Package 'robustbetareg'

May 9, 2026

Version 0.3.1

Title Robust Beta Regression

Author Felipe Queiroz [aut, cre],
Yuri Maluf [aut],
Silvia Ferrari [ctb]

Maintainer Felipe Queiroz <ffelipeq@outlook.com>

Description Robust estimators for the beta regression, useful for modeling bounded continuous data. Currently, four types of robust estimators are supported. They depend on a tuning constant which may be fixed or selected by a data-driven algorithm also implemented in the package. Diagnostic tools associated with the fitted model, such as the residuals and goodness-of-fit statistics, are implemented. Robust Wald-type tests are available. More details about robust beta regression are described in Maluf et al. (2025) <doi:10.1007/s00184-024-00949-1>.

Depends R (>= 3.5.0), betareg

Imports Rmpfr, rstudioapi, crayon, pracma, numDeriv, Formula, robustbase, zoo, methods, graphics, BBmisc, MASS, miscTools, Matrix

License GPL-3

NeedsCompilation no

Encoding UTF-8

LazyData true

RoxygenNote 7.3.2

Suggests covr, testthat (>= 3.0.0)

Config/testthat/edition 3

Repository CRAN

Date/Publication 2025-07-24 13:10:02 UTC

Contents

EGB *The Exponential Generalized Beta of the Second Type Distribution*

Description

Density, distribution function, quantile function and random generation for exponential generalized beta of the second type distribution.

Usage

```
dEGB(y_star, mu, phi, log = FALSE)

pEGB(q, mu, phi)

qEGB(p, mu, phi)

rEGB(n, mu, phi)
```

Arguments

y_star, q	vector of quantiles.
mu	mu parameter.
phi	phi parameter.
log	logical; if TRUE, probabilities p are given as log(p). Default is FALSE.
p	vector of probabilities.
n	number of observations. If length(n) > 1, the length is taken to be the number required.

Details

The EGB distribution with parameters mu = μ and phi = ϕ has density

$$f(y^\star;\mu,\phi) = B^{-1}(\mu\phi,(1-\mu)\phi)\exp\{-y^\star(1-\mu)\phi\}/(1+\exp\{-y^\star\})^\phi,$$

with $\mu \in (0,1), \phi > 0$ and $y^\star \in (-\infty,\infty)$. For this distribution, $E(y^\star) = \psi(\mu\phi) - \psi((1-\mu)\phi)$ and $Var(y^\star) = \psi'(\mu\phi) + \psi'((1-\mu)\phi)$, where ψ is the digamma function. See Kerman and McDonald (2015) for additional details. If $y \sim beta(\mu,\phi)$, with μ and ϕ representing the mean and precision of y, then $y^\star = \log(y/(1-y)) \sim EGB(\mu,\phi)$ with the density given above.

Value

dEGB gives the density, pEGB gives the distribution function, qEGB gives the quantile function, and rEGB generates random variables.

Author(s)

Yuri S. Maluf (<yurimaluf@gmail.com>), Francisco F. Queiroz (<ffelipeq@outlook.com>) and Silvia L. P. Ferrari.

References

Maluf, Y.S., Ferrari, S.L.P., and Queiroz, F.F. (2022). Robust beta regression through the logit transformation. *Metrika*:61–81.

Kerman, S. and McDonald, J.B. (2015). Skewness-kurtosis bounds for EGB1, EGB2, and special cases. *Communications in Statistics - Theory and Methods*, 44:3857-3864.

Examples

```
dEGB(0.2, mu = 0.3, phi = 1)
mu = 0.2; phi = 2;
set.seed(1)
EGBsample = rEGB(1000, mu, phi)
hist(EGBsample, prob = TRUE, breaks = 15, main = "", las = 1, ylim = c(0, 0.2),
     xlim = c(-20, 10))
curve(dEGB(x, mu, phi), from = -20, to = 8, add = TRUE, col = "red")


# Showing the P(Y* < -5) = 0.17, where Y* ~ EGB(0.2, 2).
x = seq(-20, 10,0.01)
y = dEGB(x, mu, phi)
plot(x, y, type = "l", lwd = 2, las = 1)
x1 = seq(-20, -5, 0.01)
y1 = dEGB(x1, mu, phi)
polygon(c(x1, -5, -5), c(y1, 0, 0), col = "lightblue")


plot(x, pEGB(x, mu, phi), type = "l", las = 1, lwd = 2,
     ylab = expression(P("Y*"<y)), xlab = "y")
p = pEGB(0, mu, phi)
```

```
q = qEGB(p, mu, phi)
points(q, p, pch = 16, col = 2, cex = 1.5)
text(2, 0.83, paste("(", 0, ",", round(p, 2), ")"), font = 2,
     cex = 0.8, col = "red")
```

