Description
Beta regression for modeling beta-distributed dependent variables on the open unit interval (0, 1), e.g., rates and proportions, see Cribari-Neto and Zeileis (2010) <doi:10.18637/jss.v034.i02>. Moreover, extended-support beta regression models can accommodate dependent variables with boundary observations at 0 and/or 1. For the classical beta regression model, alternative specifications are provided: Bias-corrected and bias-reduced estimation, finite mixture models, and recursive partitioning for beta regression, see Grün, Kosmidis, and Zeileis (2012) <doi:10.18637/jss.v048.i11>.

Version 3.2-0
Date 2024-07-05
Title Beta Regression
Description
Beta regression for modeling beta-distributed dependent variables on the open unit interval (0, 1), e.g., rates and proportions, see Cribari-Neto and Zeileis (2010) <doi:10.18637/jss.v034.i02>. Moreover, extended-support beta regression models can accommodate dependent variables with boundary observations at 0 and/or 1. For the classical beta regression model, alternative specifications are provided: Bias-corrected and bias-reduced estimation, finite mixture models, and recursive partitioning for beta regression, see Grün, Kosmidis, and Zeileis (2012) <doi:10.18637/jss.v048.i11>.

Depends R (>= 3.6.0)
Imports graphics, grDevices, methods, stats, flexmix, Formula, lmtest, modeltools, sandwich
Suggests car, distributions3 (>= 0.2.1), lattice, numDeriv, partykit, statmod, strucchange
License GPL-2 | GPL-3
BugReports https://betareg.R-Forge.R-project.org/contact.html
Encoding UTF-8
NeedsCompilation yes
Author Achim Zeileis [aut, cre] (<https://orcid.org/0000-0003-0918-3766>), Francisco Cribari-Neto [aut] (<https://orcid.org/0000-0002-5909-6698>), Bettina Grün [aut] (<https://orcid.org/0000-0001-7265-4773>), Ioannis Kosmidis [aut] (<https://orcid.org/0000-0003-1556-0302>), Alexandre B. Simas [ctb] (earlier version by), Andrea V. Rocha [ctb] (earlier version by)
Maintainer Achim Zeileis <Achim.Zeileis@R-project.org>
Repository CRAN
Date/Publication 2024-07-07 10:20:03 UTC
Contents

Beta01 .......................................................... 2
beta01 .......................................................... 4
Beta4 .......................................................... 6
beta4 .......................................................... 7
betamix .......................................................... 9
BetaR .......................................................... 12
betar .......................................................... 14
betareg ......................................................... 15
betareg.control ............................................... 20
betatree ........................................................ 22
CarTask ......................................................... 24
FoodExpenditure .............................................. 25
GasolineYield .................................................. 26
gleverage ....................................................... 28
ImpreciseTask .................................................. 29
LossAversion ................................................... 30
MockJurors ..................................................... 32
plot.betareg ................................................... 34
predict.betareg ............................................... 35
ReadingSkills ................................................. 37
residuals.betareg ............................................. 39
StressAnxiety .................................................. 40
summary.betareg .............................................. 42
WeatherTask .................................................... 43
XBeta ........................................................... 44
xbeta ........................................................... 46
XBetaX .......................................................... 48
xbetax .......................................................... 50

Index .......................................................... 52

Beta01

Create a Zero- and/or One-Inflated Beta Distribution

Description

Class and methods for zero- and/or one-inflated beta distributions in regression specification using the workflow from the distributions3 package.

Usage

Beta01(mu, phi, p0 = 0, p1 = 0)
**Arguments**

- **mu** numeric. The mean of the beta distribution (on the open unit interval).
- **phi** numeric. The precision parameter of the beta distribution.
- **p0** numeric. The probability for an observation of zero (often referred to as zero inflation).
- **p1** numeric. The probability for an observation of one (often referred to as one inflation).

**Details**

The zero- and/or one-inflated beta distribution is obtained by adding point masses at zero and/or one to a standard beta distribution.

Note that the support of the standard beta distribution is the open unit interval where values of exactly zero or one cannot occur. Thus, the inflation jargon is rather misleading as there is no probability that could be inflated. It is rather a hurdle or two-part (or three-part) model.

**Value**

A `Beta01` distribution object.

**See Also**

`dbeta01`, `BetaR`

**Examples**

```r
## package and random seed
library("distributions3")
set.seed(6020)

## three beta distributions
X <- Beta01(
  mu = c(0.25, 0.50, 0.75),
  phi = c(1, 1, 2),
  p0 = c(0.1, 0, 0),
  p1 = c(0, 0, 0.3)
)

X

## compute moments of the distribution
mean(X)
variance(X)

## support interval (minimum and maximum)
support(X)

## simulate random variables
random(X, 5)
```
## histograms of 1,000 simulated observations
```r
x <- random(X, 1000)
hist(x[1, ])
hist(x[2, ])
hist(x[3, ])
```

## probability density function (PDF) and log-density (or log-likelihood)
```r
x <- c(0.25, 0.5, 0.75)
pdf(X, x)
pdf(X, x, log = TRUE)
log_pdf(X, x)
```

## cumulative distribution function (CDF)
```r
cdf(X, x)
```

## quantiles
```r
quantile(X, 0.5)
```

## cdf() and quantile() are inverses
```r
cdf(X, quantile(X, 0.5))
quantile(X, cdf(X, 1))
```

## point mass probabilities (if any) on boundary
```r
cdf(X, 0, lower.tail = TRUE)
cdf(X, 1, lower.tail = FALSE)
```

## all methods above can either be applied elementwise or for all combinations of X and x, if length(X) = length(x), also the result can be assured to be a matrix via drop = FALSE
```r
p <- c(0.05, 0.5, 0.95)
quantile(X, p, elementwise = FALSE)
quantile(X, p, elementwise = TRUE)
quantile(X, p, elementwise = TRUE, drop = FALSE)
```

## compare theoretical and empirical mean from 1,000 simulated observations
```r
cbind(
  "theoretical" = mean(X),
  "empirical" = rowMeans(random(X, 1000))
)
```

---

**beta01**  
*The Zero- and/or One-Inflated Beta Distribution in Regression Parameterization*

---

**Description**

Density, distribution function, quantile function, and random generation for the zero- and/or one-inflated beta distribution in regression parameterization.
Usage

\texttt{dbeta01(x, mu, phi, p0 = 0, p1 = 0, log = FALSE)}

\texttt{pbeta01(q, mu, phi, p0 = 0, p1 = 0, lower.tail = TRUE, log.p = FALSE)}

\texttt{qbeta01(p, mu, phi, p0 = 0, p1 = 0, lower.tail = TRUE, log.p = FALSE)}

\texttt{rbeta01(n, mu, phi, p0 = 0, p1 = 0)}

Arguments

- **x**, **q**: numeric. Vector of quantiles.
- **p**: numeric. Vector of probabilities.
- **n**: numeric. Number of observations. If \texttt{length(n) > 1}, the length is taken to be the number required.
- **mu**: numeric. The mean of the beta distribution (on the open unit interval).
- **phi**: numeric. The precision parameter of the beta distribution.
- **p0**: numeric. The probability for an observation of zero (often referred to as zero inflation).
- **p1**: numeric. The probability for an observation of one (often referred to as one inflation).
- **log**, **log.p**: logical. If TRUE, probabilities \( p \) are given as \( \log(p) \).
- **lower.tail**: logical. If TRUE (default), probabilities are \( P[X \leq x] \) otherwise, \( P[X > x] \).

Details

The zero- and/or one-inflated beta distribution is obtained by adding point masses at zero and/or one to a standard beta distribution.

Note that the support of the standard beta distribution is the open unit interval where values of exactly zero or one cannot occur. Thus, the inflation jargon is rather misleading as there is no probability that could be inflated. It is rather a hurdle or two-part (or three-part) model.

Value

\texttt{dbeta01} gives the density, \texttt{pbeta01} gives the distribution function, \texttt{qbeta01} gives the quantile function, and \texttt{rbeta01} generates random deviates.

See Also

\texttt{dbetar}, \texttt{Beta01}
Create a 4-Parameter Beta Distribution

Description

Class and methods for 4-parameter beta distributions in regression specification using the workflow from the distributions3 package.

Usage

```r
Beta4(mu, phi, theta1 = 0, theta2 = 1 - theta1)
```

Arguments

- **mu**: numeric. The mean of the beta distribution that is extended to support \([\theta_1, \theta_2]\).
- **phi**: numeric. The precision parameter of the beta distribution that is extended to support \([\theta_1, \theta_2]\).
- **theta1, theta2**: numeric. The minimum and maximum, respectively, of the 4-parameter beta distribution. By default a symmetric support is chosen by \(\theta_2 = 1 - \theta_1\) which reduces to the classic beta distribution because of the default \(\theta_1 = 0\).

Details

The distribution is obtained by a linear transformation of a beta-distributed random variable with intercept \(\theta_1\) and slope \(\theta_2 - \theta_1\).

Value

A Beta4 distribution object.

See Also

dbeta4, BetaR

Examples

```r
## package and random seed
library("distributions3")
set.seed(6020)

## three beta distributions
X <- Beta4(
  mu = c(0.25, 0.50, 0.75),
  phi = c(1, 1, 2),
  theta1 = c(0, -0.1, -0.1),
  theta2 = c(1, 1.1, 1.5)
)
```
X

## compute moments of the distribution
mean(X)
variance(X)

## support interval (minimum and maximum)
support(X)

## simulate random variables
random(X, 5)

## histograms of 1,000 simulated observations
x <- random(X, 1000)
hist(x[1, ])
hist(x[2, ])
hist(x[3, ])

## probability density function (PDF) and log-density (or log-likelihood)
x <- c(0.25, 0.5, 0.75)
pdf(X, x)
pdf(X, x, log = TRUE)
log_pdf(X, x)

## cumulative distribution function (CDF)
cdf(X, x)

## quantiles
quantile(X, 0.5)

## cdf() and quantile() are inverses
cdf(X, quantile(X, 0.5))
quantile(X, cdf(X, 1))

## all methods above can either be applied elementwise or for
## all combinations of X and x, if length(X) = length(x),
## also the result can be assured to be a matrix via drop = FALSE
p <- c(0.05, 0.5, 0.95)
quantile(X, p, elementwise = FALSE)
quantile(X, p, elementwise = TRUE)
quantile(X, p, elementwise = TRUE, drop = FALSE)

## compare theoretical and empirical mean from 1,000 simulated observations
cbind(
    "theoretical" = mean(X),
    "empirical" = rowMeans(random(X, 1000))
)
Description

Density, distribution function, quantile function, and random generation for the 4-parameter beta distribution in regression parameterization.

