Package 'gamlss.tr'

July 13, 2020

Type Package

Title Generating and Fitting Truncated `gamlss.family' Distributions

Version 5.1-7

Date 2020-07-13

Author Mikis Stasinopoulos <d.stasinopoulos@londonmet.ac.uk>, Bob
Rigby <r.rigby@londonmet.ac.uk>

Maintainer Mikis Stasinopoulos <d.stasinopoulos@londonmet.ac.uk>

LazyLoad yes

Depends R (>= 2.2.1), gamlss.dist, gamlss (>= 5.0-0), methods

Description This is an add on package to GAMLSS. The purpose of this
package is to allow users to defined truncated distributions in
GAMLSS models. The main function gen.trun() generates truncated
version of an existing GAMLSS family distribution.

License GPL-2 | GPL-3

URL http://www.gamlss.org/

NeedsCompilation no

Repository CRAN

Date/Publication 2020-07-13 13:10:02 UTC

R topics documented:

---

gamlss.tr-package   *Generating and Fitting Truncated 'gamlss.family' Distributions*

Description

This is an add on package to GAMLSS. The purpose of this package is to allow users to defined truncated distributions in GAMLSS models. The main function gen.trun() generates truncated version of an existing GAMLSS family distribution.

Details

The DESCRIPTION file:

Package:	gamlss.tr
Type:	Package
Title:	Generating and Fitting Truncated 'gamlss.family' Distributions
Version:	5.1-7
Date:	2020-07-13
Author:	Mikis Stasinopoulos <d.stasinopoulos@londonmet.ac.uk>, Bob Rigby <r.rigby@londonmet.ac.uk>
Maintainer:	Mikis Stasinopoulos <d.stasinopoulos@londonmet.ac.uk>
LazyLoad:	yes
Depends:	R (>= 2.2.1), gamlss.dist, gamlss (>= 5.0-0), methods
Description:	This is an add on package to GAMLSS. The purpose of this package is to allow users to defined truncated distr
License:	GPL-2	GPL-3
URL:	http://www.gamlss.org/

Index of help topics:

```
fitTail              For fitting truncated distribution to the tails
                     of data
gamlss.tr-package    Generating and Fitting Truncated
                     'gamlss.family' Distributions
gen.trun             Generates a truncated distribution from a
                     gamlss.family
trun                 Fits a Truncate Distribution from a
                     gamlss.family
trun.d               Truncated Probability Density Function of a
                     gamlss.family Distribution
trun.p               Truncated Cumulative Density Function of a
                     gamlss.family Distribution
trun.q               Truncated Inverse Cumulative Density Function
                     of a gamlss.family Distribution
trun.r               Generates Random Values from a Truncated
                     Density Function of a gamlss.family
                     Distribution
```

Author(s)

Mikis Stasinopoulos <d.stasinopoulos@londonmet.ac.uk>, Bob Rigby <r.rigby@londonmet.ac.uk>

Maintainer: Mikis Stasinopoulos <d.stasinopoulos@londonmet.ac.uk>

References

Rigby, R. A. and Stasinopoulos D. M. (2005). Generalized additive models for location, scale and shape,(with discussion), *Appl. Statist.*, **54**, part 3, pp 507-554.

Stasinopoulos D. M., Rigby R.A., Heller G., Voudouris V., and De Bastiani F., (2017) *Flexible Regression and Smoothing: Using GAMLSS in R*, Chapman and Hall/CRC.

(see also <http://www.gamlss.org/>).

Examples

```
# generating a t-distribution from 0 to 100
gen.trun(par=c(0,100),family="TF", name="0to100", type="both")
op<-par(mfrow=c(2,2))
plot(function(x) dTF0to100(x, mu=80 ,sigma=20, nu=5), 0, 100, ylab="pdf")
plot(function(x) pTF0to100(x, mu=80 ,sigma=20, nu=5), 0, 100, ylab="cdf")
plot(function(x) qTF0to100(x, mu=80 ,sigma=20, nu=5), 0.01, .999, ylab="invcdf")
hist(s1<-rTF0to100(1000, mu=80 ,sigma=20, nu=5), ylab="hist", xlab="x", main="generated data")
par(op)
```

---

fitTail	*For fitting truncated distribution to the tails of data*

Description

There are two functions here. The function `fitTail()` which fits a truncated distribution to certain percentage of the tail of a response variable and the function `fitTailAll()` which does a sequence of truncated fits. Plotting the results from those fits is analogous to the Hill plot, Hill (1975).

Usage

```
fitTail(y, family = "WEI3", percentage = 10, howmany = NULL,
      type = c("right", "left"), ...)

fitTailAll(y, family = "WEI3", percentage = 10, howmany = NULL,
      type = c("right", "left"), plot = TRUE,
      print = TRUE, save = FALSE, start = 5, trace = 0,  ...)
```

Arguments

y	The variable of interest
family	a gamlsss.family distribution
percentage	what percentage of the tail need to be modelled, default is 10%

howmany	how many observations in the tail needed. This is an alternative to `percentage`. If it specified it take over from the `percentage` argument otherwise `percentage` is used.
type	which tall needs checking the right (default) of the left
plot	whether to plot with default equal `TRUE`
print	whether to print the coefficients with default equal `TRUE`
save	whether to save the fitted linear model with default equal `FALSE`
start	where to start fitting from the tail of the data
trace	0: no output 1: minimal 2: print estimates
...	for further argument to the fitting function

Details

The idea here is to fit a truncated distribution to the tail of the data. Truncated log-normal and Weibull distributions could be appropriate distributions. More details can be found in Chapter 6 of "The Distribution Toolbox of GAMLSS" book which can be found in http://www.gamlss.org/).

