Package: ZeroOneDists (via r-universe)

October 7, 2026

Title One Zero Statistical Distributions

Version 1.0.1

Description Implementation of new statistical distributions in (0, 1) interval. Each distribution includes the traditional functions as well as an additional function called the family function, which can be used to estimate parameters using Generalized Additive Models for Location, Scale and Shape, GAMLSS by Rigby & Stasinopoulos (2005) <doi:10.1111/j.1467-9876.2005.00510.x>.

License MIT + file LICENSE

Encoding UTF-8

Roxygen list(markdown = TRUE)

URL https://github.com/fhernanb/ZeroOneDists

BugReports https://github.com/fhernanb/ZeroOneDists/issues

Imports gamlss

Config/roxygen2/version 8.1.0

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

Date/Publication 2026-09-07 13:25:56 UTC

RemoteUrl https://github.com/fhernanb/zeroonedists

RemoteRef HEAD

RemoteSha 1369ad4d030d953414c72e1255f943bd5556bc39

Contents

BER 2
BER2 3
checklink 5
dBER 6
dBER2 8
dUHLG 11
dUMB 13

dUPHN 16
make.link.gamlss 18
UHLG 20
UMB 21
UPHN 26

Index **28**

BER *Beta Rectangular distribution*

Description

The Beta Rectangular family

Usage

```
BER(mu.link = "logit", sigma.link = "log", nu.link = "logit")
```

Arguments

`mu.link`	defines the mu.link, with "logit" link as the default for the mu parameter.
`sigma.link`	defines the sigma.link, with "log" link as the default for the sigma parameter.
`nu.link`	defines the nu.link, with "logit" link as the default for the nu parameter.

Details

The Beta Rectangular distribution with parameters `mu`, `sigma` and `nu` has density given by

$f(x|\mu,\sigma,\nu) = \nu + (1-\nu)b(x|\mu,\sigma)$

for $0 < x < 1$, $0 < \mu < 1$, $\sigma > 0$ and $0 < \nu < 1$. The function $b(.)$ corresponds to the traditional beta distribution that can be computed by `dbeta(x, shape1=mu*sigma, shape2=(1-mu)*sigma)`.

Value

Returns a gamlss.family object which can be used to fit a BER distribution in the `gamlss()` function.

Author(s)

Karina Maria Garay, <kgarayo@unal.edu.co>

References

Bayes, C. L., Bazán, J. L., & García, C. (2012). A new robust regression model for proportions. Bayesian Analysis, 7(4), 841-866.

See Also

dBER

Examples

```
# Example 1
# Generating some random values with
# known mu and sigma
y <- rBER(n=500, mu=0.5, sigma=10, nu=0.5)

# Fitting the model
library(gamlss)
mod1 <- gamlss(y~1, family=BER)

# Extracting the fitted values for mu, sigma and nu
# using the inverse link function
inv_logit <- function(x) 1/(1 + exp(-x))

inv_logit(coef(mod1, what="mu"))
exp(coef(mod1, what="sigma"))
inv_logit(coef(mod1, what="nu"))

# Example 2
# Generating random values under some model

# A function to simulate a data set with Y ~ BER
gendat <- function(n) {
  x1 <- runif(n, min=0.4, max=0.6)
  x2 <- runif(n, min=0.4, max=0.6)
  x3 <- runif(n, min=0.4, max=0.6)
  mu    <- inv_logit(-0.5 + 1*x1)
  sigma <- exp(-1 + 4.8*x2)
  nu    <- inv_logit(-1 + 0.5*x3)
  y <- rBER(n=n, mu=mu, sigma=sigma, nu=nu)
  data.frame(y=y, x1=x1, x2=x2, x3=x3)
}

set.seed(1234)
datos <- gendat(n=500)

mod2 <- gamlss(y~x1, sigma.fo=~x2, nu.fo=~x3,
               family=BER, data=datos,
               control=gamlss.control(n.cyc=500, trace=TRUE))

summary(mod2)
```

BER2 *Beta Rectangular distribution version 2*

Description

The Beta Rectangular family

Usage

```
BER2(mu.link = "logit", sigma.link = "log", nu.link = "logit")
```

Arguments

mu.link	defines the mu.link, with "logit" link as the default for the mu parameter.
sigma.link	defines the sigma.link, with "log" link as the default for the sigma parameter.
nu.link	defines the nu.link, with "logit" link as the default for the nu parameter.

Details

The Beta Rectangular distribution with parameters mu, sigma and nu has density given by

$f(x|\mu,\sigma,\nu) = \nu + (1-\nu)b(x|\mu,\sigma)$

for $0 < x < 1$, $0 < \mu < 1$, $\sigma > 0$ and $0 < \nu < 1$. The function $b(.)$ corresponds to the traditional beta distribution that can be computed by `dbeta(x, shape1=mu*sigma, shape2=(1-mu)*sigma)`.

Value

Returns a gamlss.family object which can be used to fit a BER2 distribution in the `gamlss()` function.

References

Bayes, C. L., Bazán, J. L., & García, C. (2012). A new robust regression model for proportions. Bayesian Analysis, 7(4), 841-866.