Usage

dbeta4(x, mu, phi, theta1 = 0, theta2 = 1 - theta1, log = FALSE)
pbeta4(q, mu, phi, theta1 = 0, theta2 = 1 - theta1, lower.tail = TRUE, log.p = FALSE)
qbeta4(p, mu, phi, theta1 = 0, theta2 = 1 - theta1, lower.tail = TRUE, log.p = FALSE)
rbeta4(n, mu, phi, theta1 = 0, theta2 = 1 - theta1)

Arguments

x, q numeric. Vector of quantiles.
p numeric. Vector of probabilities.
n numeric. Number of observations. If length(n) > 1, the length is taken to be the number required.
mu numeric. The mean of the beta distribution that is extended to support [theta1, theta2].
phi numeric. The precision parameter of the beta distribution that is extended to support [theta1, theta2].
theta1, theta2 numeric. The minimum and maximum, respectively, of the 4-parameter beta distribution. By default a symmetric support is chosen by theta2 = 1 - theta1 which reduces to the classic beta distribution because of the default theta1 = 0.
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].

Details

The distribution is obtained by a linear transformation of a beta-distributed random variable with intercept theta1 and slope theta2 - theta1.

Value

dbeta4 gives the density, pbeta4 gives the distribution function, qbeta4 gives the quantile function, and rbeta4 generates random deviates.

See Also
dbetar, Beta4
**Description**

Fit finite mixtures of beta regression models for rates and proportions via maximum likelihood with the EM algorithm using a parametrization with mean (depending through a link function on the covariates) and precision parameter (called phi).

**Usage**

```r
betamix(formula, data, k, subset, na.action, weights, offset,
link = c("logit", "probit", "cloglog", "cauchit", "log", "loglog"),
link.phi = "log",
control = betareg.control(...), cluster = NULL,
FLXconcomitant = NULL, FLXcontrol = list(), verbose = FALSE,
nstart = if (is.null(cluster)) 3 else 1, which = "BIC",
ID, fixed, extra_components, ...)
```

```r
extraComponent(type = c("uniform", "betareg"), coef, delta,
link = "logit", link.phi = "log")
```

**Arguments**

- `formula` symbolic description of the model (of type `y ~ x` or `y ~ x | z`; for details see `betareg`).
- `data, subset, na.action` arguments controlling formula processing via `model.frame`.
- `weights` optional numeric vector of integer case weights.
- `offset` optional numeric vector with an a priori known component to be included in the linear predictor for the mean.
- `k` a vector of integers indicating the number of components of the finite mixture; passed in turn to the `k` argument of `stepFlexmix`.
- `link` character specification of the link function in the mean model (mu). Currently, "logit", "probit", "cloglog", "cauchit", "log", "loglog" are supported. Alternatively, an object of class "link-glm" can be supplied.
- `link.phi` character specification of the link function in the precision model (phi). Currently, "identity", "log", "sqrt" are supported. The default is "log" unless formula is of type `y ~ x` where the default is "identity" (for backward compatibility). Alternatively, an object of class "link-glm" can be supplied.
- `control` a list of control arguments specified via `betareg.control`.
- `cluster` Either a matrix with `k` columns of initial cluster membership probabilities for each observation; or a factor or integer vector with the initial cluster assignments of observations at the start of the EM algorithm. Default is random assignment into `k` clusters.
FLXconcomitant concomitant variable model; object of class FLXP. Default is the object returned by calling FLXPconstant. The argument FLXconcomitant can be omitted if formula is a three-part formula of type \( y \sim x \mid z \mid w \), where \( w \) specifies the concomitant variables.

FLXcontrol object of class "FLXcontrol" or a named list; controls the EM algorithm and passed in turn to the control argument of flexmix.

verbose a logical; if TRUE progress information is shown for different starts of the EM algorithm.

nstart for each value of \( k \) run stepFlexmix \( n_{\text{start}} \) times and keep only the solution with maximum likelihood.

which number of model to get if \( k \) is a vector of integers longer than one. If character, interpreted as number of components or name of an information criterion.

ID grouping variable indicating if observations are from the same individual, i.e. the component membership is restricted to be the same for these observations.

fixed symbolic description of the model for the parameters fixed over components (of type \( \sim x \mid z \)).

extra_components a list containing objects returned by extraComponent().

... arguments passed to betareg.control.

type specifies if the component follows a uniform distribution or a beta regression model.

coef a vector with the coefficients to determine the midpoint of the uniform distribution or names list with the coefficients for the mean and precision of the beta regression model.

delta numeric; half-length of the interval of the uniform distribution.

Details

The arguments and the model specification are similar to betareg. Internally stepFlexmix is called with suitable arguments to fit the finite mixture model with the EM algorithm. See Grün et al. (2012) for more details.

extra_components is a list where each element corresponds to a component where the parameters are fixed a-priori.

Value

An object of class "flexmix" containing the best model with respect to the log likelihood or the one selected according to which if \( k \) is a vector of integers longer than 1.

Author(s)

Bettina Grün and Achim Zeileis
References


See Also

`betareg`, `flexmix`, `stepFlexmix`

Examples

```r
options(digits = 4)

## data with two groups of dyslexic and non-dyslexic children
data("ReadingSkills", package = "betareg")

suppressWarnings(RNGversion("3.5.0"))
set.seed(4040)
## try to capture accuracy ~ iq relationship (without using dyslexia
## information) using two beta regression components and one additional
## extra component for a perfect reading score
rs_mix <- betamix(accuracy ~ iq, data = ReadingSkills, k = 3,
nstart = 10, extra_components = extraComponent(type = "uniform",
    coef = 0.99, delta = 0.01))

## visualize result
## intensities based on posterior probabilities
prob <- 2 * (posterior(rs_mix)[cbind(1:nrow(ReadingSkills),
    clusters(rs_mix))] - 0.5)
## associated HCL colors
col0 <- hcl(c(260, 0, 130), 65, 45, fixup = FALSE)
col1 <- col0[clusters(rs_mix)]
col2 <- hcl(c(260, 0, 130)[clusters(rs_mix)], 65 * abs(prob)^1.5,
    95 - 50 * abs(prob)^1.5, fixup = FALSE)
## scatter plot
plot(accuracy ~ iq, data = ReadingSkills, col = col2, pch = 19,
cex = 1.5, xlim = c(-2, 2))
points(accuracy ~ iq, data = ReadingSkills, cex = 1.5, pch = 1,
col = col1)
## fitted lines
iq <- -30:30/10
cf <- rbind(coef(rs_mix, model = "mean", component = 1:2),
c(qlogis(0.99), 0))
for(i in 1:3)
```
lines(iq, plogis(cf[i, 1] + cf[i, 2] * iq), lwd = 2, col = col0[i])

## refit the model including a concomitant variable model using the
dyslexia information with some noise to avoid complete separation
## between concomitant variable and component memberships
set.seed(4040)
w <- rnorm(nrow(ReadingSkills),
    c(-1, 1)[as.integer(ReadingSkills$dyslexia)])

## The argument FLXconcomitant can be omitted when specifying
## the model via a three part formula given by
## accuracy ~ iq | 1 | w
## The posteriors from the previously fitted model are used
## for initialization.
library("flexmix")
rs_mix2 <- betamix(accuracy ~ iq, data = ReadingSkills,
    extra_components = extraComponent(type = "uniform",
        coef = 0.99, delta = 0.01), cluster = posterior(rs_mix),
    FLXconcomitant = FLXPmultinom(~w))
coef(rs_mix2, which = "concomitant")
summary(rs_mix2, which = "concomitant")

BetaR

Create a Beta Regression Distribution

Description

Class and methods for beta distributions in regression specification using the workflow from the
distributions3 package.

Usage

BetaR(mu, phi)

Arguments

mu numeric. The mean of the beta distribution.
phi numeric. The precision parameter of the beta distribution.

Details

Alternative parameterization of the classic beta distribution in terms of its mean mu and precision
parameter phi. Thus, the distribution provided by BetaR is equivalent to the Beta distribution with
parameters alpha = mu * phi and beta = (1 - mu) * phi.

Value

A BetaR distribution object.
See Also

dbetar, Beta

Examples

```r
## package and random seed
library("distributions3")
set.seed(6020)

## three beta distributions
X <- BetaR(
  mu = c(0.25, 0.50, 0.75),
  phi = c(1, 1, 2)
)
X

## compute moments of the distribution
mean(X)
variance(X)
skewness(X)
kurtosis(X)

## support interval (minimum and maximum)
support(X)

## simulate random variables
random(X, 5)

## histograms of 1,000 simulated observations
x <- random(X, 1000)
hist(x[1, ])
hist(x[2, ])
hist(x[3, ])

## probability density function (PDF) and log-density (or log-likelihood)
x <- c(0.25, 0.5, 0.75)
pdf(X, x)
pdf(X, x, log = TRUE)
log_pdf(X, x)

## cumulative distribution function (CDF)
cdf(X, x)

## quantiles
quantile(X, 0.5)

## cdf() and quantile() are inverses (except at censoring points)
cdf(X, quantile(X, 0.5))
quantile(X, cdf(X, 1))

## all methods above can either be applied elementwise or for
```
## all combinations of X and x, if length(X) = length(x),
## also the result can be assured to be a matrix via drop = FALSE
p <- c(0.05, 0.5, 0.95)
quantile(X, p, elementwise = FALSE)
quantile(X, p, elementwise = TRUE)
quantile(X, p, elementwise = TRUE, drop = FALSE)

## compare theoretical and empirical mean from 1,000 simulated observations
cbind(
  "theoretical" = mean(X),
  "empirical" = rowMeans(random(X, 1000))
)

---

betar

The Beta Distribution in Regression Parameterization

Description

Density, distribution function, quantile function, and random generation for the beta distribution in regression parameterization.

Usage

dbetar(x, mu, phi, log = FALSE)
pbetar(q, mu, phi, lower.tail = TRUE, log.p = FALSE)
qbetar(p, mu, phi, lower.tail = TRUE, log.p = FALSE)
rbetar(n, mu, phi)

Arguments

- x, q: numeric. Vector of quantiles.
- n: numeric. Number of observations. If length(n) > 1, the length is taken to be the number required.
- mu: numeric. The mean of the beta distribution.
- phi: numeric. The precision parameter of the beta distribution.
- 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].

Details

This is the reparameterization of the beta distribution with mean mu and precision phi, as employed in beta regression. The classic parameterization of the beta distribution is obtained by setting shape1 = mu * phi and shape2 = (1 - mu) * phi, respectively.
Value
dbetal gives the density, pbetal gives the distribution function, qbetal gives the quantile function, and rbetal generates random deviates.