Value

A fitted gamlss model

Author(s)

Bob Rigby, Mikis Stasinopoulos and Vlassios Voudouris

References

Hill B. M. (1975) A Simple General Approach to Inference About the Tail of a Distribution *Ann. Statist.* Volume **3**, Number 5, pp 1163-1174.

Rigby, R. A. and Stasinopoulos D. M. (2005). Generalized additive models for location, scale and shape,(with discussion), *Appl. Statist.*, **54**, part 3, pp 507-554.

Stasinopoulos D. M., Rigby R.A. and Akantziliotou C. (2006) Instructions on how to use the GAMLSS package in R. Accompanying documentation in the current GAMLSS help files, (see also http://www.gamlss.org/).

Stasinopoulos D. M. Rigby R.A. (2007) Generalized additive models for location scale and shape (GAMLSS) in R. *Journal of Statistical Software*, Vol. **23**, Issue 7, Dec 2007, http://www.jstatsoft.org/v23/i07.

See Also

loglogSurv, logSurv

Examples

```
data(film90)
F90 <- exp(film90$lborev1)# original scale
# trucated plots
# 10%
w403<- fitTail(F90, family=WEI3)
qqnorm(resid(w403))
abline(0,1, col="red")

## Not run:
# hill -sequential plot 10
w1<-fitTailAll(F90)
# plot sigma
plot(w1[,2])
#-----------------
#LOGNO
l403<- fitTail(F90, family=LOGNO)
plot(l403)
qqnorm(resid(l403))
abline(0,1)
#  hill -sequential plot 10
l1<-fitTailAll(F90, family=LOGNO)
plot(l1[,2])
#------------------------

## End(Not run)
```

---

gen.trun   *Generates a truncated distribution from a gamlss.family*

Description

The gen.trun() function allows the user to generate d, p, q, and r distribution functions plus an extra gamlss.family function for fitting a truncated distribution with gamlss.

For continuous distributions left truncation at 3 means that the random variable can take the value 3. For discrete distributions left truncation at 3 means that the random variable can take values from 4 onwards. This is the same for right truncation. Truncation at 15 for a discrete variable means that 15 and greater values are not allowed but for continuous variable it mean values greater that 15 are not allowed (so 15 is a possible value).

If the user want a different link (rather the default) for any of the parameters she/he has to declare at the generation of the functions, see example.

Usage

```
gen.trun(par = c(0), family = "NO", name = "tr",
         type = c("left", "right", "both"),
         varying = FALSE,  print=TRUE, ...)
```

Arguments

par	a vector with one (for "left" or "right" truncation) or two elements for "both". When the argument varying = TRUE then par can be a vector or a matrix with two columns respectively.
family	a gamlss.family object, which is used to define the distribution and the link functions of the various parameters. The distribution families supported by gamlss() can be found in gamlss.family.
name	the extra characters to be added to the name of new truncated distribution, by default it adds tr
type	whether "left", "right" or in "both" sides truncation is required
varying	whether the truncation varies for different observations. This can be useful in regression analysis. If varying = TRUE then par should be an n-length vector for type equal "left" and "right" and an n by 2 matrix for type="both"
print	whether to print the names of the created distribution
...	for extra arguments

Value

Returns the d, the p, the q, the r and the fitting functions of a truncated gamlss.family distribution.

Author(s)

Mikis Stasinopoulos <mikis.stasinopoulos@gamlss.org> and Bob Rigby

References

Rigby, R. A. and Stasinopoulos D. M. (2005). Generalized additive models for location, scale and shape,(with discussion), *Appl. Statist.*, **54**, part 3, pp 507-554.

Stasinopoulos D. M., Rigby R.A. and Akantziliotou C. (2003) Instructions on how to use the GAMLSS package in R. Accompanying documentation in the current GAMLSS help files, (see also http://www.gamlss.org/).

See Also

trun.d, trun.p, trun.q, trun.r

Examples

```
# generating a t-distribution from 0 to 100
gen.trun(par=c(0,100),family="TF", name="0to100", type="both")
op<-par(mfrow=c(2,2))
plot(function(x) dTF0to100(x, mu=80 ,sigma=20, nu=5), 0, 100, ylab="pdf")
plot(function(x) pTF0to100(x, mu=80 ,sigma=20, nu=5), 0, 100, ylab="cdf")
plot(function(x) qTF0to100(x, mu=80 ,sigma=20, nu=5), 0.01, .999, ylab="invcdf")
hist(s1<-rTF0to100(1000, mu=80 ,sigma=20, nu=5), ylab="hist", xlab="x",
           main="generated data")
par(op)
m1<-histDist(s1, family=TF0to100, xlim=c(0,100))# fitting the data
```

```
# using the argumnt varying
# left part varies right part equal 100
leftPAR <- rPO(100)
gen.trun(par=cbind(leftPAR,rep(100, 100)),family="TF", name="0to100Varying",
          type="both", varying=TRUE)
YY<- rTF0to100Varying(100, mu=80, sigma=20, nu=5)
m1<-gamlss(YY~1, family=TF0to100Varying)
m1
```

---

trun	*Fits a Truncate Distribution from a gamlss.family*

Description

This function can be used to fit truncated distributions. It takes as an argument an existing GAMLSS family distribution and a parameter vector, of the type c(left.value, right.value), and generates a gamlss.family object which then can be used to fit a truncated distribution.

