[
  \exp((\theta \ast q(u))^2 + (\theta \ast q(v))^2 - 2 \ast \theta \ast q(u) \ast q(v))/(2 \ast (1 + \theta^2))/(1 - \theta^2)^{0.5}
  \]
  with \(\theta \in [-1, 1]\), \(q\) is the inverse of the standard normal distribution function.
pbcAMH: Family of Ali-Mikhail-Haq (AMH) copulas:

\[ u \ast v / (1 - \theta \ast (1 - u) \ast (1 - v)) \]

with \( \theta \in [0, 1) \).

pbcJoe: Family of Joe copulas:

\[ 1 - ((1 - u)^{\theta} + (1 - v)^{\theta} - (1 - u)^{\theta} \ast (1 - v)^{\theta})^{1/\theta} \]

with \( \theta \in [1, \infty) \).

Value

A "PBC" object.

References


See Also

pbc

Examples

```r
## Example for the FGM family
graph <- graph.formula(X1-X2,X2-X3, simplify = FALSE)
## Create a PBC object
fgmObject <- pbcFGM(graph)
## alternatively
g <- graph.formula(X1-X2,X2-X3, simplify = FALSE)
fgmObject <- pbc(g, model="fgm")
```

pbcOptim

Maximum likelihood estimation for a PBC copula

Description

This function performs maximum-likelihood inference in a PBC copula.

Usage

`pbcOptim(par, data, pbcObj, method, lower = -Inf, upper = Inf, ...)`
Arguments

- **par**: the argument *par* in *optim*.
- **data**: data matrix.
- **pbcObj**: an object of class *PBC*.
- **method**: method for the *optim* function. Two methods are available: Broyden-Fletcher-Goldfarb-Shanno (BFGS) and limited memory BFGS with bounds (L-BFGS-B).
- **lower, upper**: bounds on the variables for the "L-BFGS-B" method.
- **...**: currently nothing.

Details

The R routine *optim* is used to minimize -log likelihood. Compared to brute use of *optim*, *pbcOptim* saves one call to *mpAlgo*. See *optim*.

Value

A numeric vector giving the estimate.

Source

The code for *pbcOptim* is based on that of *optim*. In particular, *pbcOptim* calls `/src/lbfgsb.cpp` which is a slight adaptation of `lbfgsb.c` and `optim.c` part of the R software (http://www.r-project.org/). See *optim* and `?optim`.

The code for BFGS method is based on Pascal code in J.C. Nash, 'Compact Numerical Methods for Computers', 2nd edition, converted by p2c then re-crafted by B.D. Ripley. It is modified by V.T. Pham for *pbcOptim* to increase speed of this function.

The code for L-BFGS-B method is based on Fortran code by Zhu, Byrd, Lu-Chen and Nocedal obtained from Netlib, then modified by V.T. Pham.

See Also

*optim*

Examples

```r
## Example with normal linking family
g <- graph.formula(X1~X4, X4~X2, X2~X3, X4~X5, simplify = FALSE)
pbcNormal <- pbcNormal(g)
theta <- runif(4)
pbcDataNormal <- rPBC(5, theta, pbcNormal)

## estimation
# L-BFGS-B method with Lower and upper bound
init <- rep(.5,4) # it's better if you can provide an estimate based
# on pairwise likelihood to increase the chances to get a good minimizer.

opt = pbcOptim(init, pbcDataNormal, pbcNormal,
```
method="L-BFGS-B", lower=rep(0,4), upper=rep(0.99,4))

