Package: PUcopula (via r-universe)

May 19, 2026

Type Package

Title Structures and functions for partition of unity copulas

Version 0.1.1

Author Andreas Maendle <maendle@uni-bremen.de>

Maintainer Andreas Maendle <maendle@uni-bremen.de>

Description Structures for creating some types of partition of unity copula. Based on joint work with Dietmar Pfeifer and Olena Ragulina.

License GPL (>= 2)

Encoding UTF-8

LazyData true

Depends R (>= 2.10)

RoxygenNote 7.3.3

Suggests copula

Imports matrixStats, methods, msm, pracma

Repository https://amaendle.r-universe.dev

Date/Publication 2026-03-19 21:58:35 UTC

RemoteUrl https://github.com/amaendle/PUcopula

RemoteRef HEAD

RemoteSha 74dbcfa66cbb7923216790c80c6808029b13fe39

Contents

natperils *losses from natural perils data set*

Description

This is the 19-dimensional data set presented in Neumann et al. (2019), Tab. 2 and Tab. 3.

Details

The data contains insurance losses from a non-life portfolio of natural perils in 19 areas in central Europe over a time period of 20 years. The monetary unit is 1 million €.

References

Neumann, A., Bodnar, T., Pfeifer, D., & Dickhaus, T. (2019). Multivariate multiple test procedures based on nonparametric copula estimation. Biometrical Journal, 61(1), 40-61.

PUCopula *Create a PUCopula object*

Description

Constructs an object of class `PUCopula`. This is the main user-facing function for creating partition of unity copulas.

Usage

```
PUCopula(
  dim = 0,
  family = c("binom", "nbinom", "poisson", "sample", "gamma", "beta", "power"),
  pars.a = c(10, 10),
 patch = c("rook", "uFrechet", "lFrechet", "varwc", "Bernstein", "Gauss", "sample"),
  patchpar = list(NULL),
  data,
  numericCDF = FALSE
)
```

Arguments

`dim`	Integer. Dimension of the copula. If 0, it is inferred from `data`.
`family`	Character vector specifying the copula family. Possible values include `"binom"`, `"nbinom"`, `"poisson"`, `"sample"`, `"gamma"`, `"beta"`, `"power"`.
`pars.a`	Numeric vector of parameters controlling the family.
`patch`	Character. Patchwork type, e.g. `"rook"`, `"uFrechet"`, `"lFrechet"`, `"Bernstein"`, `"Gauss"`, `"sample"`, `"varwc"`.

patchpar	List. Parameters for the patchwork (e.g., correlation for Gaussian copula).
data	Matrix. Input data used to construct the copula.
numericCDF	Logical. Whether to use numerical CDF approximations.

Value

An object of class PUCopula.

Examples

```
data(stormflood)

x <- PUCopula(
  family = "gamma",
  pars.a = c(40, 43),
  patch = "lFrechet",
  data = stormflood
)

plot(x@ranks)
```

PUCopula-class *S4 class representing a partition of unity copula*

Description

The PUCopula class implements partition-of-unity copulas as developed in Pfeifer et al. (2016, 2017, 2019). It provides a framework for constructing copulas using general partitions of unity, including both discrete and continuous cases.

Value

An object of class PUCopula.

Slots

dim Numeric scalar. Dimension of the copula.

family Character vector. Specifies the family of the PU copula. (and therefore the densities fdenss)

par.factor Numeric vector. Scaling parameter for ranks.

pars.a Numeric vector. Parameters controlling the distribution of φ....

patchpar List. Parameters for the copula driver, e.g. rho in case of Gauss copula driver, K,m in case of Bernstein,...

p Matrix. Internal parameter representation.

phi Function. Base density $\varphi(u, s)$.... (obsolete due to phis)

psy Function. Secondary density function.... (obsolete due to phis)

phis List of functions. Collection of component densities.... $\varphi_k(s,u)$ for $k = 1, dots, d \in N$. Either continuous case: A list of functions which represent Lebesgue densities of distributions over R with a parameter u in (0,1), i.e.

$$\varphi_k(s,u) \geq 0 \text{ and } \int_{-\infty}^{\infty} \varphi_k(s,u)\, ds = 1 \text{ for } u \in (0,1),$$

Or (discrete case): A list of functions which represent discrete probabilities over Z+ with a parameter u in (0,1).