Usage

```
trun(par = c(0), family = "NO",  type = c("left", "right", "both"), name = "tr",
        local = TRUE, delta=NULL, varying = FALSE, ...)
```

Arguments

par
: a vector with one (for "left" or "right" truncation) or two elements for "both". When the argument varying = TRUE then par can be a vector or a matrix with two columns respectively.

family
: an existing gamlss.family distribution

type
: what type of truncation is required, left, right or both. If both the par should be a vector of length two. (the default is left truncation)

name
: a character string to be added to name of the created object i.e. with family=TF and name=trZero the gamlss.family object will be called TFtrZero

local
: if TRUE the function will try to find the environment of gamlss to generate the d and p functions required for the fitting, if FALSE the functions will be generated in the global environment

delta
: the delta increment used in the numerical derivatives

varying
: whether the truncation varies for diferent observations. This can be usefull in regression analysis. If varying = TRUE then par should be an n-length vector for type equal "left" and "right" and an n by 2 matrix for type="both"

...
: for extra arguments

Details

This function is created to help the user to fit a truncated form of existing `gamlss` distribution. It does this by taking an existing `gamlss.family` and changing some of the components of the distribution to help the fitting process. It particular it i) creates a pdf (d) and a cdf (p) function within `gamlss`, ii) changes the global deviance function `G.dev.incr`, the first derivative functions (see note below) and the quantile residual function.

Value

It returns a `gamlss.family` object which has all the components needed for fitting a distribution in `gamlss`.

Note

This function is experimental and could be changed. The function `trun` changes the first derivatives of the original gamlss family d function to numerical derivatives for the new truncated d function. The default increment `delta`, for this numerical derivatives function, is `eps * pmax(abs(x),1)` where `eps<-sqrt(.Machine$double.eps)`. The default `delta` could be inappropriate for specific applications and can be overwritten by using the argument `delta`.

Author(s)

Mikis Stasinopoulos <mikis.stasinopoulos@gamlss.org> and Bob Rigby

References

Rigby, R. A. and Stasinopoulos D. M. (2005). Generalized additive models for location, scale and shape,(with discussion), *Appl. Statist.*, **54**, part 3, pp 507-554.

Stasinopoulos D. M., Rigby R.A. and Akantziliotou C. (2003) Instructions on how to use the GAMLSS package in R. Accompanying documentation in the current GAMLSS help files, (see also http://www.gamlss.org/).

See Also

trun.d, trun.p, trun.q, trun.r, gen.trun

Examples

```
# generate a left truncated zero t family
gen.trun(0,family="TF")
# take a random sample of 1000 observations
sam<-rTFtr(1000,mu=10,sigma=5, nu=5 )
hist(sam)
# fit the distribution to the data
mod1<-gamlss(sam~1, family=trun(0,TF))
mod1
# now create a gamlss.family object before the fitting
Ttruc.Zero<- trun(par=0,family=TF, local=FALSE)
mod2<-gamlss(sam~1, family=Ttruc.Zero)
# now check the sensitivity of delta
```

```
Ttruc.Zero<- trun(par=0,family=TF, local=FALSE, delta=c(0.01,0.01, 0.01))
mod3<-gamlss(sam~1, family=Ttruc.Zero)
```

---

trun.d *Truncated Probability Density Function of a gamlss.family Distribution*

Description

Creates a truncated probability density function version from a current GAMLSS family distribution

For continuous distributions left truncation at 3 means that the random variable can take the value 3. For discrete distributions left truncation at 3 means that the random variable can take values from 4 onwards. This is the same for right truncation. Truncation at 15 for a discrete variable means that 15 and greater values are not allowed but for continuous variable it mean values greater that 15 are not allowed (so 15 is a possible value).

Usage

```
trun.d(par, family = "NO", type = c("left", "right", "both"),
       varying = FALSE, ...)
```

Arguments

par   a vector with one (for "left" or "right" truncation) or two elements for "both". When the argument varying = TRUE then par can be a vector or a matrix with two columns respectively.

family   a gamlss.family object, which is used to define the distribution and the link functions of the various parameters. The distribution families supported by gamlss() can be found in gamlss.family. Functions such as BI() (binomial) produce a family object.

type   whether left, right or in both sides truncation is required, (left is the default).

varying   whether the truncation varies for diferent observations. This can be usefull in regression analysis. If varying = TRUE then par should be an n-length vector for type equal "left" and "right" and an n by 2 matrix for type="both"

...   for extra arguments

Value

Returns a d family function

Author(s)

Mikis Stasinopoulos <mikis.stasinopoulos@gamlss.org> and Bob Rigby

References

Rigby, R. A. and Stasinopoulos D. M. (2005). Generalized additive models for location, scale and shape,(with discussion), *Appl. Statist.*, **54**, part 3, pp 507-554.