`Firm` *Firm Cost*

Description

A dataset on risk management practices of 73 firms.

Usage

```
data("Firm", package = "robustbetareg")
```

Format

A data frame with 73 rows and 7 variables:

FIRMCOST total property and casualty premiums and uninsured losses as a percentage of total assets.

ASSUME per occurrence retention amount as a percentage of total assets.

CAP indicates that the firm owns a captive insurance company; 1 if the firm uses a captive, 0 otherwise.

SIZELOG logarithm of total assets.

INDCOST a measure of the firm's industry risk.

CENTRAL a measure of the importance of the local managers in choosing the amount of risk to be retained.

SOPH a measure of the degree of importance in using analytical tools.

Details

The dataset was introduced and analyzed by Schmit and Roth (1990) and is available in the personal web page of Professor E. Frees (Wisconsin School of Business Research). The response variable is `FIRMCOST`, smaller values of firm cost are attributed to firms that have a good risk management performance.

Source

https://instruction.bus.wisc.edu/jfrees/jfreesbooks/Regression%20Modeling/BookWebDec2010/CSVData/RiskSurvey.csv

References

Schmit, J.T. and Roth, K. (1990). Cost effectiveness of risk management practices. *Journal of Risk and Insurance*. 57:455-470.

HIC *Health Insurance Coverage*

Description

A dataset containing the proportion of the population from several cities of the state of São Paulo, Brazil, in 2010.

Usage

```
data("HIC", package = "robustbetareg")
```

Format

A data frame with 80 rows and 4 variables:

CITY the corresponding city.

URB proportion of the total population living in the city's urban zone.

GDP per capita gross domestic product.

HIC the health insurance coverage index.

Details

This dataset was collected by the Institute of Applied Economic Research (Instituto de Pesquisa Econômica Aplicada, IPEA, Brazil). It includes information on 80 cities in the state of São Paulo, Brazil, in 2010.

Source

http://www.ipeadata.gov.br/Default.aspx

methodsrobustbetareg *Methods for robustbetareg Objects*

Description

Some S3 methods for objects of class "robustbetareg".

Usage

```
## S3 method for class 'robustbetareg'
summary(object, type = "sweighted2", ...)

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

## S3 method for class 'summary.robustbetareg'
print(x, digits = max(3, getOption("digits") - 3), ...)
```

Arguments

object, x	fitted model of class robustbetareg.
type	character specifying the type of residuals to be included in the summary output, see residuals.robustbetareg.
...	currently not used.
model	character specifying for which component of the model the coefficients should be extracted.
digits	the number of significant digits to use when printing.

Details

A set of methods for fitted model objects of class robustbetareg, including methods to the generic functions print and summary, which print the estimated coefficients along with some further information.

Value

methodsrobustbetareg returns different outputs for objects of class robustbetareg, depending on the used method.

See Also

robustbetareg

Examples

```
data("HIC", package="robustbetareg")
fit=robustbetareg(HIC~URB+GDP|1,data=HIC,alpha=0.06)
summary(fit)
coef(fit)
```

---

plot.robustbetareg *Diagnostic Plots for robustbetareg Objects*

Description

Several types of standard diagnostic plots can be produced interactively, involving different types of residuals.

Usage

```
## S3 method for class 'robustbetareg'
plot(x, ask = TRUE, ...)
```

Arguments

x	fitted model object of class "robustbetareg".
ask	logical. If "TRUE" the user is asked before each plot.
...	graphical parameters passed to the plot function (see par).