continuous Logical vector. Indicates whether components are continuous....

alph Function. Integral of φ....

beta Function. Integral of ψ....

alphs List of functions. Marginal densities.... Integral over the elements of phis w.r.t. u, i.e.

$$\alpha_k(s) := \int_0^1 \varphi_k(s,u)\, du \in (0,\infty)$$

alphsCDF List of functions. CDFs corresponding to densities alphs, i.e.

$$A_k(s) := \int_{-\infty}^{s} \alpha_k(w)\, dw \text{ for } s \in R$$

opstart List. Starting values for optimization....

cdflim List. Limits for numerically stable CDF evaluation....

cdfdom List. Effective domain of the CDFs....

alphsquantf List. Fast quantile approximations....

cdfappx List. Approximate CDF functions....

alphsobjective List. Objective functions for quantiles....

alphsquant List. Quantile functions....

fdens Function. Density function.... (obsolete due to fdenss)

gdens Function. Secondary density.... (obsolete due to fdenss)

fdenss List of functions. Contains densities obtained from normalizing the functions $\varphi(s,u)$ (from slot phis) w.r.t. $u \in 0, 1$, i.e.

$$f_k(s,u) := \frac{\varphi_k(s,u)}{\alpha_k(s)},\ u \in (0,1) \text{ for } s \in R$$

cpatch Function. Copula driver density....

PUdens Function. Density of the (continuous) PU copula defined by

$$c(\mathbf{u}) := \int_{-\infty}^{\infty} \cdots \int_{-\infty}^{\infty} p(s_1, \cdots, s_d) \prod_{k=1}^{d} f_k(s_k, u_k)\, ds_1 \cdots ds_d, \mathbf{u} = (u_1, \cdots, u_d) \in (0,1)^d$$

where $p(s_1, \cdots, s_d)$ denotes the density of an arbitrary d-dimensional random vector $\mathbf{S} = S_1, \cdots, S_d)$ over R^d with marginal densities $\alpha(cdot)$ for S_k.

`fq` Function. Auxiliary function.

`gq` Function. Auxiliary function.

`patch` Character. Patchwork type.

`dPWork` Function. Patchwork density....

`data` Matrix. Input data.

`ranks` Matrix. Rank-transformed data.

`relRanks` Matrix. Relative ranks.

`rpatch` Function. Patchwork sampler - generates random samples from the specified patchwork.

`rand` Function. Random number generator to simulate from PU copula.

References

Pfeifer, D., Tsatedem, H. A., Mändle, A., and Girschig, C. (2016). New copulas based on general partitions-of-unity and their applications to risk management. *Dependence Modeling*, 4(1). doi:10.1515/demo20160006

Pfeifer, D., Mändle, A., and Ragulina, O. (2017). New copulas based on general partitions-of-unity and their applications to risk management (part II). *Dependence Modeling*, 5(1), 246–255. doi:10.1515/demo20170014

Pfeifer, D., Mändle, A., Ragulina, O., and Girschig, C. (2019). New copulas based on general partitions-of-unity (part III) — the continuous case. *Dependence Modeling*, 7(1), 181–201. doi:10.1515/demo20190009