See Also
dbeta, BetaR

betareg

Beta Regression for Rates and Proportions

Description
Fit beta regression models for rates and proportions via maximum likelihood using a parametrization with mean (depending through a link function on the covariates) and precision parameter (called phi).

Usage
betareg(formula, data, subset, na.action, weights, offset, link = c("logit", "probit", "cloglog", "cauchit", "log", "loglog"), link.phi = NULL, type = c("ML", "BC", "BR"), dist = NULL, nu = NULL, control = betareg.control(...), model = TRUE, y = TRUE, x = FALSE, ...)
betareg.fit(x, y, z = NULL, weights = NULL, offset = NULL, link = "logit", link.phi = "log", type = "ML", control = betareg.control(), dist = NULL, nu = NULL)

Arguments
formula symbolic description of the model, either of type y ~ x (mean submodel, constant precision) or y ~ x | z (submodels for both mean and precision); for details see below.
data, subset, na.action arguments controlling formula processing via model.frame.
weights optional numeric vector of case weights.
offset optional numeric vector with an a priori known component to be included in the linear predictor for the mean. In betareg.fit, offset may also be a list of two offsets for the mean and precision equation, respectively.
link character specification of the link function in the mean model (mu). Currently, "logit", "probit", "cloglog", "cauchit", "log", "loglog" are supported. Alternatively, an object of class "link-glm" can be supplied.
link.phi character specification of the link function in the precision model (phi). Currently, "identity", "log", "sqrt" are supported. The default is "log" unless formula is of type y ~ x where the default is "identity" (for backward compatibility). Alternatively, an object of class "link-glm" can be supplied.
**type** character specification of the type of estimator. Currently, maximum likelihood ("ML"), ML with bias correction ("BC"), and ML with bias reduction ("BR") are supported.

**dist** character specification of the response distribution. Usually, this does not have to be set by the user because by default the classical "beta" distribution is used when all observations for the dependent variable are in (0, 1). In the presence of boundary observations (0 or 1, which cannot be accommodated by "beta") the extended-support beta mixture distribution ("xbeta") is used. Additionally, dist = "xbeta" can be used with fixed exceedence parameter nu, mostly for testing and debugging purposes.

**nu** numeric. The fixed value of the expected exceedence parameter nu in case the extended-support beta mixture distribution is used. By default, nu does not need to be specified and is estimated if needed. So setting nu is mostly for profiling and debugging.

**control** a list of control arguments specified via `betareg.control`.

**model, y, x** logicals. If TRUE the corresponding components of the fit (model frame, response, model matrix) are returned. For `betareg.fit`, x should be a numeric regressor matrix and y should be the numeric response vector (with values in (0,1)).

**z** numeric matrix. Regressor matrix for the precision model, defaulting to an intercept only.

... arguments passed to `betareg.control`.

**Details**

Beta regression as suggested by Ferrari and Cribari-Neto (2004) and extended by Simas, Barreto-Souza, and Rocha (2010) is implemented in betareg. It is useful in situations where the dependent variable is continuous and restricted to the unit interval (0, 1), e.g., resulting from rates or proportions. It is modeled to be beta-distributed with parametrization using mean and precision parameter (called mu and phi, respectively). The mean mu is linked, as in generalized linear models (GLMs), to the explanatory variables through a link function and a linear predictor. Additionally, the precision parameter phi can be linked to another (potentially overlapping) set of regressors through a second link function, resulting in a model with variable dispersion (see Cribari-Neto and Zeileis 2010). Estimation is performed by default using maximum likelihood (ML) via `optim` with analytical gradients and starting values from an auxiliary linear regression of the transformed response. Subsequently, the `optim` result may be enhanced by an additional Fisher scoring iteration using analytical gradients and expected information. Alternative estimation methods are bias-corrected (BC) or bias-reduced (BR) maximum likelihood (see Grün, Kosmidis, and Zeileis 2012). For ML and BC the Fisher scoring is just a refinement to move the gradients even closer to zero and can be disabled by setting `fsmaxit = 0` in the control arguments. For BR the Fisher scoring is needed to solve the bias-adjusted estimating equations.

In the beta regression as introduced by Ferrari and Cribari-Neto (2004), the mean of the response is linked to a linear predictor described by \( y \sim x_1 + x_2 \) using a link function while the precision parameter phi is assumed to be constant. Simas et al. (2009) suggest to extend this model by linking phi to an additional set of regressors \( (z_1 + z_2, \text{say}) \). In `betareg` this can be specified in a formula of type \( y \sim x_1 + x_2 | z_1 + z_2 \) where the regressors in the two parts can be overlapping. In the precision model (for phi), the link function `link.phi` is used. The default is a "log" link unless no precision
model is specified. In the latter case (i.e., when the formula is of type \( y \sim x1 + x2 \)), the "identity" link is used by default for backward compatibility.

Kosmidis and Zeileis (2024) introduce a generalization of the classic beta regression model with extended support \([0, 1]\). Specifically, the extended-support beta distribution ("xbeta") leverages an underlying symmetric four-parameter beta distribution with exceedence parameter \( \nu \) to obtain support \([-\nu, 1 + \nu]\) that is subsequently censored to \([0, 1]\) in order to obtain point masses at the boundary values 0 and 1. The extended-support beta mixture distribution ("xbetax") is a continuous mixture of extended-support beta distributions where the exceedence parameter follows an exponential distribution with mean \( \nu \) (rather than a fixed value of \( \nu \)). The latter "xbetax" specification is used by default in case of boundary observations at 0 and/or 1. The "xbeta" specification with fixed \( \nu \) is mostly for testing and debugging purposes.

A set of standard extractor functions for fitted model objects is available for objects of class "betareg", including methods to the generic functions `print`, `summary`, `plot`, `coef`, `vcov`, `logLik`, `residuals`, `predict`, `terms`, `model.frame`, `model.matrix`, `cooks.distance` and `hatvalues` (see `influence.measures`), `gleverage` (new generic), `estfun` and `bread` (from the `sandwich` package), and `coeftest` (from the `lmtest` package).

See `predict.betareg`, `residuals.betareg`, `plot.betareg`, and `summary.betareg` for more details on all methods.

The main parameters of interest are the coefficients in the linear predictor of the mean model. The additional parameters in the precision model (\( \phi \)) can either be treated as full model parameters (default) or as nuisance parameters. In the latter case the estimation does not change, only the reported information in output from `print`, `summary`, or `coef` (among others) will be different. See also `betareg.control`.

The implemented algorithms for bias correction/reduction follow Kosmidis and Firth (2010). Technical note: In case either bias correction or reduction is requested, the second derivative of the inverse link function is required for `link` and `link.phi`. If the two links are specified by their names (as done by default in `betareg`), then the "link-glm" objects are enhanced automatically by the required additional `d2mu.deta` function. However, if a "link-glm" object is supplied directly by the user, it needs to have the `d2mu.deta` function or, for backward compatibility, `dmu.deta`.

The original version of the package was written by Alexandre B. Simas and Andrea V. Rocha (up to version 1.2). Starting from version 2.0-0 the code was rewritten by Achim Zeileis.

**Value**

`betareg` returns an object of class "betareg", i.e., a list with components as follows. For classic beta regressions (\( \text{dist} = \text{"beta"} \)) several elements are lists with the names "mean" and "precision" for the information from the respective submodels. For extended-support beta regressions (\( \text{dist} = \text{"xbetax"} \) or \( \text{"xbeta"} \)), the corresponding names are "mu" and "phi" because they are not exactly the mean and precision due to the censoring in the response variable.

`betareg.fit` returns an unclassed list with components up to converged.

- **coefficients**: a list with elements "mean" (or "mu") and "precision" (or "phi") containing the coefficients from the respective submodels and for extended-support beta regressions an additional element "nu",
- **residuals**: a vector of raw residuals (observed - fitted),
- **fitted.values**: a vector of fitted means,
output from the \texttt{optim} call for maximizing the log-likelihood(s),
the method argument passed to the \texttt{optim} call,
the control arguments passed to the \texttt{optim} call,
the starting values for the parameters passed to the \texttt{optim} call,
the weights used (if any),
a list of offset vectors used (if any),
number of observations,
number of observations with non-zero weights,
residual degrees of freedom in the null model (constant mean and dispersion),
i.e., \( n - 2 \),
residual degrees of freedom in the fitted model,
logical indicating whether the precision (\( \phi \)) coefficients will be treated as full model parameters or nuisance parameters in subsequent calls to \texttt{print}, \texttt{summary}, \texttt{coef} etc.,
log-likelihood of the fitted model,
covariance matrix of all parameters in the model,
pseudo R-squared value (squared correlation of linear predictor and link-transformed response),
a list with elements "mean" (or "\( \mu \)") and "precision" (or "\( \phi \)"") containing the link objects for the respective submodels,
logical indicating successful convergence of \texttt{optim},
the original function call,
the original formula,
a list with elements "mean" (or "\( \mu \)"), "precision" (or "\( \phi \)"), and "full" containing the terms objects for the respective models,
a list with elements "mean" (or "\( \mu \)"), "precision" (or "\( \phi \)"), and "full" containing the levels of the categorical regressors,
a list with elements "mean" (or "\( \mu \)"), "precision" (or "\( \phi \)"") containing the contrasts corresponding to \texttt{levels} from the respective models,
the full model frame (if \texttt{model} = \texttt{TRUE}),
the response proportion vector (if \texttt{y} = \texttt{TRUE}),
a list with elements "mean" (or "\( \mu \)"), "precision" (or "\( \phi \)"") containing the model matrices from the respective models (if \texttt{x} = \texttt{TRUE}).

\textbf{References}


### See Also

*summary.betareg*, *predict.betareg*, *residuals.betareg*, *Formula*

### Examples

```r
options(digits = 4)

## Section 4 from Ferrari and Cribari-Neto (2004)
data("GasolineYield", package = "betareg")
data("FoodExpenditure", package = "betareg")

## Table 1
gy <- betareg(yield ~ batch + temp, data = GasolineYield)
summary(gy)

## Table 2
fe_lin <- lm(I(food/income) ~ income + persons, data = FoodExpenditure)
library("lmtest")
bptest(fe_lin)
fe_beta <- betareg(I(food/income) ~ income + persons, data = FoodExpenditure)
summary(fe_beta)

## nested model comparisons via Wald and LR tests
fe_beta2 <- betareg(I(food/income) ~ income, data = FoodExpenditure)
lrtest(fe_beta, fe_beta2)
waldtest(fe_beta, fe_beta2)

## Section 3 from online supplements to Simas et al. (2010)
## mean model as in gy above
## precision model with regressor temp
gy2 <- betareg(yield ~ batch + temp | temp, data = GasolineYield)

## MLE column in Table 19
summary(gy2)

## LRT row in Table 18
lrtest(gy, gy2)
```
betareg.control  Control Parameters for Beta Regression

Description

Various parameters that control fitting of beta regression models using betareg.