Value

plot method for robustbetareg objects returns several diagnostic plots.

See Also

robustbetareg, residuals.robustbetareg, plotenvelope

Examples

```
get(data("HIC", package = "robustbetareg"))
hic <- robustbetareg(HIC ~ URB + GDP | GDP,
                     data = HIC, alpha = 0.06)
```

---

plotenvelope *Normal Probability Plots of Residuals with Simulated Envelope for robustbetareg Objects*

Description

plotenvelope is used to display normal probability plots of residuals with simulated envelope for the robust beta regression. Currently, eight types of residuals are supported: sweighted2, pearson, weighted, sweighted, sweighted.gamma, sweighted2.gamma, combined, and combined.projection residuals.

Usage

```
plotenvelope(
  object,
  type = c("sweighted2", "pearson", "weighted", "sweighted", "sweighted.gamma",
    "sweighted2.gamma", "combined", "combined.projection"),
  conf = 0.95,
  n.sim = 100,
  PrgBar = TRUE,
  control = robustbetareg.control(...),
  ...
)
```

Arguments

object	fitted model object of class robustbetareg.
type	character indicating the type of residuals to be used, see residuals.robustbetareg. Default is type = "sweighted2".
conf	numeric specifying the confidence level of the simulated envelopes. Default is conf = 0.95.
n.sim	a positive integer representing the number of iterations to generate the simulated envelopes. Default is n.sim = 100.
PrgBar	logical. If PrgBar = TRUE the progress bar will be shown in the console. Default is PrgBar = TRUE.
control	a list of control arguments specified via robustbetareg.control.
...	arguments passed to plot.

Details

The plotenvelope creates normal probability plots with simulated envelope (see Atkinson (1985) for details). Under the correct model, approximately 100*conf of the residuals are expected to be inside the envelope.

Value

plotenvelope returns normal probability plot of residuals with simulated envelope.

Author(s)

Yuri S. Maluf (<yurimaluf@gmail.com>), Francisco F. Queiroz (<ffelipeq@outlook.com>) and Silvia L. P. Ferrari.

References

Maluf, Y.S., Ferrari, S.L.P., and Queiroz, F.F. (2022). Robust beta regression through the logit transformation. *Metrika*:61–81.

Atkinson, A.C. (1985) Plots, transformations and regression: an introduction to graphical methods of diagnostic regression analysis. *Oxford Science Publications*, Oxford.

See Also

robustbetareg, robustbetareg.control, residuals.robustbetareg

Examples

```
get(data("HIC", package = "robustbetareg"))
hic <- robustbetareg(HIC ~ URB + GDP | GDP,
data = HIC, alpha = 0.06)
plotenvelope(hic, n.sim = 50)

get(data("Firm", package = "robustbetareg"))
```

```
rmc <- robustbetareg(FIRMCOST ~ INDCOST + SIZELOG | INDCOST + SIZELOG, data = Firm)
plotenvelope(rmc, conf = 0.90)
```

---

predict *Prediction Methods for robustbetareg Objects Class*

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, from `robustbetareg` objects.

Usage

```
predict(
  object,
  newdata = NULL,
  type = c("response", "link", "precision", "variance", "quantile"),
  at = 0.5,
  ...
)
```

Arguments

`object`	fitted model object of class "`robustbetareg`".
`newdata`	optional, a data frame with new predictor values. If omitted, the original predictors 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`").
`at`	numeric vector indicating the level(s) at which quantiles should be predicted (only if `type = "quantile"`). Default is the median `at = 0.5`.
`...`	currently not used.

Value

Return a vector with the predicted values.

Examples

```
get(data("HIC", package = "robustbetareg"))
hic <- robustbetareg(HIC ~ URB + GDP | 1, data = HIC, alpha = 0.04)
cbind(predict(hic, type = "response"), predict(hic, type = "quantile", at = c(0.25, 0.5, 0.75)))
```

`residuals.robustbetareg` *Residuals Method for robustbetareg Objects*

Description

The function provides several types of residuals for the robust beta regression models: Pearson residuals (raw residuals scaled by square root of variance function) and different kinds of weighted residuals proposed by Espinheira et al. (2008) and Espinheira et al. (2017).

