Package ‘betareg’

February 3, 2020

Version 3.1-3
Date 2020-01-28
Title Beta Regression
Description Beta regression for modeling beta-distributed dependent variables, e.g., rates and proportions. In addition to maximum likelihood regression (for both mean and precision of a beta-distributed response), bias-corrected and bias-reduced estimation as well as finite mixture models and recursive partitioning for beta regressions are provided.
Depends R (>= 3.0.0)
Imports graphics, grDevices, methods, stats, flexmix, Formula, lmtest, modeltools, sandwich
Suggests car, lattice, partykit, strucchange
License GPL-2 | GPL-3
NeedsCompilation no
Author Achim Zeileis [aut, cre], Francisco Cribari-Neto [aut], Bettina Gruen [aut], Ioannis Kosmidis [aut], 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 2020-02-03 09:40:05 UTC

R topics documented:

betamix ................................................................. 2
betareg ................................................................. 5
betareg.control ....................................................... 9
betatree ............................................................... 11
CarTask ............................................................... 13
FoodExpenditure .................................................... 14
**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`. 

---

**betamix**

*Finite Mixtures of Beta Regression for Rates and Proportions*

---

Index

- GasolineYield
- gleverage
- ImpreciseTask
- MockJurors
- plot.betareg
- predict.betareg
- ReadingSkills
- residuals.betareg
- StressAnxiety
- summary.betareg
- WeatherTask

---

**betamix**

*Finite Mixtures of Beta Regression for Rates and Proportions*
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 \(y \sim x\) 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 \texttt{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 \texttt{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 \texttt{flexmix}.

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

nstart for each value of \(k\) run \texttt{stepFlexmix nstart} 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 \texttt{extraComponent()}. ... arguments passed to \texttt{betareg.control}.

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

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 \texttt{betareg}. Internally \texttt{stepFlexmix} is called with suitable arguments to fit the finite mixture model with the EM algorithm. See Grün et al. (2012) for more details.

\texttt{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

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

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

set.seed(4840)
## 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,
## scatter plot
plot(accuracy ~ iq, data = ReadingSkills, col = col2, pch = 19, cex = 1.5, xlab = 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
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")

---

### 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"), 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())
Arguments

- `formula`: symbolic description of the model (of type `y ~ x` or `y ~ x | z`; 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.
- `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 phi). The mean is linked, as in generalized linear models (GLMs), to the responses 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. Estimation is performed by maximum likelihood (ML) via `optim` using analytical gradients and (by default) 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. This slightly improves the optimization by moving the gradients even closer to zero (for type = "ML" and "BC") or solving the bias-adjusted estimating equations (for type = "BR"). For the former two estimators, the optional Fisher scoring can be disabled by setting `fsmaxit = 0` in the control arguments. See Cribari-Neto and Zeileis (2010) and Grün et al. (2012) for details.
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 ($z_1 + z_2$, 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 $y \sim x_1 + x_2$), the "identity" link is used by default for backward compatibility.

Simas et al. (2009) also suggest further extensions (non-linear specifications, bias correction) which are not yet implemented in betareg. However, Kosmidis and Firth (2010) discuss general algorithms for bias correction/reduction, both of which are available in betareg by setting the type argument accordingly. (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 dmu.deta function. However, if a "link-glm" object is supplied directly by the user, it needs to have the dmu.deta function.)

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.

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 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**

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

References


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, hessian = FALSE, trace = FALSE, start = NULL, fsmaxit = 200, fstol = 1e-8, ...)

See Also

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

Examples

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)
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 character string specifying the method argument passed to optim.

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).

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.

... 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)
```

betatree  

**Beta Regression Trees**

**Description**

Fit beta regression trees via model-based recursive partitioning.

**Usage**

```r
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., \(~ p1 + p2 \). The argument `partition` can be omitted if `formula` is a three-part formula of type \( y \sim x \mid z \mid p1 + p2 \).
- **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.
betatree

control a list of control arguments for the beta regression specified via \texttt{betareg.control}.

... further control arguments for the recursive partitioning passed to \texttt{mob.control}.

Details

Beta regression trees are an application of model-based recursive partitioning (implemented in \texttt{mob}, see Zeileis et al. 2008) to beta regression (implemented in \texttt{betareg}, see Cribari-Neto and Zeileis 2010). See also Grün et 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 \texttt{node.bivplot} panel-generating function.

Value

\texttt{betatree()} returns an object of S3 class "betatree" which inherits from "modelparty".

References


See Also

\texttt{betareg}, \texttt{betareg.fit}, \texttt{mob}

Examples

\begin{verbatim}
  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)
\end{verbatim}
## 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, msize = 10)
plot(bt2)

## inspect result
coef(bt2)
if(require("strucchange")) sctest(bt2)
## IGNORE_RDIFF_BEGIN
summary(bt2, node = 2)
summary(bt2, node = 3)
## IGNORE_RDIFF_END

---

**CarTask**

*Partition-primed Probability Judgement Task for Car Dealership*

---

**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 form a specific salesperson out of four possible salespersons.

**Usage**

data(CarTask)

**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 task the 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))
```

<table>
<thead>
<tr>
<th>FoodExpenditure</th>
<th>Proportion of Household Income Spent on Food</th>
</tr>
</thead>
</table>

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")
```
Gasoline Yield

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)
```

---

Gasoline Yield  Estimation of Gasoline Yields from Crude Oil

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

data("GasolineYield")

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)
```

### Data

**ImpreciseTask**

**Imprecise Probabilities for Sunday Weather and Boeing Stock Task**

**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)
```

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

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.

Usage

data("MockJurors")
MockJurors

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: \(((\text{original confidence}/100) \times 103 - 0.5) / 104\).

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

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)
```r
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 = "sweighted2", 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.
- `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

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", "quantile"),
    na.action = na.pass, at = 0.5, ...)
```
**ReadingSkills**

**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.

**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))
)
```

**Description**

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

**Usage**

```r
data("ReadingSkills")
```

**Format**

A data frame containing 44 observations on 3 variables.