See Also

dBER2

Examples

```
# Example 1
# Generating some random values with
# known mu and sigma
y <- rBER2(n=500, mu=0.3, sigma=7, nu=0.4)

# Fitting the model
library(gamlss)
mod1 <- gamlss(y~1, family=BER2,
               control=gamlss.control(n.cyc=500, trace=FALSE))

# Extracting the fitted values for mu, sigma and nu
# using the inverse link function
inv_logit <- function(x) 1/(1 + exp(-x))

inv_logit(coef(mod1, what="mu"))
exp(coef(mod1, what="sigma"))
inv_logit(coef(mod1, what="nu"))
```

```
# Example 2
# Generating random values under some model

# A function to simulate a data set with Y ~ BER2
gendat <- function(n) {
  x1 <- runif(n, min=0.4, max=0.6)
  x2 <- runif(n, min=0.4, max=0.6)
  x3 <- runif(n, min=0.4, max=0.6)
  mu    <- inv_logit(-0.5 + 1*x1)
  sigma <- exp(-1 + 4.8*x2)
  nu    <- inv_logit(-1 + 0.5*x3)
  y <- rBER2(n=n, mu=mu, sigma=sigma, nu=nu)
  data.frame(y=y, x1=x1, x2=x2, x3=x3)
}

set.seed(1234)
datos <- gendat(n=500)

mod2 <- gamlss(y~x1, sigma.fo=~x2, nu.fo=~x3,
               family=BER2, data=datos,
               control=gamlss.control(n.cyc=500, trace=TRUE))

summary(mod2)
```

---

checklink *Set the Right Link Function for Specified Parameter and Distribution*

Description

This function is used within the distribution family specification of a GAMLSS model to define the right link for each of the parameters of the distribution. This function should not be called by the user unless he/she specify a new distribution family or wishes to change existing link functions in the parameters.

Usage

```
checklink(which.link = NULL, which.dist = NULL, link = NULL, link.List = NULL)
```

Arguments

`which.link`	which parameter link e.g. `which.link="mu.link"`
`which.dist`	which distribution family e.g. `which.dist="Cole.Green"`
`link`	a repetition of `which.link` e.g. `link=substitute(mu.link)`
`link.List`	what link function are required e.g. `link.List=c("inverse", "log", "identity")`

Details

This function is the same `checklink` fuction from gamlss.dist package.

Value

Defines the right link for each parameter.

`dBER` *Beta Rectangular distribution*

Description

These functions define the density, distribution function, quantile function and random generation for the Beta Rectangular distribution with parameters μ, σ and ν.

Usage

```
dBER(x, mu, sigma, nu, log = FALSE)

pBER(q, mu, sigma, nu, lower.tail = TRUE, log.p = FALSE)

qBER(p, mu, sigma, nu, lower.tail = TRUE, log.p = FALSE)

rBER(n, mu, sigma, nu)
```

Arguments

`x`, `q`	vector of (non-negative integer) quantiles.
`mu`	vector of the mu parameter.
`sigma`	vector of the sigma parameter.
`nu`	vector of the nu parameter.
`log`, `log.p`	logical; if TRUE, probabilities p are given as log(p).
`lower.tail`	logical; if TRUE (default), probabilities are $P[X <= x]$, otherwise, $P[X > x]$.
`p`	vector of probabilities.
`n`	number of random values to return.

Details

The Beta Rectangular distribution with parameters μ, σ and ν has a support in $(0, 1)$ and density given by

$f(x|\mu, \sigma, \nu) = \nu + (1 - \nu)b(x|\mu, \sigma)$

for $0 < x < 1$, $0 < \mu < 1$, $\sigma > 0$ and $0 < \nu < 1$. The function $b(.)$ corresponds to the traditional beta distribution that can be computed by `dbeta(x, shape1=mu*sigma, shape2=(1-mu)*sigma)`.

Value

dBER gives the density, pBER gives the distribution function, qBER gives the quantile function, rBER generates random deviates.

Author(s)

Karina Maria Garay, <kgarayo@unal.edu.co>

References

Bayes, C. L., Bazán, J. L., & García, C. (2012). A new robust regression model for proportions. Bayesian Analysis, 7(4), 841-866.

See Also

BER.

Examples

```
# Example 1
# Plotting the density function for different parameter values
curve(dBER(x, mu=0.5, sigma=10, nu=0),
      from=0, to=1, col="green", las=1, ylab="f(x)")

curve(dBER(x, mu=0.5, sigma=10, nu=0.2),
      add=TRUE, col= "blue1")

curve(dBER(x, mu=0.5, sigma=10, nu=0.4),
      add=TRUE, col="yellow")

curve(dBER(x, mu=0.5, sigma=10, nu=0.6),
      add=TRUE, col="red")

legend("topleft", col=c("green", "blue1", "yellow", "red"),
       lty=1, bty="n",
       legend=c("mu=0.5, sigma=10, nu=0",
                "mu=0.5, sigma=10, nu=0.2",
                "mu=0.5, sigma=10, nu=0.4",
                "mu=0.5, sigma=10, nu=0.6"))


curve(dBER(x, mu=0.3, sigma=10, nu=0),
       from=0, to=1, col="green", las=1, ylab="f(x)")

curve(dBER(x, mu=0.3, sigma=10, nu=0.2),
       add=TRUE, col= "blue1")

curve(dBER(x, mu=0.3, sigma=10, nu=0.4),
       add=TRUE, col="yellow")

curve(dBER(x, mu=0.3, sigma=10, nu=0.6),
       add=TRUE, col="red")
```

```
legend("topright", col=c("green", "blue1", "yellow", "red"),
       lty=1, bty="n",
       legend=c("mu=0.5, sigma=10, nu=0",
                "mu=0.5, sigma=10, nu=0.2",
                "mu=0.5, sigma=10, nu=0.4",
                "mu=0.5, sigma=10, nu=0.6"))


# Example 2
# Checking if the cumulative curves converge to 1
curve(pBER(x, mu=0.5, sigma=10, nu=0),
       from=0, to=1, col="green", las=1, ylab="f(x)")

curve(pBER(x, mu=0.5, sigma=10, nu=0.2),
       add=TRUE, col= "blue1")

curve(pBER(x, mu=0.5, sigma=10, nu=0.4),
       add=TRUE, col="yellow")

curve(pBER(x, mu=0.5, sigma=10, nu=0.6),
       add=TRUE, col="red")

legend("topleft", col=c("green", "blue1", "yellow", "red"),
       lty=1, bty="n",
       legend=c("mu=0.5, sigma=10, nu=0",
                "mu=0.5, sigma=10, nu=0.2",
                "mu=0.5, sigma=10, nu=0.4",
                "mu=0.5, sigma=10, nu=0.6"))

# Example 3
# Checking the quantile function
mu <- 0.5
sigma <- 10
nu <- 0.4
p <- seq(from=0.01, to=0.99, length.out=100)
plot(x=qBER(p, mu=mu, sigma=sigma, nu=nu), y=p,
     xlab="Quantile", las=1, ylab="Probability")
curve(pBER(x, mu=mu, sigma=sigma, nu=nu), add=TRUE, col="red")

# Example 4
# Comparing the random generator output with
# the theoretical density
x <- rBER(n= 10000, mu=0.5, sigma=10, nu=0.1)
hist(x, freq=FALSE)
curve(dBER(x, mu=0.5, sigma=10, nu=0.1),
      col="tomato", add=TRUE)
```