Usage

betareg.control(phi = TRUE, method = "BFGS", maxit = 5000,
    gradient = NULL, hessian = FALSE, trace = FALSE, start = NULL,
    fsmaxit = 200, fstol = 1e-8, quad = 20, ...)

Arguments

phi  logical indicating whether the precision parameter phi should be treated as a full model parameter (TRUE, default) or as a nuisance parameter.

method  characters string specifying the method argument passed to optim. Additionally, method = "nlminb" can be used to employ nlminb, instead.

maxit  integer specifying the maxit argument (maximal number of iterations) passed to optim.

trace  logical or integer controlling whether tracing information on the progress of the optimization should be produced (passed to optim).

gradient  logical. Should the analytical gradient be used for optimizing the log-likelihood? If set to FALSE a finite-difference approximation is used instead. The default of NULL signals that analytical gradients are only used for the classical "beta" distribution but not for "xbetax" or "xbeta".

hessian  logical. Should the numerical Hessian matrix from the optim output be used for estimation of the covariance matrix? By default the analytical solution is employed. For details see below.

start  an optional vector with starting values for all parameters (including phi).

fsmaxit  integer specifying maximal number of additional (quasi) Fisher scoring iterations. For details see below.

fstol  numeric tolerance for convergence in (quasi) Fisher scoring. For details see below.

quad  numeric. The number of quadrature points for numeric integration in case of dist = "xbetax" is used in the beta regression.

... arguments passed to optim.
Details

All parameters in `betareg` are estimated by maximum likelihood using `optim` with control options set in `betareg.control`. Most arguments are passed on directly to `optim`, and `start` controls how `optim` is called.

After the `optim` maximization, an additional (quasi) Fisher scoring can be performed to further enhance the result or to perform additional bias reduction. If `fsmaxit` is greater than zero, this additional optimization is performed and it converges if the threshold `fstol` is attained for the cross-product of the step size.

Starting values can be supplied via `start` or estimated by `lm.wfit`, using the link-transformed response. Covariances are in general derived analytically. Only if `type = "ML"` and `hessian = TRUE`, they are determined numerically using the Hessian matrix returned by `optim`. In the latter case no Fisher scoring iterations are performed.

The main parameters of interest are the coefficients in the linear predictor of the model and the additional precision parameter `phi` which can either be treated as a full model parameter (default) or as a nuisance parameter. In the latter case the estimation does not change, only the reported information in output from `print`, `summary`, or `coef` (among others) will be different. See also examples.

Value

A list with the arguments specified.

See Also

`betareg`

Examples

```r
options(digits = 4)
data("GasolineYield", package = "betareg")

## regression with phi as full model parameter
gy1 <- betareg(yield ~ batch + temp, data = GasolineYield)
gy1

## regression with phi as nuisance parameter
gy2 <- betareg(yield ~ batch + temp, data = GasolineYield, phi = FALSE)
gy2

## compare reported output
coeff(gy1)
coeff(gy2)
summary(gy1)
summary(gy2)
```
Beta Regression Trees

Description

Fit beta regression trees via model-based recursive partitioning.

Usage

betatree(formula, partition, data, subset = NULL, na.action = na.omit, weights, offset, cluster, link = "logit", link.phi = "log", control = betareg.control(), ...)

Arguments

formula symbolic description of the model of type \( y \sim x \) or \( y \sim x \mid z \), specifying the variables influencing mean and precision of \( y \), respectively. For details see betareg.

partition symbolic description of the partitioning variables, e.g., \( \sim p_1 + p_2 \). The argument partition can be omitted if formula is a three-part formula of type \( y \sim x \mid z \mid p_1 + p_2 \).

data, subset, na.action, weights, offset, cluster arguments controlling data/model processing passed to mob.

link character specification of the link function in the mean model (mu). Currently, "logit", "probit", "cloglog", "cauchit", "log", "loglog" are supported. Alternatively, an object of class "link-glm" can be supplied.

link.phi character specification of the link function in the precision model (phi). Currently, "identity", "log", "sqrt" are supported. Alternatively, an object of class "link-glm" can be supplied.

control a list of control arguments for the beta regression specified via betareg.control.

Details

Beta regression trees are an application of model-based recursive partitioning (implemented in mob, see Zeileis et al. 2008) to beta regression (implemented in betareg, see Cribari-Neto and Zeileis 2010). See also Grün at al. (2012) for more details.

Various methods are provided for "betatree" objects, most of them inherit their behavior from "mob" objects (e.g., print, summary, coef, etc.). The plot method employs the node_bivplot panel-generating function.

Value

betatree() returns an object of S3 class "betatree" which inherits from "modelparty".
References


See Also

`betareg`, `betareg.fit`, `mob`

Examples

```r
options(digits = 4)
suppressWarnings(RNGversion("3.5.0"))

## data with two groups of dyslexic and non-dyslexic children
data("ReadingSkills", package = "betareg")
## additional random noise (not associated with reading scores)
set.seed(1071)
ReadingSkills$x1 <- rnorm(nrow(ReadingSkills))
ReadingSkills$x2 <- runif(nrow(ReadingSkills))
ReadingSkills$x3 <- factor(rnorm(nrow(ReadingSkills)) > 0)

## fit beta regression tree: in each node
## - accuracy's mean and precision depends on iq
## - partitioning is done by dyslexia and the noise variables x1, x2, x3
## only dyslexia is correctly selected for splitting
bt <- betatree(accuracy ~ iq | iq, ~ dyslexia + x1 + x2 + x3,
data = ReadingSkills, minsize = 10)
plot(bt)

## inspect result
coef(bt)
if(require("strucchange")) sctest(bt)
## IGNORE_RDIFF_BEGIN
summary(bt, node = 2)
summary(bt, node = 3)
## IGNORE_RDIFF_END

## add a numerical variable with relevant information for splitting
ReadingSkills$x4 <- rnorm(nrow(ReadingSkills), c(-1.5, 1.5)[ReadingSkills$dyslexia])

bt2 <- betatree(accuracy ~ iq | iq, ~ x1 + x2 + x3 + x4,
data = ReadingSkills, minsize = 10)
plot(bt2)

## inspect result
coef(bt2)
if(require("strucchange")) sctest(bt2)
```
## IGNORE_RDIFF_BEGIN

```r
summary(bt2, node = 2)
summary(bt2, node = 3)
```

## IGNORE_RDIFF_END

<table>
<thead>
<tr>
<th>CarTask</th>
<th>Partition-primed Probability Judgement Task for Car Dealership</th>
</tr>
</thead>
</table>

### Description

In this study participants were asked to judge how likely it is that a customer trades in a coupe or that a customer buys a car from a specific salesperson out of four possible salespersons.

### Usage

```r
data("CarTask", package = "betareg")
```

### Format

A data frame with 155 observations on the following 3 variables.

- **task** a factor with levels Car and Salesperson indicating the condition.
- **probability** a numeric vector of the estimated probability.
- **NFCCscale** a numeric vector of the NFCC scale.

### Details

All participants in the study were undergraduate students at The Australian National University, some of whom obtained course credit in first-year Psychology for their participation in the study.

The NFCC scale is a combined scale of the Need for Closure and Need for Certainty scales which are strongly correlated.

For the task questions were:

**Car** What is the probability that a customer trades in a coupe?

**Salesperson** What is the probability that a customer buys a car from Carlos?

### Source

Taken from Smithson et al. (2011) supplements.

### References


Examples

```r
data("CarTask", package = "betareg")
library("flexmix")
car_betamix <- betamix(probability ~ 1, data = CarTask, k = 3,
extra_components = list(extraComponent(type = "uniform", coef = 1/2,
delta = 0.01), extraComponent(type = "uniform", coef = 1/4, delta = 0.01)),
FLXconcomitant = FLXPmultinom(~ task))
```

Description

Data on proportion of income spent on food for a random sample of 38 households in a large US city.

Usage

```r
data("FoodExpenditure", package = "betareg")
```

Format

A data frame containing 38 observations on 3 variables.

- **food**: household expenditures for food.
- **income**: household income.
- **persons**: number of persons living in household.

Source

Taken from Griffiths et al. (1993, Table 15.4).

References


See Also

- `betareg`
Examples

```r
data("FoodExpenditure", package = "betareg")

## Section 4
fe_lin <- lm(I(food/income) ~ income + persons, data = FoodExpenditure)
library("lmtest")
bptest(fe_lin)

## Table 2
fe_beta <- betareg(I(food/income) ~ income + persons, data = FoodExpenditure)
summary(fe_beta)
```

---

**GasolineYield**

*Estimation of Gasoline Yields from Crude Oil*

**Description**

Operational data of the proportion of crude oil converted to gasoline after distillation and fractionation.

**Usage**

```r
data("GasolineYield", package = "betareg")
```

**Format**

A data frame containing 32 observations on 6 variables.

- **yield**: proportion of crude oil converted to gasoline after distillation and fractionation.
- **gravity**: crude oil gravity (degrees API).
- **pressure**: vapor pressure of crude oil (lbf/in2).
- **temp10**: temperature (degrees F) at which 10 percent of crude oil has vaporized.
- **temp**: temperature (degrees F) at which all gasoline has vaporized.
- **batch**: factor indicating unique batch of conditions gravity, pressure, and temp10.

**Details**

This dataset was collected by Prater (1956), its dependent variable is the proportion of crude oil after distillation and fractionation. This dataset was analyzed by Atkinson (1985), who used the linear regression model and noted that there is “indication that the error distribution is not quite symmetrical, giving rise to some unduly large and small residuals” (p. 60).

The dataset contains 32 observations on the response and on the independent variables. It has been noted (Daniel and Wood, 1971, Chapter 8) that there are only ten sets of values of the first three explanatory variables which correspond to ten different crudes and were subjected to experimentally controlled distillation conditions. These conditions are captured in variable batch and the data were ordered according to the ascending order of temp10.
Source

Taken from Prater (1956).

References


See Also

`betareg`

Examples

```r
## IGNORE_RDIFF_BEGIN
data("GasolineYield", package = "betareg")

gy1 <- betareg(yield ~ gravity + pressure + temp10 + temp, data = GasolineYield)
summary(gy1)

gy2 <- betareg(yield ~ batch + temp, data = GasolineYield)
## Table 1
summary(gy2)
## Figure 2
par(mfrow = c(3, 2))
plot(gy2, which = 1, type = "pearson", sub.caption = "")
plot(gy2, which = 1, type = "deviance", sub.caption = "")
plot(gy2, which = 5, type = "deviance", sub.caption = "")
plot(gy2, which = 4, type = "pearson", sub.caption = "")
plot(gy2, which = 2:3)
par(mfrow = c(1, 1))

## exclude 4th observation
gy2a <- update(gy2, subset = -4)
gy2a
summary(gy2a)
## IGNORE_RDIFF_END
```
**gleverage**

*Generalized Leverage Values*

**Description**

Compute the generalized leverages values for fitted models.