Usage

```
## S3 method for class 'robustbetareg'
residuals(
  object,
  type = c("sweighted2", "pearson", "weighted", "sweighted", "sweighted.gamma",
    "sweighted2.gamma", "combined", "combined.projection"),
  ...
)
```

Arguments

`object`	fitted model object of class `robustbetareg`.
`type`	character indicating type of residuals to be used.
`...`	currently not used.

Details

The definitions of the first four residuals are provided in Espinheira et al. (2008): equation (2) for "`pearson`", equation (6) for "`weighted`", equation (7) for "`sweighted`", and equation (8) for "`sweighted2`". For the last four residuals the definitions are described in Espinheira et al. (2017): equations (7) and (10) for the "`sweighted.gamma`" and "`sweighted2.gamma`", respectively, equation (9) for "`combined`", and equation (11) for "`combined.projection`".

Value

`residuals` returns a vector with the residuals of the type specified in the `type` argument.

References

Maluf, Y.S., Ferrari, S.L.P., and Queiroz, F.F. (2022). Robust beta regression through the logit transformation. *Metrika*:61–81.

Espinheira, P.L., Ferrari, S.L.P., and Cribari-Neto, F. (2008). On Beta Regression Residuals. *Journal of Applied Statistics*, 35:407–419.

Espinheira, P.L., Santos, E.G.and Cribari-Neto, F. (2017). On nonlinear beta regression residuals. *Biometrical Journal*, 59:445-461.

See Also

robustbetareg

Examples

```
get(data("HIC", package = "robustbetareg"))
fit.hic <- robustbetareg(HIC ~ URB + GDP | 1,
                         data = HIC, alpha = 0.04)
res <- residuals(fit.hic, type = "sweighted2")
#plot(res)
#abline(h = 0)
```

---

robustbetareg *Robust Beta Regression*

Description

Fit robust beta regression models for rates and proportions via LSMLE, LMDPDE, SMLE and MDPDE. Both mean and precision of the response variable are modeled through parametric functions.

Usage

```
robustbetareg(
  formula,
  data,
  alpha,
  type = c("LSMLE", "LMDPDE", "SMLE", "MDPDE"),
  link = c("logit", "probit", "cloglog", "cauchit", "loglog"),
  link.phi = NULL,
  control = robustbetareg.control(...),
  model = TRUE,
  ...
)

LMDPDE.fit(y, x, z, alpha = NULL, link = "logit",
link.phi = "log", control = robustbetareg.control(...), ...)

LSMLE.fit(y, x, z, alpha = NULL, link = "logit",
link.phi = "log", control = robustbetareg.control(...), ...)
```

```
MDPDE.fit(y, x, z, alpha = NULL, link = "logit",
link.phi = "log", control = robustbetareg.control(...), ...)

SMLE.fit(y, x, z, alpha = NULL, link = "logit",
link.phi = "log", control = robustbetareg.control(...), ...)
```

Arguments

formula	symbolic description of the model. See Details for further information.
data	dataset to be used.
alpha	numeric in $[0, 1)$ indicating the value of the tuning constant alpha. alpha = 0 leads to the maximum likelihood estimator. Robust procedures require alpha greater than zero. If this argument is suppressed, the tuning constant will be selected automatically through the data-driven algorithm proposed by Ribeiro and Ferrari (2022).
type	character specifying the type of robust estimator to be used in the estimation process. Supported estimators are "LSMLE" , "LMDPDE", "SMLE", and "MDPDE"; for details, see Maluf et al. (2022). The "LSMLE" is the default.
link	an optional character that specifies the link function of the mean submodel (mu). The "logit", "probit", "cloglog", "cauchit", "loglog" functions are supported. The logit function is the default.
link.phi	an optional character that specifies the link function of the precision submodel (phi). The "identity", "log", "sqrt" functions are supported. The default is log unless formula is of type y ~ x where the default is "identity".
control	a list of control arguments specified via robustbetareg.control.
model	logical. If TRUE the corresponding components of the fit (model frame, response, model matrix) are returned.
...	argument to be passed to robustbetareg.control.
y, x, z	y must be a numeric response vector (with values in $(0, 1)$), x must be a numeric regressor matrix for the mean submodel, and z must be a numeric regressor matrix for the precision submodel.