Examples

```
# Use dataset stormflood
data(stormflood)

# example: choose Gamma copula model
x <- PUCopula(family="gamma", pars.a=c(40, 43), patch="lFrechet",data=stormflood)

# The following plots show the common ranks and several copula drivers;
#   these plots do not depend on the above chosen family of the PUcopula.
# 2*3 plots in one
par(mfrow=c(2,3))
# plot the empirical rank vectors
plot(x@ranks, sub="emp. rank vectors", xlab="", ylab="")
# plot the lower Fréchet driver, i.e. phi=-1
plot(x@rpatch(2000,"lFrechet"), sub="lower Fréchet driver, phi=-1", xlab="", ylab="")
# plot for phi=-0.8
plot(
  x@rpatch(2000,"Bernstein",list(m=20,K=20)),
  sub="Bernstein copula, m=20, K=20",
  xlab="", ylab=""
)
# plot for phi=0 (rook copula)
plot(x@rpatch(2000,"rook"), sub="rook copula, phi=0", xlab="", ylab="")
# plot for phi=0.9
plot(x@rpatch(2000,"Gauss",0.9), sub="phi=0.9", xlab="", ylab="")
```

```
# plot for phi=1 (upper Fréchet driver)
plot(x@rpatch(2000,"uFrechet"), sub="upper Fréchet driver, phi=1", xlab="", ylab="")
# single plot window
par(mfrow=c(1,1))

# plot the densities given by phis (=densities of Gamma distributions)
# plot for s=1,3,5,7,...,999 (s must be in (0,inf))
palette(rainbow(10))
xs=c(seq(0,0.01,length.out=200),seq(0.02,1,length.out=200))
# function phi for a fixed s
foo <- function(u) x@phi(u,1)
ys=foo(xs)
# plot for s=1
plot(
  x=xs,y=ys,
  col=1,
  ylim=c(0,0.4), xlim=c(0,1),
  type = "l"
) #plot(foo, col=1, ylim=c(0,0.94), xlim=c(0,1)) # ylim=c(0,1e-88)
# repeat for  s=3,5,7,...,999
for (i in 2:500) {
  foo <- function(u) x@phi(u,i*2-1)
  ys=foo(xs)
    lines(x=xs,y=ys, col=i+1) #plot(foo, add=TRUE, col=i+1)
}
# for a Gamma PUC the alphs are densities of inverse Pareto distributions...
plot(x@alphs[[1]], xlim=c(0,500))
# the fdenss are basically a normalization of the phis using alphs
# for a fixes s they are densities of exponentially transformed
# Gamma distributions
# heatplot:
us <- seq(0,1,length.out=100)
ss <- seq(1,500,length.out=100)
zs <- matrix(nrow=length(us),ncol=length(ss))
for (s in 1:length(ss)) zs[,s] <- x@fdenss[[1]](us, s)
image(x=us, y=ss, z=zs, col=terrain.colors(15))

#For the simulation from the Gamma copula, the slots alphsCDF
# and Qk/corresponnding quantile are used internally
# plot the CDF corresponding to density alphs[1]
## Not run:
x <- PUCopula(family="gamma", pars.a=c(40, 43), patch="lFrechet",data=stormflood, numericCDF=TRUE)

warnings()
# Warning messages:
#   1: In phis[[i]](u, s) : outside range (0,inf) for s = -752.818778594017
# 2: In phis[[i]](u, s) : outside range (0,inf) for s = -752.818778594017
# 3: In phis[[i]](u, s) : outside range (0,inf) for s = -752.818778594017
# 4: In phis[[i]](u, s) : outside range (0,inf) for s = -752.818778594017
# 5: In phis[[i]](u, s) : outside range (0,inf) for s = -752.818778594017
# 6: In phis[[i]](u, s) : outside range (0,inf) for s = -752.818778594017
# 7: In phis[[i]](u, s) : outside range (0,inf) for s = -752.818778594017
# 8: In phis[[i]](u, s) : outside range (0,inf) for s = -752.818778594017
```

```
# 9: In phis[[i]](u, s) : outside range (0,inf) for s = -752.818778594017
# 10: In phis[[i]](u, s) : outside range (0,inf) for s = -752.818778594017
# 11: In phis[[i]](u, s) : outside range (0,inf) for s = -752.818778594017
# 12: In phis[[i]](u, s) : outside range (0,inf) for s = -752.818778594017
# 13: In phis[[i]](u, s) : outside range (0,inf) for s = -752.818778594017
# 14: In phis[[i]](u, s) : outside range (0,inf) for s = -752.818778594017
# 15: In phis[[i]](u, s) : outside range (0,inf) for s = -752.818778594017
# 16: In phis[[i]](u, s) : outside range (0,inf) for s = -752.818778594017
# 17: In phis[[i]](u, s) : outside range (0,inf) for s = -752.818778594017
# 18: In phis[[i]](u, s) : outside