**Usage**

```r
gleverage(model, ...) 
```

**Arguments**

- `model`: a model object.
- `...`: further arguments passed to methods.

**Value**

`gleverage` is a new generic for computing generalized leverage values as suggested by Wei, Hu, and Fung (1998). Currently, there is only a method for `betareg` models, implementing the formulas from Rocha and Simas (2011) which are consistent with the formulas from Ferrari and Cribari-Neto (2004) for the fixed dispersion case.

Currently, the vector of generalized leverages requires computations and storage of order $n \times n$.

**References**


**See Also**

- `betareg`

**Examples**

```r
options(digits = 4)
data("GasolineYield", package = "betareg")
gy <- betareg(yield ~ batch + temp, data = GasolineYield)
gleverage(gy)
```
### Description

In this study participants were asked to estimate upper and lower probabilities for event to occur and not to occur.

### Usage

```r
data("ImpreciseTask", package = "betareg")
```

### Format

A data frame with 242 observations on the following 3 variables.

- **task**: a factor with levels Boeing stock and Sunday weather.
- **location**: a numeric vector of the average of the lower estimate for the event not to occur and the upper estimate for the event to occur.
- **difference**: a numeric vector of the differences of the lower and upper estimate for the event to occur.

### Details

All participants in the study were either first- or second-year undergraduate students in psychology, none of whom had a strong background in probability or were familiar with imprecise probability theories.

For the Sunday weather task see `WeatherTask`. For the Boeing stock task participants were asked to estimate the probability that Boeing’s stock would rise more than those in a list of 30 companies.

For each task participants were asked to provide lower and upper estimates for the event to occur and not to occur.

### Source

Taken from Smithson et al. (2011) supplements.

### References


Examples

data("ImpreciseTask", package = "betareg")
library("flexmix")
w_t_betamix <- betamix(location ~ difference * task, data = ImpreciseTask, k = 2,
extra_components = extraComponent(type = "betareg", coef =
list(mean = 0, precision = 8)),
FLXconcomitant = FLXPmultinom(~ task))

---

LossAversion

(No) Myopic Loss Aversion in Adolescents

Description

Data for assessing the extent of myopic loss aversion among adolescents (mostly aged 11 to 19).

Usage

data("LossAversion", package = "betareg")

Format

A data frame containing 570 observations on 7 variables.

invest numeric. Average proportion of points invested across all 9 rounds.

gender factor. Gender of the player (or team of players).

male factor. Was (at least one of) the player(s) male (in the team)?

age numeric. Age in years (averaged for teams).

treatment factor. Type of treatment: long vs. short.

grade factor. School grades: 6-8 (11-14 years) vs. 10-12 (15-18 years).

arrangement factor. Is the player a single player or team of two?

Details

Myopic loss aversion is a phenomenon in behavioral economics, where individuals do not behave economically rationally when making short-term decisions under uncertainty. Example: In lotteries with positive expected payouts investments are lower than the maximum possible (loss aversion). This effect is enhanced for short-term investments (myopia or short-mindedness).

The data in LossAversion were collected by Matthias Sutter and Daniela Glätzle-Rützler (Universität Innsbruck) in an experiment with high-school students in Tyrol, Austria (Schwaz and Innsbruck). The students could invest X points (0-100) in each of 9 rounds in a lottery. The payouts were 100 + 2.5 * X points with probability 1/3 and 100 - X points with probability 2/3. Thus, the expected payouts were 100 + 1/6 * X points. Depending on the treatment in the experiment, the investments could either be modified in each round (treatment: "short") or only in round 1, 4, 7 (treatment "long"). Decisions were either made alone or in teams of two. The points were converted to monetary payouts using a conversion of EUR 0.5 per 100 points for lower grades (Unterstufe, 6-8) or EUR 1.0 per 100 points for upper grades (Oberstufe, 10-12).
From the myopic loss aversion literature (on adults) one would expect that the investments of the players (either single players or teams of two) would depend on all factors: Investments should be

- lower in the short treatment (which would indicate myopia),
- higher for teams (indicating a reduction in loss aversion),
- higher for (teams with) male players,
- increase with age/grade.

See Glätzle-Rützler et al. (2015) for more details and references to the literature. In their original analysis, the investments are analyzes using a panel structure (i.e., 9 separate investments for each team). Here, the data are averaged across rounds for each player, leading to qualitatively similar results. The full data along with replication materials are available in the Harvard Dataverse.

Source


References


See Also

betareg

Examples

```r
options(digits = 4)

## data and add ad-hoc scaling (a la Smithson & Verkuilen)
data("LossAversion", package = "betareg")
LossAversion <- transform(LossAversion,
  invests = (invest * (nrow(LossAversion) - 1) + 0.5)/nrow(LossAversion))

## models: normal (with constant variance), beta, extended-support beta mixture
la_n <- lm(invest ~ grade * (arrangement + age) + male, data = LossAversion)
summary(la_n)

la_b <- betareg(invests ~ grade * (arrangement + age) + male | arrangement + male + grade, data = LossAversion)
summary(la_b)

la_xbx <- betareg(invest ~ grade * (arrangement + age) + male | arrangement + male + grade, data = LossAversion)
summary(la_xbx)
```
## coefficients in XBX are typically somewhat shrunk compared to beta

```r
cbind(XBX = coef(la_xbx), Beta = c(coef(la_b), NA))
```

## predictions on subset: (at least one) male players, higher grades, around age 16

```r
la <- subset(LossAversion, male == "yes" & grade == "10-12" & age >= 15 & age <= 17)
la_nd <- data.frame(arrangement = c("single", "team"), male = "yes", age = 16, grade = "10-12")
```

## empirical vs fitted E(Y)

```r
la_nd$mean_emp <- aggregate(invest ~ arrangement, data = la, FUN = mean)$invest
la_nd$mean_n <- predict(la_n, la_nd)
la_nd$mean_b <- predict(la_b, la_nd)
la_nd$mean_xbx <- predict(la_xbx, la_nd)
la_nd
```

## visualization: all means rather similar

```r
la_mod <- c("Emp", "N", "B", "XBX")
la_col <- unname(palette.colors())[c(1, 2, 4, 4)]
la_lty <- c(1, 5, 5, 1)
matplot(la_nd[, paste0("mean_", tolower(la_mod))], type = "l",
col = la_col, lty = la_lty, lwd = 2, ylab = "E(Y)", main = "E(Y)" , xaxt = "n")
axis(1, at = 1:2, labels = la_nd$arrangement)
legend("topleft", la_mod, col = la_col, lty = la_lty, lwd = 2, bty = "n")
```

## empirical vs. fitted P(Y > 0.95)

```r
la_nd$prob_emp <- aggregate(invest >= 0.95 ~ arrangement, data = la, FUN = mean)$invest
la_nd$prob_n <- pnorm(0.95, mean = la_nd$mean_n, sd = summary(la_n)$sigma, lower.tail = FALSE)
la_nd$prob_b <- 1 - predict(la_b, la_nd, type = "probability", at = 0.95)
la_nd$prob_xbx <- 1 - predict(la_xbx, la_nd, type = "probability", at = 0.95)
la_nd[, -(5:8)]
```

## visualization: only XBX works well

```r
matplot(la_nd[, paste0("prob_", tolower(la_mod))], type = "l",
col = la_col, lty = la_lty, lwd = 2, ylab = "P(Y > 0.95)" , main = "P(Y > 0.95)" , xaxt = "n")
axis(1, at = 1:2, labels = la_nd$arrangement)
legend("topleft", la_mod, col = la_col, lty = la_lty, lwd = 2, bty = "n")
```

---

### MockJurors

**Confidence of Mock Jurors in Their Verdicts**

#### Description

Data with responses of naive mock jurors to the conventional conventional two-option verdict (guilt vs. acquittal) versus a three-option verdict setup (the third option was the Scottish ‘not proven’ alternative), in the presence/absence of conflicting testimonial evidence.
MockJurors

Usage

data("MockJurors", package = "betareg")

Format

A data frame containing 104 observations on 3 variables.

- **verdict**: factor indicating whether a two-option or three-option verdict is requested. (A sum contrast rather than treatment contrast is employed.)
- **conflict**: factor. Is there conflicting testimonial evidence? (A sum contrast rather than treatment contrast is employed.)
- **confidence**: jurors degree of confidence in his/her verdict, scaled to the open unit interval (see below).

Details

The data were collected by Daily (2004) among first-year psychology students at Australian National University. Smithson and Verkuilen (2006) employed the data scaling the original confidence (on a scale 0–100) to the open unit interval: \(((\text{original}_\text{confidence}/100) * 103 - 0.5) / 104\).

The original coding of **conflict** in the data provided from Smithson’s homepage is -1/1 which Smithson and Verkuilen (2006) describe to mean no/yes. However, all their results (sample statistics, histograms, etc.) suggest that it actually means yes/no which was employed in **MockJurors**.

Source

Example 1 from Smithson and Verkuilen (2006) supplements.

References


See Also

- **betareg**, **ReadingSkills**, **StressAnxiety**

Examples

```r
data("MockJurors", package = "betareg")
library("lmtest")

## Smithson & Verkuilen (2006, Table 1)
## variable dispersion model
## (NOTE: numerical rather than analytical Hessian is used for replication,  
## Smithson & Verkuilen erroneously compute one-sided p-values)

mj_vd <- betareg(confidence ~ verdict * conflict | verdict * conflict,  
data = MockJurors, hessian = TRUE)
```
summary(mj_vd)

## model selection for beta regression: null model, fixed dispersion model (p. 61)
mj_null <- betareg(confidence ~ 1 | 1, data = MockJurors)
mj_fd <- betareg(confidence ~ verdict * conflict | 1, data = MockJurors)
lrtest(mj_null, mj_fd)
lrtest(mj_null, mj_vd)
## McFadden's pseudo-R-squared
1 - as.vector(logLik(mj_null)/logLik(mj_vd))

## visualization
if(require("lattice")) {
  histogram(~ confidence | conflict + verdict, data = MockJurors,
            col = "lightgray", breaks = 0:10/10, type = "density")
}

## see demo("SmithsonVerkuilen2006", package = "betareg") for more details

---

**plot.betareg**  
**Diagnostic Plots for betareg Objects**

**Description**

Various types of standard diagnostic plots can be produced, involving various types of residuals, influence measures etc.