Details

Beta regression models are employed to model continuous response variables in the unit interval, like rates and proportions. The maximum likelihood-based inference suffers from the lack of robustness in the presence of outliers. Based on the density power divergence, Ghosh (2019) proposed the minimum density power divergence estimator (MDPDE). Ribeiro and Ferrari (2022) proposed an estimator based on the maximization of a reparameterized Lq-likelihood; it is called SMLE. These estimators require suitable restrictions in the parameter space. Maluf et al. (2022) proposed robust estimators based on the MDPDE and the SMLE which have the advantage of overcoming this drawback. These estimators are called LMDPDE and LSMLE. For details, see the cited works. The four estimators are implemented in the robustbetareg function. They depend on a tuning constant (called α). When the tuning constant is fixed and equal to 0, all of the estimators coincide with the maximum likelihood estimator. Ribeiro and Ferrari (2022) and Maluf et al. (2022) suggest using a data-driven algorithm to select the optimum value of α. This algorithm is implemented in

robustbetareg by default when the argument "alpha" is suppressed.

The formulation of the model has the same structure as in the usual functions glm and betareg. The argument formula can comprise of three parts (separated by the symbols " " and "|"), namely: observed response variable in the unit interval, predictor of the mean submodel, with link function link and predictor of the precision submodel, with link.phi link function. If the model has constant precision, the third part may be omitted and the link function for phi is "identity" by default. The tuning constant alpha may be treated as fixed or not (chosen by the data-driven algorithm). If alpha is fixed, its value must be specified in the alpha argument.

Some methods are available for objects of class "robustbetareg", see plot.robustbetareg, summary.robustbetareg, coef.robustbetareg, and residuals.robustbetareg, for details and other methods.

Value

robustbetareg returns an object of class "robustbetareg" with a list of the following components:

coefficients	a list with the "mean" and "precision" coefficients.
vcov	covariance matrix.
converged	logical indicating successful convergence of the iterative process.
fitted.values	a vector with the fitted values of the mean submodel.
start	a vector with the starting values used in the iterative process.
weights	the weights of each observation in the estimation process.
Tuning	value of the tuning constant (automatically chosen or fixed) used in the estimation process.
residuals	a vector of standardized weighted residual 2 (see Espinheira et al. (2008)).
n	number of observations.
link	link function used in the mean submodel.
link.phi	link function used in the precision submodel.
Optimal.Tuning	logical indicating whether the data-driven algorithm was used.
pseudo.r.squared	pseudo R-squared value.
control	the control arguments passed to the data-driven algorithm and optim call.
std.error	the standard errors.

method	type of estimator used.
call	the original function call.
formula	the formula used.
model	the full model frame.
terms	a list with elements "mean", "precision" and "full" containing the term objects for the respective mo
y	the response variable.
data	the dataset used.

Author(s)

Yuri S. Maluf (<yurimaluf@gmail.com>), Francisco F. Queiroz (<ffelipeq@outlook.com>) and Silvia L. P. Ferrari.

References

Maluf, Y.S., Ferrari, S.L.P., and Queiroz, F.F. (2022). Robust beta regression through the logit transformation. *Metrika*:61–81.

Ribeiro, T.K.A. and Ferrari, S.L.P. (2022). Robust estimation in beta regression via maximum Lq-likelihood. *Statistical Papers*. DOI: 10.1007/s00362-022-01320-0.

Ghosh, A. (2019). Robust inference under the beta regression model with application to health care studies. *Statistical Methods in Medical Research*, 28:271-888.

Espinheira, P.L., Ferrari, S.L.P., and Cribari-Neto, F. (2008). On beta regression residuals. *Journal of Applied Statistics*, 35:407–419.