range (0,inf) for s = -752.818778594017
# 19: In phis[[i]](u, s) : outside range (0,inf) for s = -752.818778594017
# 20: In phis[[i]](u, s) : outside range (0,inf) for s = -752.818778594017
# 21: In phis[[i]](u, s) : outside range (0,inf) for s = -752.818778594017
# 22: In phis[[i]](u, s) : outside range (0,inf) for s = -752.818778594017
# 23: In phis[[i]](u, s) : outside range (0,inf) for s = -752.818778594017
# 24: In phis[[i]](u, s) : outside range (0,inf) for s = -752.818778594017
# 25: In phis[[i]](u, s) : outside range (0,inf) for s = -752.818778594017
# 26: In phis[[i]](u, s) : outside range (0,inf) for s = -752.818778594017
# 27: In phis[[i]](u, s) : outside range (0,inf) for s = -752.818778594017
# 28: In phis[[i]](u, s) : outside range (0,inf) for s = -752.818778594017
# 29: In phis[[i]](u, s) : outside range (0,inf) for s = -752.818778594017
# 30: In phis[[i]](u, s) : outside range (0,inf) for s = -752.818778594017
# 31: In phis[[i]](u, s) : outside range (0,inf) for s = -752.818778594017
# 32: In phis[[i]](u, s) : outside range (0,inf) for s = -752.818778594017
# 33: In phis[[i]](u, s) : outside range (0,inf) for s = -752.818778594017
# 34: In phis[[i]](u, s) : outside range (0,inf) for s = -752.818778594017
# 35: In phis[[i]](u, s) : outside range (0,inf) for s = -752.818778594017
# 36: In phis[[i]](u, s) : outside range (0,inf) for s = -752.818778594017
# 37: In phis[[i]](u, s) : outside range (0,inf) for s = -752.818778594017
# 38: In phis[[i]](u, s) : outside range (0,inf) for s = -752.818778594017
# 39: In phis[[i]](u, s) : outside range (0,inf) for s = -91.7078148698631
# 40: In phis[[i]](u, s) : outside range (0,inf) for s = -91.7078148698631
# 41: In phis[[i]](u, s) : outside range (0,inf) for s = -91.7078148698631
# 42: In phis[[i]](u, s) : outside range (0,inf) for s = -91.7078148698631
# 43: In phis[[i]](u, s) : outside range (0,inf) for s = -91.7078148698631
# 44: In phis[[i]](u, s) : outside range (0,inf) for s = -91.7078148698631
# 45: In phis[[i]](u, s) : outside range (0,inf) for s = -91.7078148698631
# 46: In phis[[i]](u, s) : outside range (0,inf) for s = -91.7078148698631
# 47: In phis[[i]](u, s) : outside range (0,inf) for s = -91.7078148698631
# 48: In phis[[i]](u, s) : outside range (0,inf) for s = -91.7078148698631
# 49: In phis[[i]](u, s) : outside range (0,inf) for s = -91.7078148698631
# 50: In phis[[i]](u, s) : outside range (0,inf) for s = -91.7078148698631

plot(x@alphsCDF[[1]],xlim=c(0,100))
x@alphsobjective[[1]](5,0.8)
x@alphsquant[[1]](0.8)
x@rand(1)

## End(Not run)

# Create a beta copula
x <- PUCopula(family="beta", pars.a=c(40, 43), patch="lFrechet",data=stormflood)
x@phi(0.5,0.5)
```

```
#[1] 2.210851e-11
x@alph(0.6)
#[1] 1.458522e-11
#x@fdens(0.5,0.6)
#[1] 0.0002381545
#x@gdens(0.5,0.6)
#[1] 0.0001616076
x@fdenss[[1]](0.5,0.6)
#[1] 0.0002381545
x@fdenss[[2]](0.5,0.6)
#[1] 0.0001616076
x@rand(5)
#does not work
# example 2: gamma copula
x <- PUCopula(family="gamma", pars.a=c(40, 43), patch="lFrechet",data=stormflood)
#plot alpha(s)
plot(x@alphs[[1]])
#the numerically computed alpha is quite close:
s<-10
x@pars.a[1]*s^(x@pars.a[1]-1)/((1+s)^(x@pars.a[1]+1))
#[1] 0.008034519
x@alph(s)
#[1] 0.008034519
# but of course shows some inaccuracies:
s<-2.5
x@alph(s)
#[1] 6.530307e-06
x@pars.a[1]*s^(x@pars.a[1]-1)/((1+s)^(x@pars.a[1]+1))
#[1] 6.530261e-06
```

`stormflood` *storm and flood losses data set*

Description

This is the data set used in Cottin, Pfeifer (2014). The table contains some original data from an insurance portfolio of storm and flooding losses, observed over a period of 20 years.

References

Cottin, Claudia, and Dietmar Pfeifer. "From Bernstein polynomials to Bernstein copulas." J. Appl. Funct. Anal 9.3-4 (2014): 277-288.

Index