**Usage**

```r
## S3 method for class 'betareg'
plot(x, which = 1:4,
caption = c("Residuals vs indices of obs.", "Cook's distance plot",
  "Generalized leverage vs predicted values", "Residuals vs linear predictor",
  "Half-normal plot of residuals", "Predicted vs observed values"),
sub.caption = paste(deparse(x$call), collapse = "\n"), main = "",
ask = prod(par("mfcol")) < length(which) & dev.interactive(),
..., type = "quantile", nsim = 100, level = 0.9)
```

**Arguments**

- `x` : fitted model object of class "betareg".
- `which` : numeric. If a subset of the plots is required, specify a subset of the numbers 1:6.
- `caption` : character. Captions to appear above the plots.
- `sub.caption` : character. Common title-above figures if there are multiple.
- `main` : character. Title to each plot in addition to the above caption.
- `ask` : logical. If TRUE, the user is asked before each plot.
- `...` : other parameters to be passed through to plotting functions.
predict.betareg

- **type**: character indicating type of residual to be used, see `residuals.betareg`.
- **nsim**: numeric. Number of simulations in half-normal plots.
- **level**: numeric. Confidence level in half-normal plots.

**Details**

The `plot` method for `betareg` objects produces various types of diagnostic plots. Most of these are standard for regression models and involve various types of residuals, influence measures etc. See Ferrari and Cribari-Neto (2004) for a discussion of some of these displays.

The `which` argument can be used to select a subset of currently six supported types of displays. The corresponding element of `caption` contains a brief description. In some more detail, the displays are: Residuals (as selected by `type`) vs indices of observations (`which = 1`). Cook’s distances vs indices of observations (`which = 2`). Generalized leverage vs predicted values (`which = 3`). Residuals vs linear predictor (`which = 4`). Half-normal plot of residuals (`which = 5`), which is obtained using a simulation approach. Predicted vs observed values (`which = 6`).

**References**


**See Also**

`betareg`

**Examples**

```r
data("GasolineYield", package = "betareg")

gy <- betareg(yield ~ gravity + pressure + temp10 + temp, data = GasolineYield)

par(mfrow = c(3, 2))
plot(gy, which = 1:6)
par(mfrow = c(1, 1))
```

**predict.betareg**

*Prediction Method for betareg Objects*

**Description**

Extract various types of predictions from beta regression models: either on the scale of responses in (0, 1) or the scale of the linear predictor.
Usage

```r
## S3 method for class 'betareg'
predict(object, newdata = NULL,
    type = c("response", "link", "precision", "variance", "parameters",
             "density", "probability", "quantile"),
    na.action = na.pass, at = 0.5, ...)
```

Arguments

- **object**: fitted model object of class "betareg".
- **newdata**: optionally, a data frame in which to look for variables with which to predict. If omitted, the original observations are used.
- **type**: character indicating type of predictions: fitted means of response ("response"), corresponding linear predictor ("link"), fitted precision parameter phi ("precision"), fitted variances of response ("variance"), or fitted quantile(s) of the response distribution ("quantile").
- **na.action**: function determining what should be done with missing values in newdata. The default is to predict NA.
- **at**: numeric vector indicating the level(s) at which quantiles should be predicted (only if type = "quantile"), defaulting to the median at = 0.5.
- **...**: currently not used.

Details

FIXME: Update to extended type and at processing.
FIXME: Add comments about pit and rootogram.

Examples

```r
options(digits = 4)

data("GasolineYield", package = "betareg")

gy2 <- betareg(yield ~ batch + temp | temp, data = GasolineYield)

cbind(
    predict(gy2, type = "response"),
    predict(gy2, type = "link"),
    predict(gy2, type = "precision"),
    predict(gy2, type = "variance"),
    predict(gy2, type = "quantile", at = c(0.25, 0.5, 0.75))
)

## evaluate cumulative _p_robabilities for (small) new data set
gyd <- GasolineYield[c(1, 5, 10), ]
## CDF at 0.1 for each observation
predict(gy2, newdata = gyd, type = "probability", at = 0.1)
## CDF at each combination of 0.1/0.2 and observations
```
predict(gy2, newdata = gyd, type = "probability", at = c(0.1, 0.2))
## CDF at pairwise combinations of 0.1/0.2/0.3 and observations
predict(gy2, newdata = gyd, type = "probability", at = c(0.1, 0.2, 0.3))
## CDF at all combinations of 0.1/0.2/0.3 and observations
predict(gy2, newdata = gyd, type = "probability", at = rbind(c(0.1, 0.2, 0.3)))

---

**ReadingSkills**

**Dyslexia and IQ Predicting Reading Accuracy**

### Description

Data for assessing the contribution of non-verbal IQ to children’s reading skills in dyslexic and non-dyslexic children.

### Usage

data("ReadingSkills", package = "betareg")

### Format

A data frame containing 44 observations on 3 variables.

- **accuracy** numeric. Reading score with maximum restricted to be 0.99 rather than 1 (see below).
- **dyslexia** factor. Is the child dyslexic? (A sum contrast rather than treatment contrast is employed.)
- **iq** numeric. Non-verbal intelligence quotient transformed to z-scores.
- **accuracy1** numeric. Unrestricted reading score with a maximum of 1 (see below).

### Details

The data were collected by Pammer and Kevan (2004) and employed by Smithson and Verkuilen (2006). The original reading accuracy score was transformed by Smithson and Verkuilen (2006) so that accuracy is in the open unit interval (0, 1) and beta regression can be employed. First, the original accuracy was scaled using the minimal and maximal score (a and b, respectively) that can be obtained in the test: accuracy1 = (original_accuracy - a) / (b - a) (a and b are not provided). Subsequently, accuracy was obtained from accuracy1 by replacing all observations with a value of 1 with 0.99.

Kosmidis and Zeileis (2024) propose to investigate the original unrestricted accuracy1 variable using their extended-support beta mixture regression.

### Source

References


See Also

`betareg`, `MockJurors`, `StressAnxiety`

Examples

```r
options(digits = 4)
data("ReadingSkills", package = "betareg")

## Smithson & Verkuilen (2006, Table 5)
## OLS regression
## (Note: typo in iq coefficient: 0.3954 instead of 0.3594)
rs_ols <- lm(qlogis(accuracy) ~ dyslexia * iq, data = ReadingSkills)
summary(rs_ols)

## Beta regression (with numerical rather than analytic standard errors)
## (Note: Smithson & Verkuilen erroneously compute one-sided p-values)
rs_beta <- betareg(accuracy ~ dyslexia * iq | dyslexia + iq, 
  data = ReadingSkills, hessian = TRUE)
summary(rs_beta)

## Extended-support beta mixture regression (Kosmidis & Zeileis 2024)
rs_xbx <- betareg(accuracy1 ~ dyslexia * iq | dyslexia + iq, data = ReadingSkills)
summary(rs_xbx)

## Coefficients in XBX are typically somewhat shrunken compared to beta
cbind(XBX = coef(rs_xbx), Beta = c(coef(rs_beta), NA))

## Visualization
plot(accuracy1 ~ iq, data = ReadingSkills, col = c(4, 2)[dyslexia], pch = 19)
nd <- data.frame(dyslexia = "no", iq = -30:30/10)
lines(nd$id, predict(rs_xbx, nd), col = 4)
lines(nd$id, predict(rs_beta, nd), col = 4, lty = 5)
lines(nd$id, plogis(predict(rs_ols, nd)), col = 4, lty = 3)
nd <- data.frame(dyslexia = "yes", iq = -30:30/10)
lines(nd$id, predict(rs_xbx, nd), col = 2)
lines(nd$id, predict(rs_beta, nd), col = 2, lty = 5)
lines(nd$id, plogis(predict(rs_ols, nd)), col = 2, lty = 3)
```
residuals.betareg

Description

Extract various types of residuals from beta regression models: raw response residuals (observed - fitted), Pearson residuals (raw residuals scaled by square root of variance function), deviance residuals (scaled log-likelihood contributions), and different kinds of weighted residuals suggested by Espinheira et al. (2008).

Usage

## S3 method for class 'betareg'
residuals(object, type = c("quantile", "deviance", "pearson", "response", "weighted", "sweighted", "sweighted2"), ...)

Arguments

object
fitted model object of class "betareg".
type
character indicating type of residuals.
...
currently not used.

Details

The default residuals (starting from version 3.2-0) are quantile residuals as proposed by Dunn and Smyth (1996) and explored in the context of beta regression by Pereira (2017). In case of extended support beta regression with boundary observations at 0 and/or 1, the quantile residuals for the boundary observations are randomized.

The definitions of all other residuals are provided in Espinheira et al. (2008): Equation 2 for "pearson", last equation on page 409 for "deviance", Equation 6 for "weighted", Equation 7 for "sweighted", and Equation 8 for "sweighted2".

Espinheira et al. (2008) recommend to use "sweighted2", hence this was the default prior to version 3.2-0. However, these are rather burdensome to compute because they require operations of $O(n^2)$ and hence are typically prohibitively costly in large sample. Also they are not available for extended support beta regression. Finally, Pereira (2017) found quantile residuals to have better distributional properties.
References


See Also

*betareg*

Examples

```r
options(digits = 4)

data("GasolineYield", package = "betareg")

gy <- betareg(yield ~ gravity + pressure + temp10 + temp, data = GasolineYield)

gy_res <- cbind(
  "quantile" = residuals(gy, type = "quantile"),
  "pearson" = residuals(gy, type = "pearson"),
  "deviance" = residuals(gy, type = "deviance"),
  "response" = residuals(gy, type = "response"),
  "weighted" = residuals(gy, type = "weighted"),
  "sweighted" = residuals(gy, type = "sweighted"),
  "sweighted2" = residuals(gy, type = "sweighted2")
)
pairs(gy_res)

cor(gy_res)
```

<table>
<thead>
<tr>
<th>StressAnxiety</th>
<th>Dependency of Anxiety on Stress</th>
</tr>
</thead>
</table>

Description

Stress and anxiety among nonclinical women in Townsville, Queensland, Australia.

Usage

```r
data("StressAnxiety", package = "betareg")
```
Format

A data frame containing 166 observations on 2 variables.

- **stress**: score, linearly transformed to the open unit interval (see below).
- **anxiety**: score, linearly transformed to the open unit interval (see below).

Details

Both variables were assess on the Depression Anxiety Stress Scales, ranging from 0 to 42. Smithson and Verkuilen (2006) transformed these to the open unit interval (without providing details about this transformation).