---

dBER2 *Beta Rectangular distribution version 2*

Description

These functions define the density, distribution function, quantile function and random generation for the Beta Rectangular distribution with parameters μ, σ and ν reparameterized to ensure $E(X) = \mu$.

Usage

```
dBER2(x, mu, sigma, nu, log = FALSE)

pBER2(q, mu, sigma, nu, lower.tail = TRUE, log.p = FALSE)

qBER2(p, mu, sigma, nu, lower.tail = TRUE, log.p = FALSE)

rBER2(n, mu, sigma, nu)
```

Arguments

x, q	vector of (non-negative integer) quantiles.
mu	vector of the mu parameter.
sigma	vector of the sigma parameter.
nu	vector of the nu parameter.
log, log.p	logical; if TRUE, probabilities p are given as log(p).
lower.tail	logical; if TRUE (default), probabilities are $P[X <= x]$, otherwise, $P[X > x]$.
p	vector of probabilities.
n	number of random values to return.

Details

The Beta Rectangular distribution with parameters μ, σ and ν has a support in $(0, 1)$ and density given by

$f(x|\mu, \sigma, \nu) = \nu + (1 - \nu)b(x|\mu, \sigma)$

for $0 < x < 1$, $0 < \mu < 1$, $\sigma > 0$ and $0 < \nu < 1$. The function $b(.)$ corresponds to the traditional beta distribution that can be computed by `dbeta(x, shape1=mu*sigma, shape2=(1-mu)*sigma)`.

Value

`dBER2` gives the density, `pBER2` gives the distribution function, `qBER2` gives the quantile function, `rBER2` generates random deviates.

References

Bayes, C. L., Bazán, J. L., & García, C. (2012). A new robust regression model for proportions. Bayesian Analysis, 7(4), 841-866.

See Also

BER2.

Examples

```
# Example 1
# Plotting the density function for different parameter values
curve(dBER2(x, mu=0.5, sigma=10, nu=0),
      from=0, to=1, col="green", las=1, ylab="f(x)")

curve(dBER2(x, mu=0.5, sigma=10, nu=0.2),
      add=TRUE, col= "blue1")

curve(dBER2(x, mu=0.5, sigma=10, nu=0.4),
      add=TRUE, col="yellow")

curve(dBER2(x, mu=0.5, sigma=10, nu=0.6),
      add=TRUE, col="red")

legend("topleft", col=c("green", "blue1", "yellow", "red"),
       lty=1, bty="n",
       legend=c("mu=0.5, sigma=10, nu=0",
                "mu=0.5, sigma=10, nu=0.2",
                "mu=0.5, sigma=10, nu=0.4",
                "mu=0.5, sigma=10, nu=0.6"))


curve(dBER2(x, mu=0.3, sigma=10, nu=0),
       from=0, to=1, col="green", las=1, ylab="f(x)")

curve(dBER2(x, mu=0.3, sigma=10, nu=0.2),
       add=TRUE, col= "blue1")

curve(dBER2(x, mu=0.3, sigma=10, nu=0.4),
       add=TRUE, col="yellow")

curve(dBER2(x, mu=0.3, sigma=10, nu=0.6),
       add=TRUE, col="red")

legend("topright", col=c("green", "blue1", "yellow", "red"),
       lty=1, bty="n",
       legend=c("mu=0.3, sigma=10, nu=0",
                "mu=0.3, sigma=10, nu=0.2",
                "mu=0.3, sigma=10, nu=0.4",
                "mu=0.3, sigma=10, nu=0.6"))


# Example 2
# Checking if the cumulative curves converge to 1
curve(pBER2(x, mu=0.5, sigma=10, nu=0),
       from=0, to=1, col="green", las=1, ylab="f(x)")

curve(pBER2(x, mu=0.5, sigma=10, nu=0.2),
       add=TRUE, col= "blue1")

curve(pBER2(x, mu=0.5, sigma=10, nu=0.4),
```

```
       add=TRUE, col="yellow")

curve(pBER2(x, mu=0.5, sigma=10, nu=0.6),
       add=TRUE, col="red")

legend("topleft", col=c("green", "blue1", "yellow", "red"),
       lty=1, bty="n",
       legend=c("mu=0.5, sigma=10, nu=0",
                "mu=0.5, sigma=10, nu=0.2",
                "mu=0.5, sigma=10, nu=0.4",
                "mu=0.5, sigma=10, nu=0.6"))

# Example 3
# Checking the quantile function
mu <- 0.5
sigma <- 10
nu <- 0.4
p <- seq(from=0.01, to=0.99, length.out=100)
plot(x=qBER2(p, mu=mu, sigma=sigma, nu=nu), y=p,
     xlab="Quantile", las=1, ylab="Probability")
curve(pBER2(x, mu=mu, sigma=sigma, nu=nu), add=TRUE, col="red")

# Example 4
# Comparing the random generator output with
# the theoretical density
x <- rBER2(n= 10000, mu=0.3, sigma=10, nu=0.1)
hist(x, freq=FALSE)
curve(dBER2(x, mu=0.3, sigma=10, nu=0.1),
      col="tomato", add=TRUE)
```