Stasinopoulos D. M., Rigby R.A. and Akantziliotou C. (2003) Instructions on how to use the GAMLSS package in R. Accompanying documentation in the current GAMLSS help files, (see also <http://www.gamlss.org/>).

See Also

trun.p, trun.q, trun.r, gen.trun

Examples

```
#-------------------------------------------------------------------------------------------
# continuous distribution
# left truncation
test1<-trun.d(par=c(0), family="TF", type="left")
test1(1)
dTF(1)/(1-pTF(0))
if(abs(test1(1)-(dTF(1)/pTF(0)))>0.00001) stop("error in left trucation")
test1(1, log=TRUE)
log(dTF(1)/(1-pTF(0)))
if(abs(test1(1, log=TRUE)-log(dTF(1)/pTF(0)))>0.00001)
                    stop("error in left trucation")
integrate(function(x) test1(x, mu=-2, sigma=1, nu=1),0,Inf)
# the pdf is defined even with negative mu
integrate(function(x) test1(x, mu=0, sigma=10, nu=1),0,Inf)
integrate(function(x) test1(x, mu=5, sigma=5, nu=10),0,Inf)
plot(function(x) test1(x, mu=-3, sigma=1, nu=1),0,10)
plot(function(x) test1(x, mu=3, sigma=5, nu=10),0,10)
#-------------------------------------------------------------------------------------
# right truncation
test2<-trun.d(par=c(10), family="BCT", type="right")
test2(1)
dBCT(1)/(pBCT(10))
#if(abs(test2(1)-(dBCT(1)/pBCT(10)))>0.00001) stop("error in right trucation")
test2(1, log=TRUE)
log(dBCT(1)/(pBCT(10)))
if(abs(test2(1, log=TRUE)-log(dBCT(1)/(pBCT(10))))>0.00001)
                    stop("error in right trucation")
integrate(function(x) test2(x, mu=2, sigma=1, nu=1),0,10)
integrate(function(x) test2(x, mu=2, sigma=.1, nu=1),0,10)
integrate(function(x) test2(x, mu=2, sigma=.1, nu=10),0,10)
plot(function(x) test2(x, mu=2, sigma=.1, nu=1),0,10)
plot(function(x) test2(x, mu=2, sigma=1, nu=1),0,10)
#-------------------------------------------------------------------------------------
# both left and right truncation
test3<-trun.d(par=c(-3,3), family="TF", type="both")
test3(0)
dTF(0)/(pTF(3)-pTF(-3))
if(abs(test3(0)-dTF(0)/(pTF(3)-pTF(-3)))>0.00001)
```

```
                stop("error in right trucation")
test3(0, log=TRUE)
log(dTF(0)/(pTF(3)-pTF(-3)))
if(abs(test3(0, log=TRUE)-log(dTF(0)/(pTF(3)-pTF(-3))))>0.00001
            stop("error in both trucation")
plot(function(x) test3(x, mu=0, sigma=1, nu=1),-3,3)
integrate(function(x) test3(x, mu=2, sigma=1, nu=1),-3,3)
#-------------------------------------------------------------------------------------
# discrete distribution
# left
# Poisson truncated at zero means zero is excluded
test4<-trun.d(par=c(0), family="PO", type="left")
test4(1)
dPO(1)/(1-pPO(0))
if(abs(test4(1)-dPO(1)/(1-pPO(0)))>0.00001) stop("error in left trucation")
test4(1, log=TRUE)
log(dPO(1)/(1-pPO(0)))
if(abs(test4(1, log=TRUE)-log(dPO(1)/(1-pPO(0))))>0.00001)
                stop("error in left trucation")
sum(test4(x=1:20, mu=2)) #
sum(test4(x=1:200, mu=80)) #
plot(function(x) test4(x, mu=20), from=1, to=51, n=50+1, type="h") # pdf
# right
# right truncated at 10 means 10 is excluded
test5<-trun.d(par=c(10), family="NBI", type="right")
test5(2)
dNBI(2)/(pNBI(9))
if(abs(test5(1)-dNBI(1)/(pNBI(9)))>0.00001) stop("error in right trucation")
test5(1, log=TRUE)
log(dNBI(1)/(pNBI(9)))
if(abs(test5(1, log=TRUE)-log(dNBI(1)/(pNBI(9))))>0.00001) stop("error in right trucation")
sum(test5(x=0:9, mu=2,   sigma=2)) #
sum(test5(x=0:9, mu=300, sigma=5)) # can have mu > parameter
plot(function(x) test5(x, mu=20, sigma=3), from=0, to=9, n=10, type="h") # pdf
plot(function(x) test5(x, mu=300, sigma=5), from=0, to=9, n=10, type="h") # pdf
#-------------------------------------------------------------------------------------
# both
test6<-trun.d(par=c(0,10), family="NBI", type="both")
test6(2)
dNBI(2)/(pNBI(9)-pNBI(0))
if(abs(test6(2)-dNBI(2)/(pNBI(9)-pNBI(0)))>0.00001)
        stop("error in right trucation")
test6(1, log=TRUE)
log(dNBI(1)/(pNBI(9)-pNBI(0)))
if(abs(test6(1, log=TRUE)-log(dNBI(1)/(pNBI(9)-pNBI(0))))>0.00001)
  stop("error in right trucation")
sum(test6(x=1:9, mu=2,   sigma=2)) #
sum(test6(x=1:9, mu=100, sigma=5)) # can have mu > parameter
plot(function(x) test6(x, mu=20, sigma=3), from=1, to=9, n=9, type="h") # pdf
plot(function(x) test6(x, mu=300, sigma=.4), from=1, to=9, n=9, type="h") # pdf
#-------------------------------------------------------------------------------------
# now try when the trucated points varies for each observarion
# this will be appropriate for regression models only
```

```
# continuous
#-------------------------------------------------------------------------------------
# left truncation
test7<-trun.d(par=c(0,1,2), family="TF", type="left", varying=TRUE)
test7(c(1,2,3))
dTF(c(1,2,3))/(1-pTF(c(0,1,2)))
test7(c(1,2,3), log=TRUE)
#-------------------------------------------------------------------------------------
# right truncation
test8<-trun.d(par=c(10,11,12), family="BCT", type="right", varying=TRUE)
test8(c(1,2,3))
dBCT(c(1,2,3))/(pBCT(c(10,11,12)))
test8(c(1,2,3), log=TRUE)
#-------------------------------------------------------------------------------------
# both left and right truncation
test9<-trun.d(par=cbind(c(0,1,2),c(10,11,12) ), family="TF", type="both",
              varying=TRUE)
test9(c(1,2,3))
dTF(c(1,2,3))/ (pTF(c(10,11,12))-pTF(c(0,1,2)))
test3(c(1,2,3), log=TRUE)
#-------------------------------------------------------------------------------------
# discrete
# left
test10<-trun.d(par=c(0,1,2), family="PO", type="left", varying=TRUE)
test10(c(1,2,3))
dPO(c(1,2,3))/(1-pPO(c(0,1,2)))
# right
test11<-trun.d(par=c(10,11,12), family="NBI", type="right", varying=TRUE)
test11(c(1,2,3))
dNBI(c(1,2,3))/pNBI(c(9,10,11))
# both
test12<-trun.d(par=rbind(c(0,10), c(1,11), c(2,12)), family="NBI", type="both", varying=TRUE)
test12(c(2,3,4))
dNBI(c(2,3,4))/(pNBI(c(9,10,11))-pNBI(c(0,1,2)))
```