Source


References


See Also

- betareg
- MockJurors
- ReadingSkills

Examples

data("StressAnxiety", package = "betareg")
StressAnxiety <- StressAnxiety[order(StressAnxiety$stress),]

## Smithson & Verkuilen (2006, Table 4)
sa_null <- betareg(anxiety ~ 1 | 1,
  data = StressAnxiety, hessian = TRUE)
sa_stress <- betareg(anxiety ~ stress | stress,
  data = StressAnxiety, hessian = TRUE)
sa_null <- betareg(as.numeric(stress) - 1 | 1,
  data = StressAnxiety, hessian = TRUE)
sa_stress <- betareg(as.numeric(anxiety) - stress | stress,
  data = StressAnxiety, hessian = TRUE)
sa_null <- betareg(as.numeric(stress) - 1 | 1,
  data = StressAnxiety, hessian = TRUE)
sa_stress <- betareg(as.numeric(anxiety) - stress | stress,
  data = StressAnxiety, hessian = TRUE)

summary(sa_null)
sa_stress
AIC(sa_null, sa_stress)
1 - as.vector(logLik(sa_null)/logLik(sa_stress))

## visualization
attach(StressAnxiety)
plot(jitter(anxiety) ~ jitter(stress),
  xlab = "Stress", ylab = "Anxiety",
  xlim = c(0, 1), ylim = c(0, 1))
lowess_plot <- lowess(anxiety ~ stress)
lowess_plot <- lowess_plot
lines(lowess_plot)
lines(fitted(sa_stress) ~ stress, lty = 2)
lines(fitted(lm(anxiety ~ stress)) ~ stress, lty = 3)
legend("topleft", c("lowess", "betareg", "lm"), lty = 1:3, bty = "n")
detach(StressAnxiety)

## see `demo("SmithsonVerkuilen2006", package = "betareg")` for more details
Methods for extracting information from fitted beta regression model objects of class "betareg".

Usage

## S3 method for class 'betareg'
summary(object, phi = NULL, type = "quantile", ...)

## S3 method for class 'betareg'
coef(object, model = c("full", "mean", "precision"), phi = NULL, ...)

## S3 method for class 'betareg'
vcov(object, model = c("full", "mean", "precision"), phi = NULL, ...)

## S3 method for class 'betareg'
bread(x, phi = NULL, ...)

## S3 method for class 'betareg'
estfun(x, phi = NULL, ...)

Arguments

object, x fitted model object of class "betareg".

phi logical indicating whether the parameters in the precision model (for phi) should be reported as full model parameters (TRUE) or nuisance parameters (FALSE). The default is taken from object$phi.

type character specifying type of residuals to be included in the summary output, see residuals.betareg.

model character specifying for which component of the model coefficients/covariance should be extracted. (Only used if phi is NULL.)

... currently not used.

Details

A set of standard extractor functions for fitted model objects is available for objects of class "betareg", including methods to the generic functions print and summary which print the estimated coefficients along with some further information. The summary in particular supplies partial Wald tests based on the coefficients and the covariance matrix. As usual, the summary method returns an object of class "summary.betareg" containing the relevant summary statistics which can subsequently be printed using the associated print method.

A logLik method is provided, hence AIC can be called to compute information criteria.
WeatherTask

References


See Also

betareg

Examples

```r
options(digits = 4)
data("GasolineYield", package = "betareg")
gy2 <- betareg(yield ~ batch + temp | temp, data = GasolineYield)
summary(gy2)
coef(gy2)
vcov(gy2)
logLik(gy2)
AIC(gy2)

coef(gy2, model = "mean")
coef(gy2, model = "precision")
summary(gy2, phi = FALSE)
```

WeatherTask

*Weather Task With Priming and Precise and Imprecise Probabilities*

Description

In this study participants were asked to judge how likely Sunday is to be the hottest day of the week.

Usage

```r
data("WeatherTask", package = "betareg")
```

Format

A data frame with 345 observations on the following 3 variables.

- **priming**: a factor with levels two-fold (case prime) and seven-fold (class prime).
- **eliciting**: a factor with levels precise and imprecise (lower and upper limit).
- **agreement**: a numeric vector, probability indicated by participants or the average between minimum and maximum probability indicated.
Details

All participants in the study were either first- or second-year undergraduate students in psychology, none of whom had a strong background in probability or were familiar with imprecise probability theories.

For priming the questions were:

**two-fold** [What is the probability that] the temperature at Canberra airport on Sunday will be higher than every other day next week?

**seven-fold** [What is the probability that] the highest temperature of the week at Canberra airport will occur on Sunday?

For eliciting the instructions were if

- **precise** to assign a probability estimate,
- **imprecise** to assign a lower and upper probability estimate.

Source

Taken from Smithson et al. (2011) supplements.

References


Examples

```r
data("WeatherTask", package = "betareg")
library("flexmix")
wt_betamix <- betamix(agreement ~ 1, data = WeatherTask, k = 2,
extra_components = extraComponent(type = "betareg", coef =
list(mean = 0, precision = 2)),
FLXconcomitant = FLXPmultinom(~ priming + eliciting))
```

XBeta

Create an Extended-Support Beta Distribution

Description

Class and methods for extended-support beta distributions using the workflow from the *distributions3* package.

Usage

```r
XBeta(mu, phi, nu = 0)
```
Arguments

mu numeric. The mean of the underlying beta distribution on [-\(\nu\), 1 + \(\nu\)].

phi numeric. The precision parameter of the underlying beta distribution on [-\(\nu\), 1 + \(\nu\)].

\(\nu\) numeric. Exceedence parameter for the support of the underlying beta distribution on [-\(\nu\), 1 + \(\nu\)] that is censored to [0, 1].

Details

In order to obtain an extended-support beta distribution on [0, 1] an additional exceedence parameter \(\nu\) is introduced. If \(\nu > 0\), this scales the underlying beta distribution to the interval [-\(\nu\), 1 + \(\nu\)] where the tails are subsequently censored to the unit interval [0, 1] with point masses on the boundaries 0 and 1. Thus, \(\nu\) controls how likely boundary observations are and for \(\nu = 0\) (the default), the distribution reduces to the classic beta distribution (in regression parameterization) without boundary observations.

Value

A \textit{XBeta} distribution object.

See Also

dxbeta, BetaR

Examples

```r
## package and random seed
library("distributions3")
set.seed(6020)

## three beta distributions
X <- XBeta(
  mu = c(0.25, 0.50, 0.75),
  phi = c(1, 1, 2),
  nu = c(0, 0.1, 0.2)
)

X

## compute moments of the distribution
mean(X)
variance(X)

## support interval (minimum and maximum)
support(X)

## it is only continuous when there are no point masses on the boundary
is_continuous(X)
cdf(X, 0)
cdf(X, 1, lower.tail = FALSE)
```
## simulate random variables
random(X, 5)

## histograms of 1,000 simulated observations
x <- random(X, 1000)
hist(x[, 1])
hist(x[, 2])
hist(x[, 3])

## probability density function (PDF) and log-density (or log-likelihood)
x <- c(0.25, 0.5, 0.75)
pdf(X, x)
pdf(X, x, log = TRUE)
log_pdf(X, x)

## cumulative distribution function (CDF)
cdf(X, x)

## quantiles
quantile(X, 0.5)

## cdf() and quantile() are inverses (except at censoring points)
cdf(X, quantile(X, 0.5))
quantile(X, cdf(X, 1))

## all methods above can either be applied elementwise or for
## all combinations of X and x, if length(X) = length(x),
## also the result can be assured to be a matrix via drop = FALSE
p <- c(0.05, 0.5, 0.95)
quantile(X, p, elementwise = FALSE)
quantile(X, p, elementwise = TRUE)
quantile(X, p, elementwise = TRUE, drop = FALSE)

## compare theoretical and empirical mean from 1,000 simulated observations
cbind(
    "theoretical" = mean(X),
    "empirical" = rowMeans(random(X, 1000))
)

---

**xbeta**

**The Extended-Support Beta Distribution**

**Description**

Density, distribution function, quantile function, and random generation for the extended-support beta distribution (in regression parameterization) on [0, 1].
Usage

dxbeta(x, mu, phi, nu = 0, log = FALSE)
pxbeta(q, mu, phi, nu = 0, lower.tail = TRUE, log.p = FALSE)
qxbeta(p, mu, phi, nu = 0, lower.tail = TRUE, log.p = FALSE)
rxbeta(n, mu, phi, nu = 0)

Arguments

x, q numeric. Vector of quantiles.
p numeric. Vector of probabilities.
n numeric. Number of observations. If length(n) > 1, the length is taken to be the number required.
mu numeric. The mean of the underlying beta distribution on [-nu, 1 + nu].
phi numeric. The precision parameter of the underlying beta distribution on [-nu, 1 + nu].
nu numeric. Exceedence parameter for the support of the underlying beta distribution on [-nu, 1 + nu] that is censored to [0, 1].
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].

Details

In order to obtain an extended-support beta distribution on [0, 1] an additional exceedence parameter nu is introduced. If nu > 0, this scales the underlying beta distribution to the interval [-nu, 1 + nu] where the tails are subsequently censored to the unit interval [0, 1] with point masses on the boundaries 0 and 1. Thus, nu controls how likely boundary observations are and for nu = 0 (the default), the distribution reduces to the classic beta distribution (in regression parameterization) without boundary observations.

Value

dxbeta gives the density, pxbeta gives the distribution function, qxbeta gives the quantile function, and rxbeta generates random deviates.

See Also
dbetar, XBeta
Create an Extended-Support Beta Mixture Distribution

Description

Class and methods for extended-support beta distributions using the workflow from the `distributions3` package.

Usage

XBetaX(mu, phi, nu = 0)

Arguments

- **mu**: numeric. The mean of the underlying beta distribution on [-nu, 1 + nu].
- **phi**: numeric. The precision parameter of the underlying beta distribution on [-nu, 1 + nu].
- **nu**: numeric. Mean of the exponentially-distributed exceedence parameter for the underlying beta distribution on [-nu, 1 + nu] that is censored to [0, 1].

Details

The extended-support beta mixture distribution is a continuous mixture of extended-support beta distributions on [0, 1] where the underlying exceedence parameter is exponentially distributed with mean `nu`. Thus, if `nu > 0`, the resulting distribution has point masses on the boundaries 0 and 1 with larger values of `nu` leading to higher boundary probabilities. For `nu = 0` (the default), the distribution reduces to the classic beta distribution (in regression parameterization) without boundary observations.

Value

A XBetaX distribution object.