---

dUHLG *Unit Half Logistic-Geometry distribution*

Description

These functions define the density, distribution function, quantile function and random generation for the Unit Half Logistic-Geometry distribution with parameter μ.

Usage

```
dUHLG(x, mu, log = FALSE)

pUHLG(q, mu, lower.tail = TRUE, log.p = FALSE)

qUHLG(p, mu, lower.tail = TRUE, log.p = FALSE)

rUHLG(n, mu)
```

Arguments

x, q	vector of (non-negative integer) quantiles.
mu	vector of the mu parameter.
log, log.p	logical; if TRUE, probabilities p are given as log(p).
lower.tail	logical; if TRUE (default), probabilities are $P[X <= x]$, otherwise, $P[X > x]$.
p	vector of probabilities.
n	number of random values to return.

Details

The Unit Half Logistic-Geometry distribution with parameter μ has a support in $(0, 1)$ and density given by

$f(x|\mu) = \frac{2\mu}{(\mu+(2-\mu)x)^2}$

for $0 < x < 1$ and $\mu > 0$.

Value

dUHLG gives the density, pUHLG gives the distribution function, qUHLG gives the quantile function, rUHLG generates random deviates.

Author(s)

Juan Diego Suarez Hernandez, <jsuarezhe@unal.edu.co>

References

Ramadan, A. T., Tolba, A. H., & El-Desouky, B. S. (2022). A unit half-logistic geometric distribution and its application in insurance. Axioms, 11(12), 676.

See Also

UHLG.

Examples

```
# Example 1
# Plotting the density function for different parameter values
curve(dUHLG(x, mu=0.4), from=0.01, to=0.99,
      ylim=c(0, 5), lwd=2,
      col="black", las=1, ylab="f(x)")

curve(dUHLG(x, mu=1), lwd=2,
      add=TRUE, col="red")

curve(dUHLG(x, mu=2), lwd=2,
      add=TRUE, col="green")

curve(dUHLG(x, mu=7), lwd=2,
```

```
      add=TRUE, col="blue")

legend("topright",
       col=c("black", "red", "green", "blue"),
       lty=1, bty="n", lwd=2,
       legend=c("mu=0.4",
                "mu=1",
                "mu=2",
                "mu=7"))

# Example 2
# Checking if the cumulative curves converge to 1
curve(pUHLG(x, mu=0.25), lwd=2,
      from=0.001, to=0.999, col="black", las=1, ylab="F(x)")

curve(pUHLG(x, mu=0.7), lwd=2,
      add=TRUE, col="red")

curve(pUHLG(x, mu=1.8), lwd=2,
      add=TRUE, col="green")

curve(pUHLG(x, mu=2.2), lwd=2,
      add=TRUE, col="blue")

legend("bottomright", col=c("black", "red", "green", "blue"),
       lty=1, bty="n", lwd=2,
       legend=c("mu=0.25",
                "mu=0.7",
                "mu=1.8",
                "mu=2.2"))

# Example 3
# Checking the quantile function
mu <- 2
p <- seq(from=0.01, to=0.99, length.out=100)
plot(x=qUHLG(p, mu=mu), y=p,
     xlab="Quantile", las=1, ylab="Probability")
curve(pUHLG(x, mu=mu), add=TRUE, col="red")

# Example 4
# Comparing the random generator output with
# the theoretical density
x <- rUHLG(n=10000, mu=0.5)
hist(x, freq=FALSE)
curve(dUHLG(x, mu=0.5), lwd=2,
      col="tomato", add=TRUE, from=0.01, to=0.99)
```

---

dUMB *Unit Maxwell-Boltzmann distribution*

Description

These functions define the density, distribution function, quantile function and random generation for the Unit Maxwell-Boltzmann distribution with parameter μ.

Usage

```
dUMB(x, mu = 1, log = FALSE)

pUMB(q, mu = 1, lower.tail = TRUE, log.p = FALSE)

qUMB(p, mu, lower.tail = TRUE, log.p = FALSE)

rUMB(n = 1, mu = 1)
```

Arguments

x, q	vector of (non-negative integer) quantiles.
mu	vector of the mu parameter.
log, log.p	logical; if TRUE, probabilities p are given as log(p).
lower.tail	logical; if TRUE (default), probabilities are $P[X <= x]$, otherwise, $P[X > x]$.
p	vector of probabilities.
n	number of random values to return.