trun.p	*Truncated Cumulative Density Function of a gamlss.family Distribution*

Description

Creates a truncated cumulative density function version from a current GAMLSS family distribution.

For continuous distributions left truncation at 3 means that the random variable can take the value 3. For discrete distributions left truncation at 3 means that the random variable can take values from 4 onwards. This is the same for right truncation. Truncation at 15 for a discrete variable means that 15 and greater values are not allowed but for continuous variable it mean values greater that 15 are not allowed (so 15 is a possible value).

Usage

```
trun.p(par, family = "NO", type = c("left", "right", "both"),
        varying = FALSE, ...)
```

Arguments

par
: a vector with one (for "left" or "right" truncation) or two elements for "both". When the argument varying = TRUE then par can be a vector or a matrix with two columns respectively.

family
: a gamlss.family object, which is used to define the distribution and the link functions of the various parameters. The distribution families supported by gamlss() can be found in gamlss.family. Functions such as BI() (binomial) produce a family object.

type
: whether left, right or in both sides truncation is required, (left is the default)

varying
: whether the truncation varies for diferent observations. This can be usefull in regression analysis. If varying = TRUE then par should be an n-length vector for type equal "left" and "right" and an n by 2 matrix for type="both"

...
: for extra arguments

Value

Return a p family function

Author(s)

Mikis Stasinopoulos <mikis.stasinopoulos@gamlss.org> and Bob Rigby

References

Rigby, R. A. and Stasinopoulos D. M. (2005). Generalized additive models for location, scale and shape,(with discussion), *Appl. Statist.*, **54**, part 3, pp 507-554.

Stasinopoulos D. M., Rigby R.A. and Akantziliotou C. (2003) Instructions on how to use the GAMLSS package in R. Accompanying documentation in the current GAMLSS help files, (see also http://www.gamlss.org/).

