

# Package ‘PBC’

January 27, 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 Nosedal

**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. Jovic, 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
pbModels . . . . .	7
pbOptim . . . . .	8

<b>Index</b>	<b>11</b>
--------------	-----------

---

PBC-package

*Product of Bivariate Copulas*

---

## 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 packages provides

1. `PBC-class`: 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

G. Mazo G, S. Girard and F. Forbes. A class of high dimensional copulas based on products of bivariate copulas. <http://hal.archives-ouvertes.fr/hal-00910775>. J. C. Huang and N. Jojic. Maximum-likelihood learning of cumulative distribution functions on graphs. *Journal of Machine Learning Research W&CP Series*, 9:342–349, 2010.

## Examples

```
## 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
```

```

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

```

---

mpAlgo

*Message-Passing algorithm*


---

## 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

<code>pbcObj</code>	an object of class PBC.
<code>u</code>	point at which is evaluated the density and the gradient.
<code>theta</code>	parameter vector.
<code>output</code>	desired output: "density" or "gradient" or "both". The default is "both".
<code>...</code>	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**

J. C. Huang and N. Jojic. Maximum-likelihood learning of cumulative distribution functions on graphs. *Journal of Machine Learning Research W&CP Series*, 9:342–349, 2010.  
 G. Mazo G, S. Girard and F. Forbes. A class of high dimensional copulas based on products of bivariate copulas. <http://hal.archives-ouvertes.fr/hal-00910775>.

**See Also**

[PBC-class](#)

**Examples**

```
## 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 dsitribution (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**

pbObj	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**

```
## 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)
```

---

PBC-class

*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 [pbcmModels](#) 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 [pbcmModels](#).

**Methods**

**initialize** signature("PBC"): set the slot values, construct the graph and compute the encoding matrix.

**pbcmPlot** signature("PBC"): display the graph of the PBC model.

**See Also**

Linking copula families are detailed in [pbcmModels](#).

**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)
pbcmObj <- pbcm(g, model="gumbel")
```

## Description

Linking copula families implemented in the **PBC** package.

## Usage

```
pbcmGumbel(graph)
pbcmFGM(graph)
pbcmFrank(graph)
pbcmNormal(graph)
pbcmAMH(graph)
pbcmJoe(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

$$C_{ij}(u, v) = u^{1-1/n_i} * v^{1-1/n_j} * D_{ij}(u^{1/n_i}, v^{1/n_j}),$$

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.

**pbcmGumbel:** family of Gumbel copulas:

$$\exp(-((-\ln(u))^\theta + (-\ln(v))^\theta)^{1/\theta})$$

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

**pbcmFGM:** family of Farlie-Gumbel-Morgenstern (FGM) copulas:

$$u * v * (1 + \theta * (1 - u) * (1 - v))$$

with  $\theta \in [-1, 1]$ .

**pbcmFrank:** family of Frank copulas:

$$-\ln(1 + (\exp(-\theta * u) - 1) * (\exp(-\theta * v) - 1) / (\exp(-\theta) - 1)) / \theta$$

with  $\theta \in (0, \infty)$ .

**pbcmNormal:** family of normal copulas:

$$\exp(((\theta * q(u))^2 + (\theta * q(v))^2 - 2 * \theta * q(u) * q(v)) / (2 * (-1 + \theta^2))) / (1 - \theta^2)^{0.5}$$

with  $\theta \in [-1, 1]$ ,  $q$  is the inverse of the standard normal distribution function.

pbcAMH: Family of Ali-Mikhail-Haq (AMH) copulas:

$$u * v / (1 - \theta * (1 - u) * (1 - v))$$

with  $\theta \in [0, 1]$ .

pbcJoe: Family of Joe copulas:

$$1 - ((1 - u)^\theta + (1 - v)^\theta - (1 - u)^\theta * (1 - v)^\theta)^{1/\theta}$$

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

### Value

A "PBC" object.

### References

G. Mazo G, S. Girard and F. Forbes. A class of high dimensional copulas based on products of bivariate copulas. <http://hal.archives-ouvertes.fr/hal-00910775>. R. B. Nelsen. An Introduction to Copulas. Springer, 1999.

### See Also

[pbc](#)

### Examples

```
## 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**

```
## 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 = pbOptim(init, pbDataNormal, pbNormal, method="BFGS")
```

# Index

## \*Topic **models**

- pbcModels, 7
- compute (PBC-class), 5
- dPBC (PBC), 4
- draw (PBC-class), 5
- dxdyphi.norm (PBC-class), 5
- dxdyphi.student (PBC-class), 5
- dxnorm (PBC-class), 5
- dxnorm.student (PBC-class), 5
- getBINMAT (PBC-class), 5
- getBinMat (PBC-class), 5
- getBINMAT, PBC-method (PBC-class), 5
- getDxdyf (PBC-class), 5
- getDxdyf, PBC-method (PBC-class), 5
- getDxf (PBC-class), 5
- getDxf, PBC-method (PBC-class), 5
- getF (PBC-class), 5
- getF, PBC-method (PBC-class), 5
- getGradxdyf (PBC-class), 5
- getGradxdyf, PBC-method (PBC-class), 5
- getGradxf (PBC-class), 5
- getGradxf, PBC-method (PBC-class), 5
- getGraf (PBC-class), 5
- getGraf, PBC-method (PBC-class), 5
- getGraph (PBC-class), 5
- getGraph, PBC-method (PBC-class), 5
- getModel (PBC-class), 5
- getModel, PBC-method (PBC-class), 5
- getNIteration (PBC-class), 5
- getNIteration, PBC-method (PBC-class), 5
- getRoot (PBC-class), 5
- getRoot, PBC-method (PBC-class), 5
- gradxdyphi.norm (PBC-class), 5
- gradxdyphi2.student (PBC-class), 5
- gradxf.norm (PBC-class), 5
- gradxf2.student (PBC-class), 5
- igraph, 6, 7
- igraph-class (PBC-class), 5
- margin (PBC-class), 5
- mpAlgo, 2, 3, 4, 5
- optim, 9
- PBC, 2, 4, 4, 8
- pbc, 8
- pbc (PBC-class), 5
- PBC-class, 5
- PBC-package, 2
- pbcAMH (pbcModels), 7
- pbcFGM (pbcModels), 7
- pbcFrank (pbcModels), 7
- pbcGumbel (pbcModels), 7
- pbcJoe (pbcModels), 7
- pbcModels, 6, 7
- pbcNormal (pbcModels), 7
- pbcOptim, 2, 8
- pbcPlot (PBC-class), 5
- pbcPlot, PBC-method (PBC-class), 5
- phi.norm (PBC-class), 5
- phi.student (PBC-class), 5
- phi.student1 (PBC-class), 5
- pPBC (PBC), 4
- rPBC (PBC), 4
- setDensity (PBC-class), 5
- setDensity, PBC-method (PBC-class), 5
- setGradient (PBC-class), 5
- setGradient, PBC-method (PBC-class), 5