Details

The Unit Maxwell-Boltzmann distribution with parameter μ has a support in $(0, 1)$ and density given by

$f(x|\mu) = \frac{\sqrt{(2/\pi)}\log^2(1/x)\exp(-\frac{\log^2(1/x)}{2\mu^2})}{\mu^3 x}$

for $0 < x < 1$ and $\mu > 0$.

Value

dUMB gives the density, pUMB gives the distribution function, qUMB gives the quantile function, rUMB generates random deviates.

Author(s)

David Villegas Ceballos, <david.villegas1@udea.edu.co>

References

Biçer, C., Bakouch, H. S., Biçer, H. D., Alomair, G., Hussain, T., y Almohisen, A. (2024). Unit Maxwell-Boltzmann Distribution and Its Application to Concentrations Pollutant Data. Axioms, 13(4), 226.

See Also

UMB.

Examples

```
# Example 1
# Plotting the density function for different parameter values
curve(dUMB(x, mu=0.4), from=0, to=1,
      ylim=c(0, 12),
      col="green", las=1, ylab="f(x)")

curve(dUMB(x, mu=1),
      add=TRUE, col="blue1")

curve(dUMB(x, mu=2),
      add=TRUE, col="black")

curve(dUMB(x, mu=7),
      add=TRUE, col="red")

legend("topright",
       col=c("green", "blue1", "black", "red"),
       lty=1, bty="n",
       legend=c("mu=0.4",
                "mu=1",
                "mu=2",
                "mu=7"))

# Example 2
# Checking if the cumulative curves converge to 1
curve(pUMB(x, mu=0.25),
      from=0, to=1, col="green", las=1, ylab="F(x)")

curve(pUMB(x, mu=0.9),
      add=TRUE, col="blue1")

curve(pUMB(x, mu=1.8),
      add=TRUE, col="black")

curve(pUMB(x, mu=2.2),
      add=TRUE, col="red")

legend("bottomright", col=c("green", "blue1", "black", "red"),
       lty=1, bty="n",
       legend=c("mu=0.25",
                "mu=0.9",
                "mu=1.8",
                "mu=2.2"))

# Example 3
# Checking the quantile function
mu <- 2
```

```
p <- seq(from=0, to=1, length.out=100)
plot(x=qUMB(p, mu=mu), y=p,
     xlab="Quantile", las=1, ylab="Probability")
curve(pUMB(x, mu=mu), add=TRUE, col="red")

# Example 4
# Comparing the random generator output with
# the theoretical density
x <- rUMB(n=1000, mu=0.5)
hist(x, freq=FALSE)
curve(dUMB(x, mu=0.5),
      col="tomato", add=TRUE, from=0, to=1)
```

---

dUPHN *Unit-Power Half-Normal distribution*

Description

These functions define the density, distribution function, quantile function and random generation for the Unit-Power Half-Normal distribution with parameter μ and σ.

Usage

```
dUPHN(x, mu, sigma, log = FALSE)

pUPHN(q, mu, sigma, lower.tail = TRUE, log.p = FALSE)

qUPHN(p, mu, sigma, lower.tail = TRUE, log.p = FALSE)

rUPHN(n, mu, sigma)
```

Arguments

x, q	vector of (non-negative integer) quantiles.
mu	vector of the mu parameter.
sigma	vector of the sigma parameter.
log, log.p	logical; if TRUE, probabilities p are given as log(p).
lower.tail	logical; if TRUE (default), probabilities are $P[X <= x]$, otherwise, $P[X > x]$.
p	vector of probabilities.
n	number of random values to return.

Details

The Unit-Power Half-Normal distribution with parameters μ and σ has a support in $(0, 1)$ and density given by

$f(x|\mu, \sigma) = \frac{2\mu}{\sigma x^2}\phi(\frac{1-x}{\sigma x})(2\Phi(\frac{1-x}{\sigma x}) - 1)^{\mu-1}$

for $0 < x < 1$, $\mu > 0$ and $\sigma > 0$.

Value

dUPHN gives the density, pUPHN gives the distribution function, qUPHN gives the quantile function, rUPHN generates random deviates.

Author(s)

Juan Diego Suarez Hernandez, <jsuarezhe@unal.edu.co>

References

Santoro, K. I., Gómez, Y. M., Soto, D., & Barranco-Chamorro, I. (2024). Unit-Power Half-Normal Distribution Including Quantile Regression with Applications to Medical Data. Axioms, 13(9), 599.

See Also

dUPHN.

Examples

```
# Example 1
# Plotting the density function for different parameter values
curve(dUPHN(x, mu=1, sigma=1), ylim=c(0, 5),
      from=0.01, to=0.99, col="black", las=1, ylab="f(x)")

curve(dUPHN(x, mu=2, sigma=1),
      add=TRUE, col= "red")

curve(dUPHN(x, mu=3, sigma=1),
      add=TRUE, col="seagreen")

curve(dUPHN(x, mu=4, sigma=1),
      add=TRUE, col="royalblue2")

legend("topright", col=c("black", "red", "seagreen", "royalblue2"),
       lty=1, bty="n",
       legend=c("mu=1, sigma=1",
                "mu=2, sigma=1",
                "mu=3, sigma=1",
                "mu=4, sigma=1"))

# Example 2
# Checking if the cumulative curves converge to 1
curve(pUPHN(x, mu=1, sigma=1),
```

```
       from=0.01, to=0.99, col="black", las=1, ylab="F(x)")

curve(pUPHN(x, mu=2, sigma=1),
       add=TRUE, col= "red")

curve(pUPHN(x, mu=3, sigma=1),
       add=TRUE, col="seagreen")

curve(pUPHN(x, mu=4, sigma=1),
       add=TRUE, col="royalblue2")

legend("topleft", col=c("black", "red", "seagreen", "royalblue2"),
       lty=1, bty="n",
       legend=c("mu=1, sigma=1",
                "mu=2, sigma=1",
                "mu=3, sigma=1",
                "mu=4, sigma=1"))

# Example 3
# Checking the quantile function
mu <- 2
sigma <- 3
p <- seq(from=0.01, to=0.99, length.out=100)
plot(x=qUPHN(p, mu=mu, sigma=sigma), y=p,
     xlab="Quantile", las=1, ylab="Probability")
curve(pUPHN(x, mu=mu, sigma=sigma), add=TRUE, col="red")

# Example 4
# Comparing the random generator output with
# the theoretical density
x <- rUPHN(n= 10000, mu=4, sigma=1)
hist(x, freq=FALSE)
curve(dUPHN(x, mu=4, sigma=1),
      col="tomato", add=TRUE, from=0.01, to=0.99)
```