See Also

trun.d, trun.q, trun.r, gen.trun

Examples

```
# trucated p  continuous function
# continuous
#------------------------------------------------------------------------------------
# left
test1<-trun.p(par=c(0), family="TF", type="left")
test1(1)
(pTF(1)-pTF(0))/(1-pTF(0))
```

```
if(abs(test1(1)-(pTF(1)-pTF(0))/(1-pTF(0)))>0.00001)
                        stop("error in left trucation of p")
plot(function(x) test1(x, mu=2, sigma=1, nu=2),0,10)
#----------------------------------------------------------------------------------------
# right
test2 <- trun.p(par=c(10), family="BCT", type="right")
test2(1)
pBCT(1)/pBCT(10)
if(abs(test2(1)-pBCT(1)/pBCT(10))>0.00001) stop("error in right trucation")
test2(1, lower.tail=FALSE)
1-pBCT(1)/pBCT(10)
if(abs(test2(1, lower.tail=FALSE)-(1-pBCT(1)/pBCT(10)))>0.00001)
                 stop("error in right trucation")
test2(1, log.p=TRUE)
log(pBCT(1)/pBCT(10))
if(abs(test2(1, log.p=TRUE)-log(pBCT(1)/pBCT(10)))>0.00001)
                stop("error in right trucation")
plot(function(x) test2(x, mu=2, sigma=1, nu=2, tau=2),0,10)
plot(function(x) test2(x, mu=2, sigma=1, nu=2, tau=2,
                        lower.tail=FALSE),0,10)
#----------------------------------------------------------------------------------------
# both
test3<-trun.p(par=c(-3,3), family="TF", type="both")
test3(1)
(pTF(1)-pTF(-3))/(pTF(3)-pTF(-3))
if(abs(test3(1)-(pTF(1)-pTF(-3))/(pTF(3)-pTF(-3)))>0.00001)
              stop("error in right trucation")
test3(1, lower.tail=FALSE)
1-(pTF(1)-pTF(-3))/(pTF(3)-pTF(-3))
if(abs(test3(0,lower.tail=FALSE)-
            (1-(pTF(0)-pTF(-3))/(pTF(3)-pTF(-3))))>0.00001)
      stop("error in right trucation")
plot(function(x) test3(x, mu=2, sigma=1, nu=2, ),-3,3)
plot(function(x) test3(x, mu=2, sigma=1, nu=2, lower.tail=FALSE),-3,3)
#----------------------------------------------------------------------------------------
# Discrete
#----------------------------------------------------------------------------------------
# trucated p function
# left
test4<-trun.p(par=c(0), family="PO", type="left")
test4(1)
(pPO(1)-pPO(0))/(1-pPO(0))
if(abs(test4(1)-(pPO(1)-pPO(0))/(1-pPO(0)))>0.00001)
                stop("error in left trucation of p")
plot(function(x) test4(x, mu=2), from=1, to=10, n=10, type="h")
cdf <- stepfun(1:40, test4(1:41, mu=5), f = 0)
plot(cdf, main="cdf", ylab="cdf(x)", do.points=FALSE )
#----------------------------------------------------------------------------------------
# right
test5<-trun.p(par=c(10), family="NBI", type="right")
test5(2)
pNBI(2)/(pNBI(9))
if(abs(test5(2)-(pNBI(2)/(pNBI(9))))>0.00001)
```

```
            stop("error in right trucation of p")
plot(function(x) test5(x, mu=2), from=0, to=9, n=10, type="h")
cdf <- stepfun(0:8, test5(0:9, mu=5), f = 0)
plot(cdf, main="cdf", ylab="cdf(x)", do.points=FALSE )
#--------------------------------------------------------------------------------
# both
test6<-trun.p(par=c(0,10), family="NBI", type="both")
test6(2)
(pNBI(2)-pNBI(0))/(pNBI(9)-pNBI(0))
if(abs(test6(2)-(pNBI(2)-pNBI(0))/(pNBI(9)-pNBI(0)))>0.00001)
                stop("error in the both trucation")
test6(1, log=TRUE)
log((pNBI(1)-pNBI(0))/(pNBI(9)-pNBI(0)))
if(abs(test6(1, log=TRUE)-log((pNBI(1)-pNBI(0))/(pNBI(9)-pNBI(0))))>0.00001)
              stop("error in both trucation")
plot(function(y) test6(y, mu=20, sigma=3), from=1, to=9, n=9, type="h")
plot(function(y) test6(y, mu=300, sigma=.4), from=1, to=9, n=9, type="h")
cdf <- stepfun(1:8, test6(1:9, mu=5), f = 0)
plot(cdf, main="cdf", ylab="cdf(x)", do.points=FALSE )
#--------------------------------------------------------------------------------
# varying   truncation
#--------------------------------------------------------------------------------
# coninuous
# left
test6<-trun.p(par=c(0,1,2), family="TF", type="left", varying=TRUE)
test6(c(2,3,4))
(pTF(c(2,3,4))-pTF(c(0,1,2)))/(1-pTF(c(0,1,2)))
test6(c(2,3,4), log.p=TRUE)
#--------------------------------------------------------------------------------
# right
test7 <- trun.p(par=c(10,11,12), family="BCT", type="right", varying=TRUE)
test7(c(1,2,3))
pBCT(c(1,2,3))/pBCT(c(10,11,12))
test7(c(1,2,3), lower.tail=FALSE)
1-pBCT(c(1,2,3))/pBCT(c(10,11,12))
test7(c(1,2,3), log.p=TRUE)
#-------------------------------------------------------------------------------
# both
test8<-trun.p(par=cbind(c(0,1,2), c(10,11,12)), family="TF",
            type="both", varying=TRUE)
test8(c(1,2,3))
(pTF(c(1,2,3))-pTF(c(0,1,2)))/(pTF(c(10,11,12))-pTF(c(0,1,2)))
test8(c(1,2,3), lower.tail=FALSE)
1-(pTF(c(1,2,3))-pTF(c(0,1,2)))/(pTF(c(10,11,12))-pTF(c(0,1,2)))
#--------------------------------------------------------------------------------
# discrete
#--------------------------------------------------------------------------------
# left
test9<-trun.p(par=c(0,1,2), family="PO", type="left", varying=TRUE)
test9(c(1,2,3))
(pPO(c(1,2,3))-pPO(c(0,1,2)))/(1-pPO(c(0,1,2)))
#--------------------------------------------------------------------------------
# right
```

```
test10<-trun.p(par=c(10,11,12), family="NBI", type="right", varying=TRUE)
test10(c(2,3,4))
pNBI(c(2,3,4))/(pNBI(c(9,10,11)))
#-------------------------------------------------------------------------------
# both
test11<-trun.p(par=rbind(c(0,10), c(1,11), c(2, 12)), family="NBI",
                type="both", varying=TRUE)
test11(c(2,3,4))
(pNBI(c(2,3,4))-pNBI(c(0,1,2)))/(pNBI(c(9,10,11))-pNBI(c(0,1,2)))
#-------------------------------------------------------------------------------
```

---

trun.q   *Truncated Inverse Cumulative Density Function of a gamlss.family*
                                    Distribution

Description

Creates a function to produce the inverse of a truncated cumulative density function generated from a current GAMLSS family distribution.