See Also

dxbetax, XBeta

Examples

```r
## package and random seed
library("distributions3")
set.seed(6020)

## three beta distributions
X <- XBetaX(
  mu = c(0.25, 0.50, 0.75),
  phi = c(1, 1, 2),
  nu = c(0, 0.1, 0.2)
)```

## compute moments of the distribution

mean(X)

variance(X)

## support interval (minimum and maximum)

support(X)

## it is only continuous when there are no point masses on the boundary

is_continuous(X)
cdf(X, 0)
cdf(X, 1, lower.tail = FALSE)

## simulate random variables

random(X, 5)

## histograms of 1,000 simulated observations

x <- random(X, 1000)
hist(x[1, ])
hist(x[2, ])
hist(x[3, ])

## probability density function (PDF) and log-density (or log-likelihood)

x <- c(0.25, 0.5, 0.75)

pdf(X, x)
pdf(X, x, log = TRUE)
log_pdf(X, x)

## cumulative distribution function (CDF)

cdf(X, x)

## quantiles

quantile(X, 0.5)

## cdf() and quantile() are inverses (except at censoring points)

cdf(X, quantile(X, 0.5))
quantile(X, cdf(X, 1))

## all methods above can either be applied elementwise or for
## all combinations of X and x, if length(X) = length(x),
## also the result can be assured to be a matrix via drop = FALSE

p <- c(0.05, 0.5, 0.95)

quantile(X, p, elementwise = FALSE)
quantile(X, p, elementwise = TRUE)
quantile(X, p, elementwise = TRUE, drop = FALSE)

## compare theoretical and empirical mean from 1,000 simulated observations

cbind(
  "theoretical" = mean(X),
  "empirical" = rowMeans(random(X, 1000))
)
The Extended-Support Beta Mixture Distribution

Description

Density, distribution function, quantile function, and random generation for the extended-support beta mixture distribution (in regression parameterization) on \([0, 1]\).

Usage

\[
dxbetax(x, mu, phi, nu = 0, log = FALSE, quad = 20)
pxbetax(q, mu, phi, nu = 0, lower.tail = TRUE, log.p = FALSE, quad = 20)
qxbetax(p, mu, phi, nu = 0, lower.tail = TRUE, log.p = FALSE, quad = 20,
tol = .Machine$double.eps^0.7)
rxbetax(n, mu, phi, nu = 0)
\]

Arguments

- \(x, q\) numeric. Vector of quantiles.
- \(p\) numeric. Vector of probabilities.
- \(n\) numeric. Number of observations. If length(n) > 1, the length is taken to be the number required.
- \(mu\) numeric. The mean of the underlying beta distribution on \([-nu, 1 + nu]\).
- \(phi\) numeric. The precision parameter of the underlying beta distribution on \([-nu, 1 + nu]\).
- \(nu\) numeric. Mean of the exponentially-distributed exceedence parameter for the underlying beta distribution on \([-nu, 1 + nu]\) that is censored to \([0, 1]\).
- \(log, log.p\) logical. If TRUE, probabilities \(p\) are given as \(\log(p)\).
- \(lower.tail\) logical. If TRUE (default), probabilities are \(P[X \leq x]\) otherwise, \(P[X > x]\).
- \(quad\) numeric. The number of quadrature points for numeric integration of the continuous mixture. Alternatively, a matrix with nodes and weights for the quadrature points can be specified.
- \(tol\) numeric. Accuracy (convergence tolerance) for numerically determining quantiles based on \texttt{uniroot} and \texttt{pxbetax}. 
**Details**

The extended-support beta mixture distribution is a continuous mixture of extended-support beta distributions on [0, 1] where the underlying exceedence parameter is exponentially distributed with mean $\nu$. Thus, if $\nu > 0$, the resulting distribution has point masses on the boundaries 0 and 1 with larger values of $\nu$ leading to higher boundary probabilities. For $\nu = 0$ (the default), the distribution reduces to the classic beta distribution (in regression parameterization) without boundary observations.

**Value**

dxbetax gives the density, pxbetax gives the distribution function, qxbetax gives the quantile function, and rxbetax generates random deviates.

**See Also**

dxbeta, XBetaX
Index

* cluster
  betamix, 9
* datasets
  CarTask, 24
  FoodExpenditure, 25
  GasolineYield, 26
  ImpreciseTask, 29
  LossAversion, 30
  MockJurors, 32
  ReadingSkills, 37
  StressAnxiety, 40
  WeatherTask, 43
* distribution
  beta01, 4
  beta4, 8
  betar, 14
  xbeta, 46
  xbetax, 50
* regression
  betamix, 9
  betareg, 15
  betareg.control, 20
  gleverage, 28
  plot.betareg, 34
  predict.betareg, 35
  residuals.betareg, 39
  summary.betareg, 42
* tree
  betatree, 22

AIC, 42

Beta, 12, 13
Beta01, 2, 5
beta01, 4
Beta4, 6, 8
beta4, 7
betamix, 9
BetaR, 3, 6, 12, 15, 45
betar, 14

betareg, 9–11, 15, 20–23, 25, 27, 28, 31, 33, 35, 38, 40, 41, 43
betareg.control, 9, 10, 16, 17, 20, 21, 22
betareg.fit, 23
betatree, 22
bread, 17
bread.betareg (summary.betareg), 42

CarTask, 24
cdf.Beta01 (Beta01), 2
cdf.Beta4 (Beta4), 6
cdf.BetaR (BetaR), 12
cdf.XBeta (XBeta), 44
cdf.XBetaX (XBetaX), 48
clusters, betamix, ANY-method (betamix), 9
coef, 17
coef.betareg (summary.betareg), 42
coeffest, 17
coeffest.betareg (summary.betareg), 42
cooks.distance.betareg
  (summary.betareg), 42
dbeta, 15
dbeta01, 3
dbeta01 (beta01), 4
dbeta4, 6
dbeta4 (beta4), 8
dbetar, 5, 8, 13, 47
dbetar (betar), 14
dxbeta, 45, 51
dxbeta (xbeta), 46
dxbetax, 48
dxbetax (xbetax), 50

estfun, 17
estfun.betareg (summary.betareg), 42
extraComponent (betamix), 9
fitted, betamix-method (betamix), 9
fitted, FLXMRbeta-method (betamix), 9
flmix, 10, 11
FLXPconstant, 10
FoodExpenditure, 25
Formula, 19
GasolineYield, 26
gleverage, 17, 28
hatvalues.betareg (summary.betareg), 42
ImpreciseTask, 29
influence.measures, 17
is_continuous.Beta01 (Beta01), 2
is_continuous.Beta4 (Beta4), 6
is_continuous.BetaR (BetaR), 12
is_continuous.XBeta (XBeta), 44
is_continuous.XBetaX (XBetaX), 48
is_discrete.Beta01 (Beta01), 2
is_discrete.Beta4 (Beta4), 6
is_discrete.BetaR (BetaR), 12
is_discrete.XBeta (XBeta), 44
is_discrete.XBetaX (XBetaX), 48
kurtosis.Beta01 (Beta01), 2
kurtosis.Beta4 (Beta4), 6
kurtosis.BetaR (BetaR), 12
kurtosis.XBeta (XBeta), 44
kurtosis.XBetaX (XBetaX), 48
lm.wfit, 21
log_pdf.Beta01 (Beta01), 2
log_pdf.Beta4 (Beta4), 6
log_pdf.BetaR (BetaR), 12
log_pdf.XBeta (XBeta), 44
log_pdf.XBetaX (XBetaX), 48
logLik, 17, 42
logLik.betareg (summary.betareg), 42
LossAversion, 30
mean.Beta01 (Beta01), 2
mean.Beta4 (Beta4), 6
mean.BetaR (BetaR), 12
mean.XBeta (XBeta), 44
mean.XBetaX (XBetaX), 48
mob, 22, 23
mob_control, 22
MockJurors, 32, 38, 41
model.frame, 9, 15, 17
model.frame.betareg (summary.betareg), 42
model.matrix, 17
model.matrix.betareg (summary.betareg), 42
nlminb, 20
node_bivplot, 22
optim, 16, 20, 21
pbeta01 (beta01), 4
pbeta4 (beta4), 8
pbetar (betar), 14
df.Beta01 (Beta01), 2
df.Beta4 (Beta4), 6
df.BetaR (BetaR), 12
df.XBeta (XBeta), 44
df.XBetaX (XBetaX), 48
pit.betareg (predict.betareg), 35
plot, 17
plot.betareg, 17, 34
plot.betatree (betatree), 22
posterior, betamix, ANY-method (betamix), 9
predict, 17
predict, betamix-method (betamix), 9
predict, FLXMRbeta-method (betamix), 9
predict, FLXMRbetafix-method (betamix), 9
predict.betareg, 17, 19, 35
predict.betatree (betatree), 22
print, 17, 42
print.betareg (summary.betareg), 42
print.betatree (betatree), 22
print.summary.betareg (summary.betareg), 42
pxbeta (xbeta), 46
pxbetax (xbetax), 50
qbeta01 (beta01), 4
qbeta4 (beta4), 8
qbetar (betar), 14
quantile.Beta01 (Beta01), 2
quantile.Beta4 (Beta4), 6
quantile.BetaR (BetaR), 12
quantile.XBeta (XBeta), 44
quantile.XBetaX (XBetaX), 48
qxbeta (xbeta), 46
qxbetax (xbetax), 50
random.Beta01 (Beta01), 2
random.Beta4 (Beta4), 6
random.BetaR (BetaR), 12
random.XBeta (XBeta), 44
random.XBetaX (XBetaX), 48
rbeta01 (beta01), 4
rbeta4 (beta4), 8
rbetar (betar), 14
ReadingSkills, 33, 37, 41
residuals, 17
residuals.betareg, 17, 19, 35, 39, 42
rootogram.betareg (predict.betareg), 35
rxbeta (xbeta), 46
rxbetax (xbetax), 50
sctest.betatree (betatree), 22
skewness.Beta01 (Beta01), 2
skewness.Beta4 (Beta4), 6
skewness.BetaR (BetaR), 12
skewness.XBeta (XBeta), 44
skewness.XBetaX (XBetaX), 48
stepFlexmix, 9–11
StressAnxiety, 33, 38, 40
summary, 17, 42
summary.betareg, 17, 19, 42
support.Beta01 (Beta01), 2
support.Beta4 (Beta4), 6
support.BetaR (BetaR), 12
support.XBeta (XBeta), 44
support.XBetaX (XBetaX), 48
terms, 17
terms.betareg (summary.betareg), 42
univertex, 50
variance.Beta01 (Beta01), 2
variance.Beta4 (Beta4), 6
variance.BetaR (BetaR), 12
variance.XBeta (XBeta), 44
variance.XBetaX (XBetaX), 48
vcov, 17
vcov.betareg (summary.betareg), 42
WeatherTask, 29, 43
XBeta, 44, 47, 48
xbeta, 46
XBetaX, 48, 51
xbetax, 50