`make.link.gamlss` *Create a Link for GAMLSS families*

Description

The function `make.link.gamlss()` is used with `gamlss.family` distributions in package **gamlss()**.Given a link, it returns a link function, an inverse link function, the derivative dpar/deta where 'par' is the appropriate distribution parameter and a function for checking the domain. It differs from the usual `make.link` of `glm()` by having extra links as the `logshifto1`, and the `own`. For the use of the `own` link see the example bellow. `show.link` provides a way in which the user can identify the link functions available for each gamlss distribution. If your required link function is not available for any of the gamlss distributions you can add it in.

Usage

```
make.link.gamlss(link)
```

Arguments

link	character or numeric; one of "logit", "probit", "cloglog", "identity", "log", "sqrt", "1/mu^2", "inverse", "logshifted", "logitshifted", or number, say lambda resulting in power link μ^{λ}.

Details

The own link function is added to allow the user greater flexibility. In order to used the own link function for any of the parameters of the distribution the own link should appear in the available links for this parameter. You can check this using the function show.link. If the own do not appear in the list you can create a new function for the distribution in which own is added in the list. For example the first line of the code of the binomial distribution, BI, has change from

"mstats <- checklink("mu.link", "Binomial", substitute(mu.link), c("logit", "probit", "cloglog", "log")), in version 1.0-0 of gamlss, to

"mstats <- checklink("mu.link", "Binomial", substitute(mu.link), c("logit", "probit", "cloglog", "log", "own"))

in version 1.0-1. Given that the parameter has own as an option the user needs also to define the following four new functions in order to used an own link.

i) own.linkfun

ii) own.linkinv

iii) own.mu.eta and

iv) own.valideta.

An example is given below.

Only one parameter of the distribution at a time is allowed to have its own link, (unless the same four own functions above are suitable for more that one parameter of the distribution).

Note that from **gamlss** version 1.9-0 the user can introduce its own link function by define an appropriate function, (see the example below).

Value

For the make.link.gamlss a list with components

linkfun: Link function function(parameter)

linkinv: Inverse link function function(eta)

mu.eta: Derivative function(eta) dparameter/deta

valideta: function(eta) TRUE if all of eta is in the domain of linkinv.

For the show.link a list with components the available links for the distribution parameters

Note

For the links involving parameters as in `logshifted` and `logitshifted` the parameters can be passed in the definition of the distribution by calling the `checklink` function, for example in the definition of the `tau` parameter in BCPE distribution the following call is made: `tstats <- checklink("tau.link","Box Cox Power Exponential", substitute(tau.link),c("logshifted", "log", "identity"), par.link = c(1))`

This function is the same `make.link.gamlss` fuction from gamlss.dist package.

`UHLG` *Unit Half Logistic-Geometry distribution*

Description

The Unit Half Logistic-Geometry family

Usage

```
UHLG(mu.link = "log")
```

Arguments

`mu.link`	defines the mu.link, with "log" link as the default for the mu parameter.

Details

The Unit Half Logistic-Geometry distribution with parameter `mu`, has density given by

$f(x|\mu) = \frac{2\mu}{(\mu+(2-\mu)x)^2}$

for $0 < x < 1$ and $\mu > 0$.

Value

Returns a gamlss.family object which can be used to fit a UHLG distribution in the `gamlss()` function.

Author(s)

Juan Diego Suarez Hernandez, <jsuarezhe@unal.edu.co>

References

Ramadan, A. T., Tolba, A. H., & El-Desouky, B. S. (2022). A unit half-logistic geometric distribution and its application in insurance. Axioms, 11(12), 676.

See Also

dUHLG

Examples

```
# Example 1
# Generating some random values with
# known mu
y <- rUHLG(n=500, mu=7)

# Fitting the model
library(gamlss)
mod1 <- gamlss(y~1, family=UHLG,
               control=gamlss.control(n.cyc=500, trace=FALSE))

# Extracting the fitted values for mu, sigma and nu
# using the inverse link function
exp(coef(mod1, what="mu"))

# Example 2
# Generating random values under some model

# A function to simulate a data set with Y ~ UHLG
gendat <- function(n) {
  x1 <- runif(n, min=0.4, max=0.6)
  x2 <- runif(n, min=0.4, max=0.6)
  mu    <- exp(-0.5 + 3*x1 - 2.5*x2)
  y <- rUHLG(n=n, mu=mu)
  data.frame(y=y, x1=x1, x2=x2)
}

datos <- gendat(n=5000)

mod2 <- gamlss(y~x1+x2,
               family=UHLG, data=datos,
               control=gamlss.control(n.cyc=500, trace=TRUE))
summary(mod2)
```

UMB *Unit Maxwell-Boltzmann family*

Description

The function UMB() defines the Unit Maxwell-Boltzmann distribution, a one parameter distribution, for a gamlss.family object to be used in GAMLSS fitting using the function gamlss().