For continuous distributions left truncation at 3 means that the random variable can take the value 3. For discrete distributions left truncation at 3 means that the random variable can take values from 4 onwards. This is the same for right truncation. Truncation at 15 for a discrete variable means that 15 and greater values are not allowed but for continuous variable it mean values greater that 15 are not allowed (so 15 is a possible value).

Usage

```
trun.q(par, family = "NO", type = c("left", "right", "both"),
       varying = FALSE, ...)
```

Arguments

par	a vector with one (for "left" or "right" truncation) or two elements for "both". When the argument varying = TRUE then par can be a vector or a matrix with two columns respectively.
family	a gamlss.family object, which is used to define the distribution and the link functions of the various parameters. The distribution families supported by gamlss() can be found in gamlss.family. Functions such as BI() (binomial) produce a family object.
type	whether left, right or in both sides truncation is required, (left is the default)
varying	whether the truncation varies for diferent observations. This can be usefull in regression analysis. If varying = TRUE then par should be an n-length vector for type equal "left" and "right" and an n by 2 matrix for type="both"
...	for extra arguments

Value

Returns a q family function

Author(s)

Mikis Stasinopoulos <mikis.stasinopoulos@gamlss.org> and Bob Rigby

References

Rigby, R. A. and Stasinopoulos D. M. (2005). Generalized additive models for location, scale and shape,(with discussion), *Appl. Statist.*, **54**, part 3, pp 507-554.

Stasinopoulos D. M., Rigby R.A. and Akantziliotou C. (2003) Instructions on how to use the GAMLSS package in R. Accompanying documentation in the current GAMLSS help files, (see also http://www.gamlss.org/).

See Also

trun.d, trun.q, trun.r, gen.trun

Examples

```
# trucated q  continuous function
# continuous
#------------------------------------------------------------------------------------------
# left
test1<-trun.q(par=c(0), family="TF", type="left")
test1(.6)
qTF(pTF(0)+0.6*(1-pTF(0)))
#------------------------------------------------------------------------------------------
# right
test2 <- trun.q(par=c(10), family="BCT", type="right")
test2(.6)
qBCT(0.6*pBCT(10))
#------------------------------------------------------------------------------------------
# both
test3<-trun.q(par=c(-3,3), family="TF", type="both")
test3(.6)
qTF(0.6*(pTF(3)-pTF(-3))+pTF(-3))
#------------------------------------------------------------------------------------------
# varying  par
#------------------------------------------------------------------------------------------
# left
test7<-trun.q(par=c(0,1,2), family="TF", type="left", varying=TRUE)
test7(c(.5,.5,.6))
qTF(pTF(c(0,1,2))+c(.5,.5,.6)*(1-pTF(c(0,1,2))))
#------------------------------------------------------------------------------------------
# right
test9 <- trun.q(par=c(10,11,12), family="BCT", type="right", varying=TRUE)
test9(c(.5,.5,.6))
qBCT(c(.5,.5,.6)*pBCT(c(10,11,12)))
#------------------------------------------------------------------------------------------
# both
test10<-trun.q(par=cbind(c(0,1,2), c(10,11,12)), family="TF", type="both", varying=TRUE)
test10(c(.5, .5, .7))
qTF(c(.5, .5, .7)*(pTF(c(10,11,12))-pTF(c(0,1,2)))+pTF(c(0,1,2)))
```

```
#-------------------------------------------------------------------------------------
# FOR DISCRETE DISTRIBUTIONS
# trucated q function
# left
test4<-trun.q(par=c(0), family="PO", type="left")
test4(.6)
qPO(pPO(0)+0.6*(1-pPO(0)))
# varying
test41<-trun.q(par=c(0,1,2), family="PO", type="left", varying=TRUE)
test41(c(.6,.4,.5))
qPO(pPO(c(0,1,2))+c(.6,.4,.5)*(1-pPO(c(0,1,2))))
#-------------------------------------------------------------------------------------
# right
test5 <- trun.q(par=c(10), family="NBI", type="right")
test5(.6)
qNBI(0.6*pNBI(10))
test5(.6, mu=10, sigma=2)
qNBI(0.6*pNBI(10, mu=10, sigma=2), mu=10, sigma=2)
# varying
test51 <- trun.q(par=c(10, 11, 12), family="NBI", type="right", varying=TRUE)
test51(c(.6,.4,.5))
qNBI(c(.6,.4,.5)*pNBI(c(10, 11, 12)))
test51(c(.6,.4,.5), mu=10, sigma=2)
qNBI(c(.6,.4,.5)*pNBI(c(10, 11, 12), mu=10, sigma=2), mu=10, sigma=2)
#-------------------------------------------------------------------------------------
# both
test6<-trun.q(par=c(0,10), family="NBI", type="both")
test6(.6)
qNBI(0.6*(pNBI(10)-pNBI(0))+pNBI(0))
# varying
test61<-trun.q(par=cbind(c(0,1,2), c(10,11,12)), family="NBI", type="both", varying=TRUE)
test61(c(.6,.4,.5))
qNBI(c(.6,.4,.5)*(pNBI(c(10,11,12))-pNBI(c(0,1,2)))+pNBI(c(0,1,2)))
#-------------------------------------------------------------------------------------
```

| trun.r | *Generates Random Values from a Truncated Density Function of a gamlss.family Distribution* |