# BFGS method
opt2 = pbcOptim(init, pbcDataNormal, pbcNormal, method="BFGS")
<table>
<thead>
<tr>
<th>Topic</th>
<th>models</th>
</tr>
</thead>
<tbody>
<tr>
<td>compute</td>
<td>(PBC-class), 5</td>
</tr>
<tr>
<td>dPBC</td>
<td>(PBC), 4</td>
</tr>
<tr>
<td>draw</td>
<td>(PBC-class), 5</td>
</tr>
<tr>
<td>dxdyphi.norm</td>
<td>(PBC-class), 5</td>
</tr>
<tr>
<td>dxdyphi.student</td>
<td>(PBC-class), 5</td>
</tr>
<tr>
<td>dxf.norm</td>
<td>(PBC-class), 5</td>
</tr>
<tr>
<td>dxf.student</td>
<td>(PBC-class), 5</td>
</tr>
<tr>
<td>getBINMAT</td>
<td>(PBC-class), 5</td>
</tr>
<tr>
<td>getBinMat</td>
<td>(PBC-class), 5</td>
</tr>
<tr>
<td>getBINMAT, method</td>
<td>(PBC-class), 5</td>
</tr>
<tr>
<td>getDxdyf</td>
<td>(PBC-class), 5</td>
</tr>
<tr>
<td>getDxdyf, method</td>
<td>(PBC-class), 5</td>
</tr>
<tr>
<td>getDx.f, method</td>
<td>(PBC-class), 5</td>
</tr>
<tr>
<td>getF</td>
<td>(PBC-class), 5</td>
</tr>
<tr>
<td>getF, method</td>
<td>(PBC-class), 5</td>
</tr>
<tr>
<td>getGradx.dyf</td>
<td>(PBC-class), 5</td>
</tr>
<tr>
<td>getGradx.dyf, method</td>
<td>(PBC-class), 5</td>
</tr>
<tr>
<td>getGradx.f</td>
<td>(PBC-class), 5</td>
</tr>
<tr>
<td>getGradx.f, method</td>
<td>(PBC-class), 5</td>
</tr>
<tr>
<td>getGraf</td>
<td>(PBC-class), 5</td>
</tr>
<tr>
<td>getGraf, method</td>
<td>(PBC-class), 5</td>
</tr>
<tr>
<td>getGraph</td>
<td>(PBC-class), 5</td>
</tr>
<tr>
<td>getGraph, method</td>
<td>(PBC-class), 5</td>
</tr>
<tr>
<td>getModel</td>
<td>(PBC-class), 5</td>
</tr>
<tr>
<td>getModel, method</td>
<td>(PBC-class), 5</td>
</tr>
<tr>
<td>getNIiteration</td>
<td>(PBC-class), 5</td>
</tr>
<tr>
<td>getNIiteration, method</td>
<td>(PBC-class), 5</td>
</tr>
<tr>
<td>getRoot</td>
<td>(PBC-class), 5</td>
</tr>
<tr>
<td>getRoot, method</td>
<td>(PBC-class), 5</td>
</tr>
<tr>
<td>gradx.dyphi.norm</td>
<td>(PBC-class), 5</td>
</tr>
<tr>
<td>gradx.dyphi2.student</td>
<td>(PBC-class), 5</td>
</tr>
<tr>
<td>gradx.norm</td>
<td>(PBC-class), 5</td>
</tr>
<tr>
<td>gradx2.student</td>
<td>(PBC-class), 5</td>
</tr>
<tr>
<td>igraph</td>
<td>6, 7</td>
</tr>
<tr>
<td>igraph-class</td>
<td>(PBC-class), 5</td>
</tr>
<tr>
<td>margin</td>
<td>(PBC-class), 5</td>
</tr>
<tr>
<td>mpAlgo</td>
<td>2, 3, 4, 5</td>
</tr>
<tr>
<td>optim</td>
<td>9</td>
</tr>
<tr>
<td>PBC</td>
<td>2, 4, 4, 8</td>
</tr>
<tr>
<td>pbc</td>
<td>8</td>
</tr>
<tr>
<td>pbc (PBC-class), 5</td>
<td></td>
</tr>
<tr>
<td>PBC-class</td>
<td>5</td>
</tr>
<tr>
<td>PBC-package</td>
<td>2</td>
</tr>
<tr>
<td>pbcAMH</td>
<td>(pbcModels), 7</td>
</tr>
<tr>
<td>pbcFGM</td>
<td>(pbcModels), 7</td>
</tr>
<tr>
<td>pbcFrank</td>
<td>(pbcModels), 7</td>
</tr>
<tr>
<td>pbcGumbel</td>
<td>(pbcModels), 7</td>
</tr>
<tr>
<td>pbcJoe</td>
<td>(pbcModels), 7</td>
</tr>
<tr>
<td>pbcModels</td>
<td>6, 7</td>
</tr>
<tr>
<td>pbcNormal</td>
<td>(pbcModels), 7</td>
</tr>
<tr>
<td>pbcOptim</td>
<td>2, 8</td>
</tr>
<tr>
<td>pbcPlot</td>
<td>(PBC-class), 5</td>
</tr>
<tr>
<td>pbcPlot, method</td>
<td>(PBC-class), 5</td>
</tr>
<tr>
<td>phi.norm</td>
<td>(PBC-class), 5</td>
</tr>
<tr>
<td>phi.student</td>
<td>(PBC-class), 5</td>
</tr>
<tr>
<td>phi.student1</td>
<td>(PBC-class), 5</td>
</tr>
<tr>
<td>pPBC</td>
<td>(PBC), 4</td>
</tr>
<tr>
<td>rPBC</td>
<td>4</td>
</tr>
<tr>
<td>setDensity</td>
<td>(PBC-class), 5</td>
</tr>
<tr>
<td>setDensity, method</td>
<td>(PBC-class), 5</td>
</tr>
<tr>
<td>setGradient</td>
<td>(PBC-class), 5</td>
</tr>
<tr>
<td>setGradient, method</td>
<td>(PBC-class), 5</td>
</tr>
</tbody>
</table>