Usage

```
UMB(mu.link = "log")
```

Arguments

mu.link	defines the mu.link, with "log" link as the default for the mu.

Details

The Unit Maxwell-Boltzmann distribution with parameter μ has a support in $(0, 1)$ and density given by

$f(x|\mu) = \frac{\sqrt{(2/\pi)}\log^2(1/x)\exp(-\frac{\log^2(1/x)}{2\mu^2})}{\mu^3 x}$

for $0 < x < 1$ and $\mu > 0$.

Value

Returns a gamlss.family object which can be used to fit a UMB distribution in the `gamlss()` function.

Author(s)

David Villegas Ceballos, <david.villegas1@udea.edu.co>

References

Biçer, C., Bakouch, H. S., Biçer, H. D., Alomair, G., Hussain, T., y Almohisen, A. (2024). Unit Maxwell-Boltzmann Distribution and Its Application to Concentrations Pollutant Data. Axioms, 13(4), 226.

See Also

dUMB

Examples

```
# Example 1
# Generating some random values with
# known mu
y <- rUMB(n=300, mu=0.5)

# Fitting the model
library(gamlss)
mod1 <- gamlss(y~1, family=UMB)

# Extracting the fitted values for mu
# using the inverse link function
exp(coef(mod1, what="mu"))

# Example 2
# Generating random values under some model

# A function to simulate a data set with Y ~ UMB
gendat <- function(n) {
  x1 <- runif(n)
  mu <- exp(-0.5 + 1 * x1)
  y <- rUMB(n=n, mu=mu)
  data.frame(y=y, x1=x1)
}
```

```
datos <- gendat(n=300)

mod2 <- gamlss(y~x1, family=UMB, data=datos)
summary(mod2)

# Example 3
# The first dataset measured the concentration of air pollutant CO
# in Alberta, Canada from the Edmonton Central (downtown)
# Monitoring Unit (EDMU) station during 1995.
# Measurements are listed for the period 1976–1995.
# Taken from Bicer et al. (2024) page 12.

data1 <- c(0.19, 0.20, 0.20, 0.27, 0.30,
           0.37, 0.30, 0.25, 0.23, 0.23,
           0.26, 0.23, 0.19, 0.21, 0.20,
           0.22, 0.21, 0.25, 0.25, 0.19)

mod3 <- gamlss(data1 ~ 1, family=UMB)

# Extracting the fitted values for mu
# using the inverse link function
exp(coef(mod3, what="mu"))

# Extraction of the log likelihood
logLik(mod3)

# Example 4
# The second data set measured air quality monitoring of the
# annual average concentration of the pollutant benzo(a)pyrene (BaP).
# The data were obtained from the Edmonton Central (downtown)
# Monitoring Unit (EDMU) location in Alberta, Canada, in 1995.
# Taken from Bicer et al. (2024) page 12.

data2 <- c(0.22, 0.20, 0.25, 0.15, 0.38,
           0.18, 0.52, 0.27, 0.27, 0.27,
           0.13, 0.15, 0.24, 0.37, 0.20)

mod4 <- gamlss(data2 ~ 1, family=UMB)

# Extracting the fitted values for mu
# using the inverse link function
exp(coef(mod4, what="mu"))

# Extraction of the log likelihood
logLik(mod4)

# Replicating figure 5 from Bicer et al. (2024)
# Hist and estimated pdf of Data-I and Data-II
mu1 <- 0.8452875
mu2 <- 0.8593051

# Data-I
```

```
hist(data1, freq = FALSE,
     xlim = c(0, 1.0), ylim = c(0, 10),
     main = "Histogram of Data-I",
     xlab = "y", ylab = "f(y)",
     col = "burlywood1",
     border = "darkgoldenrod4")

curve(dUMB(x, mu = mu1), add = TRUE,
      col = "blue", lwd = 2)

legend("topright", legend = c("UMB"),
       col = c("blue"), lwd = 2, bty = "n")

# Data-II
hist(data2, freq = FALSE,
     xlim = c(0, 1.0), ylim = c(0, 6),
     main = "Histogram of Data-II",
     xlab = "y", ylab = "f(y)",
     col = "burlywood1",
     border = "darkgoldenrod4")

curve(dUMB(x, mu = mu2), add = TRUE,
      col = "blue", lwd = 2)

legend("topright",
       legend = c("UMB"),
       col = c("blue"),
       lwd = 2,
       bty = "n")

# Example 5
# The third dataset measured the concentration of sulphate
# in Calgary from 31 different periods during 1995.
# Taken from Bicer et al. (2024) page 13.

data3 <- c(0.048, 0.013, 0.040, 0.082, 0.073, 0.732, 0.302,
           0.728, 0.305, 0.322,  0.045, 0.261, 0.192,
           0.357, 0.022, 0.143, 0.208, 0.104, 0.330, 0.453,
           0.135, 0.114, 0.049, 0.011, 0.008, 0.037, 0.034,
           0.015, 0.028, 0.069, 0.029)

mod5 <- gamlss(data3 ~ 1, family=UMB)

# Extracting the fitted values for mu
# using the inverse link function
exp(coef(mod5, what="mu"))

# Extraction of the log likelihood
logLik(mod5)

# Example 6
# The fourth dataset measured the concentration of pollutant CO in Alberta, Canada
# from the Calgary northwest (residential) monitoring unit (CRMU) station during 1995.
```

```
# Measurements are listed  for the period 1976-95.
# Taken from Bicer et al. (2024) page 13.

data4 <- c(0.16, 0.19, 0.24, 0.25, 0.30, 0.41, 0.40,
           0.33, 0.23, 0.27, 0.30, 0.32, 0.26, 0.25,
           0.22, 0.22, 0.18, 0.18, 0.20, 0.23)

mod6 <- gamlss(data4 ~ 1, family=UMB)

# Extracting the fitted values for mu
# using the inverse link function
exp(coef(mod6, what="mu"))

# Extraction of the log likelihood
logLik(mod6)

# Replicating figure 6 from Bicer et al. (2024)
# Hist and estimated pdf of Data-III and Data-IV
mu3 <- 1.582003
mu4 <- 0.8161202

# Data-III
hist(data3, freq = FALSE,
     xlim = c(0, 1.0), ylim = c(0, 10),
     main = "Histogram of Data-III",
     xlab = "y", ylab = "f(y)",
     col = "burlywood1",
     border = "darkgoldenrod4")

curve(dUMB(x, mu = mu3), add = TRUE,
      col = "blue", lwd = 2)

legend("topright", legend = c("UMB"),
       col = c("blue"), lwd = 2, bty = "n")

# Data-IV
hist(data4, freq = FALSE,
     xlim = c(0, 1.0), ylim = c(0, 6),
     main = "Histogram of Data-IV",
     xlab = "y", ylab = "f(y)",
     col = "burlywood1",
     border = "darkgoldenrod4")

curve(dUMB(x, mu = mu4), add = TRUE,
      col = "blue", lwd = 2)

legend("topright",
       legend = c("UMB"),
       col = c("blue"),
       lwd = 2,
       bty = "n")
```