Description

Creates a function to generate randon values from a truncated probability density function created from a current GAMLSS family distribution

For continuous distributions left truncation at 3 means that the random variable can take the value 3. For discrete distributions left truncation at 3 means that the random variable can take values from 4 onwards. This is the same for right truncation. Truncation at 15 for a discrete variable means that 15 and greater values are not allowed but for continuous variable it mean values greater that 15 are not allowed (so 15 is a possible value).

Usage

```
trun.r(par, family = "NO", type = c("left", "right", "both"),
          varying = FALSE, ...)
```

Arguments

par   a vector with one (for "left" or "right" truncation) or two elements for "both".
               When the argument varying = TRUE then par can be a vector or a matrix with
               two columns respectively.

family   a gamlss.family object, which is used to define the distribution and the link
               functions of the various parameters. The distribution families supported by
               gamlss() can be found in gamlss.family. Functions such as BI() (binomial)
               produce a family object.

type   whether left, right or in both sides truncation is required, (left is the default)

varying   whether the truncation varies for diferent observations. This can be usefull in
               regression analysis. If varying = TRUE then par should be an n-length vector
               for type equal "left" and "right" and an n by 2 matrix for type="both"

...   for extra arguments

Value

Returns a r family function

Author(s)

Mikis Stasinopoulos <mikis.stasinopoulos@gamlss.org> and Bob Rigby

References

Rigby, R. A. and Stasinopoulos D. M. (2005). Generalized additive models for location, scale and
shape,(with discussion), *Appl. Statist.*, **54**, part 3, pp 507-554.

Stasinopoulos D. M., Rigby R.A. and Akantziliotou C. (2003) Instructions on how to use the
GAMLSS package in R. Accompanying documentation in the current GAMLSS help files, (see
also http://www.gamlss.org/).

See Also

trun.p, trun.q, trun.d, gen.trun

Examples

```
# trucated r function
# continuous
#------------------------------------------------------------------------------------
# left
test1<-trun.r(par=c(0), family="TF", type="left")
rr<-test1(1000)
hist(rr)
```

```
#------------------------------------------------------------------------------------------
# right
test2 <- trun.r(par=c(10), family="BCT", type="right")
rr<-test2(1000)
hist(rr)
#------------------------------------------------------------------------------------------
# both
test3<-trun.r(par=c(-3,3), family="TF", type="both")
rr<-test3(1000)
hist(rr)
#------------------------------------------------------------------------------------------
# discrete
# trucated r function
# left
test4<-trun.r(par=c(0), family="PO", type="left")
tN <- table(Ni <- test4(1000))
r <- barplot(tN, col='lightblue')
#------------------------------------------------------------------------------------------
# right
test5 <- trun.r(par=c(10), family="NBI", type="right")
tN <- table(Ni <- test5(1000))
r <- barplot(tN, col='lightblue')
tN <- table(Ni <- test5(1000,mu=5))
r <- barplot(tN, col='lightblue')
tN <- table(Ni <- test5(1000,mu=10, sigma=.1))
r <- barplot(tN, col='lightblue')
#------------------------------------------------------------------------------------------
# both
test6<-trun.r(par=c(0,10), family="NBI", type="both")
tN <- table(Ni <- test6(1000,mu=5))
r <- barplot(tN, col='lightblue')
#------------------------------------------------------------------------------------------
# varying = TRUE
#------------------------------------------------------------------------------------------
# continuous
#------------------------------------------------------------------------------------------
# left
test7<-trun.r(par=c(0,1,2), family="TF", type="left", varying=TRUE)
test7(3)


#------------------------------------------------------------------------------------------
# right
test8 <- trun.r(par=c(10,11,12), family="BCT", type="right", varying=TRUE)
test8(3)
#------------------------------------------------------------------------------------------
# both
test9<-trun.r(par=rbind(c(-3,3), c(-1,5), c(0,6)), , family="TF", type="both", varying=TRUE)
test9(3)
#------------------------------------------------------------------------------------------
# discrete
# trucated r function
# left
test10<-trun.r(par=c(0,1,2), family="PO", type="left", varying=TRUE)
```

```
test10(3)
#------------------------------------------------------------------------------------
# right
test11 <- trun.r(par=c(10,11,12), family="NBI", type="right", varying=TRUE)
test11(3)
test11(3, mu=10, sigma=.1)
#------------------------------------------------------------------------------------
# both
test12<-trun.r(par=rbind(c(0,10), c(1,11), c(2,12)), family="NBI", type="both", varying=TRUE)
test12(3,mu=5)
```

Index