See Also

robustbetareg.control, summary.robustbetareg, residuals.robustbetareg

Examples

```
#### Risk Manager Cost data
data("Firm")

# MLE fit (fixed alpha equal to zero)
fit_MLE <- robustbetareg(FIRMCOST ~ SIZELOG + INDCOST,
                         data = Firm, type = "LMDPDE", alpha = 0)
summary(fit_MLE)

# MDPDE with alpha = 0.04
fit_MDPDE <- robustbetareg(FIRMCOST ~ SIZELOG + INDCOST,
```

```
                            data = Firm, type = "MDPDE",
                            alpha = 0.04)
summary(fit_MDPDE)

# Choosing alpha via data-driven algorithm
fit_MDPDE2 <- robustbetareg(FIRMCOST ~ SIZELOG + INDCOST,
                             data = Firm, type = "MDPDE")
summary(fit_MDPDE2)

# Similar result for the LMDPDE fit:
fit_LMDPDE2 <- robustbetareg(FIRMCOST ~ SIZELOG + INDCOST,
                              data = Firm, type = "LMDPDE")
summary(fit_LMDPDE2)

# Diagnostic plots


#### HIC data
data("HIC")

# MLE (fixed alpha equal to zero)
fit_MLE <- robustbetareg(HIC ~ URB + GDP |
                         GDP, data = HIC, type = "LMDPDE",
                         alpha = 0)
summary(fit_MLE)

# SMLE and MDPDE with alpha selected via data-driven algorithm
fit_SMLE <- robustbetareg(HIC ~ URB + GDP |
                           GDP, data = HIC, type = "SMLE")
summary(fit_SMLE)
fit_MDPDE <- robustbetareg(HIC ~ URB + GDP |
                           GDP, data = HIC, type = "MDPDE")
summary(fit_MDPDE)
# SMLE and MDPDE return MLE because of the lack of stability

# LSMLE and LMDPDE with alpha selected via data-driven algorithm
fit_LSMLE <- robustbetareg(HIC ~ URB + GDP |
                           GDP, data = HIC, type = "LSMLE")
summary(fit_LSMLE)
fit_LMDPDE <- robustbetareg(HIC ~ URB + GDP |
                            GDP, data = HIC, type = "LMDPDE")
summary(fit_LMDPDE)
# LSMLE and LMDPDE return robust estimates with alpha = 0.06


# Plotting the weights against the residuals - LSMLE fit.
plot(fit_LSMLE$residuals, fit_LSMLE$weights, pch = "+", xlab = "Residuals",
    ylab = "Weights")

# Excluding outlier observation.
fit_LSMLEwo1 <- robustbetareg(HIC ~ URB + GDP |
                              GDP, data = HIC[-1,], type = "LSMLE")
summary(fit_LSMLEwo1)
```

```
# Normal probability plot with simulated envelope
 plotenvelope(fit_LSMLE)
```

`robustbetareg.control` *Auxiliary for Controlling robustbetareg Fitting*

Description

Several parameters that control fitting of robust beta regression models using `robustbetareg.`

Usage

```
robustbetareg.control(
  start = NULL,
  alpha.optimal = TRUE,
  tolerance = 0.001,
  maxit = 5000,
  L = 0.02,
  M = 3,
  ...
)
```

Arguments

`start`	an optional vector with starting values for the parameter estimates.
`alpha.optimal`	a logical value. If `TRUE` the tuning constant will be select via the data-driven algorithm.
`tolerance`	numeric tolerance for convergence.
`maxit`	argument passed to `optim`.
`L`	numeric specifying the threshold for the data-driven algorithm. Default is `L = 0.02`.
`M`	integer specifying the number of grid spacing for the data-driven algorithm. Default is `M = 3`.
`...`	currently not used.

Details

The `robustbetareg.control` controls the fitting process of the robust estimation in beta regression via the LMDPDE, LSMLE, MDPDE, and SMLE. The arguments `L` and `M` are passed to the data-driven algorithm for selecting the optimum alpha value; details can be found in Ribeiro and Ferrari (2022). Starting values for the parameters associated to the mean and precision submodels may be supplied via `start`.
We do not recommend changing the arguments passed to the data-driven algorithm.

Value

A list with components named as the arguments.

Author(s)

Yuri S. Maluf (<yurimaluf@gmail.com>), Francisco F. Queiroz (<ffelipeq@outlook.com>) and Silvia L. P. Ferrari.