UPHN *The Unit-Power Half-Normal family*

Description

The function `UPHN()` defines the Unit-Power Half-Normal distribution, a two parameter distribution, for a `gamlss.family` object to be used in GAMLSS fitting using the function `gamlss()`.

Usage

```
UPHN(mu.link = "log", sigma.link = "log")
```

Arguments

`mu.link`	defines the mu.link, with "log" link as the default for the mu parameter.
`sigma.link`	defines the sigma.link, with "log" link as the default for the sigma.

Details

The UPHN distribution with parameters μ and σ has a support in $(0, 1)$ and density given by

$f(x|\mu, \sigma) = \frac{2\mu}{\sigma x^2}\phi(\frac{1-x}{\sigma x})(2\Phi(\frac{1-x}{\sigma x}) - 1)^{\mu-1}$

for $0 < x < 1$, $\mu > 0$ and $\sigma > 0$.

Value

Returns a `gamlss.family` object which can be used to fit a UPHN distribution in the `gamlss()` function.

Author(s)

Juan Diego Suarez Hernandez, <jsuarezhe@unal.edu.co>

References

Santoro, K. I., Gomez, Y. M., Soto, D., & Barranco-Chamorro, I. (2024). Unit-Power Half-Normal Distribution Including Quantile Regression with Applications to Medical Data. Axioms, 13(9), 599.

See Also

dUPHN.

Examples

```
# Example 1
# Generating random values with
# known mu and sigma
require(gamlss)
mu <- 1.5
sigma <- 4.0

y <- rUPHN(1000, mu, sigma)

mod1 <- gamlss(y~1, sigma.fo=~1,  family=UPHN,
               control=gamlss.control(n.cyc=5000, trace=TRUE))

exp(coef(mod1, what="mu"))
exp(coef(mod1, what="sigma"))

# Example 2
# Generating random values under some model

# A function to simulate a data set with Y ~ UPHN
gendat <- function(n) {
  x1 <- runif(n)
  x2 <- runif(n)
  mu <- exp(0.75 - 0.69 * x1)   # Approx 1.5
  sigma <- exp(0.5 - 0.64 * x2) # Approx 1.20
  y <- rUPHN(n, mu, sigma)
  data.frame(y=y, x1=x1, x2=x2)
}

dat <- gendat(n=2000)

mod2 <- gamlss(y~x1, sigma.fo=~x2, family=UPHN, data=dat,
               control=gamlss.control(n.cyc=5000, trace=TRUE))

summary(mod2)
```

Index

BER, 2, *7*
BER2, 3, *9*

checklink, 5

dBER, *2*, 6
dBER2, *4*, 8
dUHLG, 11, *20*
dUMB, 13, *22*
dUPHN, 16, *17*, *26*

make.link.gamlss, 18

pBER *(*dBER*)*, 6
pBER2 *(*dBER2*)*, 8
pUHLG *(*dUHLG*)*, 11
pUMB *(*dUMB*)*, 13
pUPHN *(*dUPHN*)*, 16

qBER *(*dBER*)*, 6
qBER2 *(*dBER2*)*, 8
qUHLG *(*dUHLG*)*, 11
qUMB *(*dUMB*)*, 13
qUPHN *(*dUPHN*)*, 16

rBER *(*dBER*)*, 6
rBER2 *(*dBER2*)*, 8
rUHLG *(*dUHLG*)*, 11
rUMB *(*dUMB*)*, 13
rUPHN *(*dUPHN*)*, 16

UHLG, *12*, 20
UMB, *15*, 21
UPHN, 26