- **accuracy** reading score scaled to the open unit interval (see below).
- **dyslexia** factor. Is the child dyslexic? (A sum contrast rather than treatment contrast is employed.)
- **iq** non-verbal intelligence quotient transformed to z-scores.
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: (original_accuracy - a) / (b - a) (a and b are not provided). Subsequently, the scaled score is transformed to the unit interval using a continuity correction: (scaled_accuracy * (n-1) - 0.5) / n (either with some rounding or using n = 50 rather than 44).

Source


References


See Also

betareg, MockJurors, StressAnxiety

Examples

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)

## visualization
plot(accuracy ~ iq, data = ReadingSkills, col = as.numeric(dyslexia), pch = 19)
nd <- data.frame(dyslexia = "no", iq = -30:30/10)
lines(nd$iq, predict(rs_beta, nd))
```r
lines(nd$q, plogis(predict(rs_ols, nd)), lty = 2)
nd <- data.frame(dyslexia = "yes", iq = -30:30/10)
lines(nd$q, predict(rs_beta, nd), col = 2)
lines(nd$q, plogis(predict(rs_ols, nd)), col = 2, lty = 2)

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

**residuals.betareg**

**Residuals Method for betareg Objects**

**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**

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

**Arguments**

- **object**
  - Fitted model object of class "betareg".
- **type**
  - Character indicating type of residuals.
- **...**
  - Currently not used.

**Details**

The definitions of all 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 is the default in the residuals() method. Note, however, that these are rather burdensome to compute because they require operations of $O(n^2)$ and hence might be prohibitively costly in large sample.

**References**


### StressAnxiety

#### Dependency of Anxiety on Stress

**Description**

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

**Usage**

```r
data("StressAnxiety")
```

**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)
summary(sa_null)
summary(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))
lines(lowess(anxiety ~ stress))
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

summary.betareg

Methods for betareg Objects

Description

Methods for extracting information from fitted beta regression model objects of class "betareg".
Usage

```r
## S3 method for class 'betareg'
summary(object, phi = NULL, type = "sweighted2", ...)

## 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. Note that the default residuals "sweighted2" might be burdensome to compute in large samples and hence might need modification in such applications.

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

References


See Also

betareg

Examples

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)

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

data(WeatherTask)

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

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))
Index

*Topic **cluster**
- betamix, 2

*Topic **datasets**
- CarTask, 13
- FoodExpenditure, 14
- GasolineYield, 15
- ImpreciseTask, 18
- MockJurors, 19
- ReadingSkills, 23
- StressAnxiety, 26
- WeatherTask, 29

*Topic **regression**
- betamix, 2
- betareg, 5
- betareg.control, 9
- gleverage, 17
- plot.betareg, 21
- predict.betareg, 22
- residuals.betareg, 25
- summary.betareg, 27
- cooks.distance.betareg (summary.betareg), 27
- estfun, 7
- estfun.betareg (summary.betareg), 27
- extraComponent (betamix), 2
- fitted, betamix-method (betamix), 2
- fitted, FLXMRbeta-method (betamix), 2
- flexmix, 3, 4
- FLXPconstant, 3
- FoodExpenditure, 14
- Formula, 9
- GasolineYield, 15
- gleverage, 7, 17
- hatvalues.betareg (summary.betareg), 27
- ImpreciseTask, 18
- influence.measures, 7
- lm.wfit, 10
- logLik, 7, 28
- logLik.betareg (summary.betareg), 27
- mob, 11, 12
- mob_control, 12
- MockJurors, 19, 24, 27
- model.frame, 2, 6, 7
- model.frame.betareg (summary.betareg), 27
- model.matrix, 7
- model.matrix.betareg (summary.betareg), 27
- node_bivplot, 12
- optim, 6, 10
- plot, 7

AIC, 28

betamix, 2
betareg, 2–4, 5, 9–12, 15, 16, 18, 20, 22, 24, 26, 27, 29
betareg.control, 3, 6, 7, 9, 10, 12
betareg.fit, 12
betatree, 11
bread, 7
bread.betareg (summary.betareg), 27

CarTask, 13
clusters, betamix, ANY-method (betamix), 2
coeftest, 7
coeftest.betareg (summary.betareg), 27

coeftest, 7
coeftest.betareg (summary.betareg), 27
plot.betareg, 7, 21
plot.betatree (betatree), 11
posterior, betamix, ANY-method (betamix), 2
predict, 7
predict, betamix-method (betamix), 2
predict, FLXMRbeta-method (betamix), 2
predict, FLXMRbetafix-method (betamix), 2
predict.betareg, 7, 9, 22
predict.betatree (betatree), 11
print, 7, 28
print.betareg (summary.betareg), 27
print.betatree (betatree), 11
print.summary.betareg
    (summary.betareg), 27

ReadingSkills, 20, 23, 27
residuals, 7
residuals.betareg, 7, 9, 21, 25, 28

sctest.betatree (betatree), 11
stepFlexmix, 2–4
StressAnxiety, 20, 24, 26
summary, 7, 28
summary.betareg, 7, 9, 27

terms, 7
terms.betareg (summary.betareg), 27

vcov, 7
vcov.betareg (summary.betareg), 27

WeatherTask, 19, 29