References

Maluf, Y.S., Ferrari, S.L.P., and Queiroz, F.F. (2022). Robust beta regression through the logit transformation. *Metrika*:61–81.

Ribeiro, K.A.T. and Ferrari, S.L.P. (2022). Robust estimation in beta regression via maximum Lq-likelihood. *Statistical Papers*. DOI: 10.1007/s00362-022-01320-0.

See Also

robustbetareg

Examples

```
data("Firm")

# Using a robust start value for the parameters associated with the
# mean submodel
# using the robustbase package
# robust regression to obtain a starting value for beta
fit_lm_Rob <- robustbase::lmrob(FIRMCOST ~ SIZELOG + INDCOST,
                                data = Firm)
initials_beta_rob <-  as.numeric(coef(fit_lm_Rob))
etarob <- model.matrix(fit_lm_Rob)%*%initials_beta_rob
muhat_Rob <- set.link(link.mu = "logit",
                      link.phi = "log")$linkfun.mu$inv.link(etarob)
T_1_Rob <- 1/set.link(link.mu = "logit",
                      link.phi = "log")$linkfun.mu$d.linkfun(muhat_Rob)
#estimate of variance of y based on robustbase package
sigma2hat_Rob <- ((fit_lm_Rob$scale^2)*(T_1_Rob^2))
#phi estimate from robust method
phihat_Rob <- mean((muhat_Rob*(1-muhat_Rob))/sigma2hat_Rob)
gama1hat_rob <- log(phihat_Rob)
#gamma estimates from robustbase package
initials_gama_rob <-  as.numeric(gama1hat_rob)
#robust starting values for beta and gamma
thetaStart <- c(initials_beta_rob, initials_gama_rob)

fit_LSMLE <- robustbetareg(FIRMCOST ~ SIZELOG + INDCOST,
                           data = Firm,
                           type = "LSMLE", link.phi = "log",
```

```
                    control = robustbetareg.control(start = thetaStart))
```

`set.link` *Link functions for robust beta regression*

Description

This function provides several link functions for robust beta regression.

Usage

```
set.link(link.mu = "logit", link.phi = "log")
```

Arguments

`link.mu`	character specifying the mean link function. Currently, the functions "`logit`", "`probit`", "`cauchit`", "`cloglog`", and "`loglog`" are supported. Default is "`logit`".
`link.phi`	character specifying the precision link function. Currently, the functions "`log`", "`identity`", and "`sqrt`" are supported. Default is "`log`".

Value

`set.link` provides the link function, inverse link function, first and second derivatives for both mean and precision submodels.

Examples

```
links = set.link(link.mu = "cauchit", link.phi = "sqrt")
attributes(links)
```

`waldtypetest` *Robust Wald-type Tests*

Description

`waldtypetest` provides Wald-type tests for both simple and composite hypotheses for beta regression based on the robust estimators (LSMLE, LMDPDE, SMLE, and MDPDE).

Usage

```
waldtypetest(object, FUN, ...)
```

Arguments

`object`	fitted model object of class `robustbetareg` (see `robustbetareg`).
`FUN`	function representing the null hypothesis to be tested.
`...`	further arguments to be passed to the `FUN` function.

Details

The function provides a robust Wald-type test for a general hypothesis $m(\theta) = \eta_0$, for a fixed $\eta_0 \in R^d$, against a two sided alternative; see Maluf et al. (2022) for details. The argument `FUN` specifies the function $m(\theta) - \eta_0$, which defines the null hypothesis. For instance, consider a model with $\theta = (\beta_1, \beta_2, \beta_3, \gamma_1)^\top$ and let the null hypothesis be $\beta_2 + \beta_3 = 4$. The `FUN` argument can be `FUN = function(theta) {theta[2] + theta[3] - 4}`. It is also possible to define the function as `FUN = function(theta, B) {theta[2] + theta[3] - B}`, and pass the `B` argument through the `waldtypetest` function. If no function is specified, that is, `FUN=NULL`, the `waldtypetest` returns a test in which the null hypothesis is that all the coefficients are zero.

Value

`waldtypetest` returns an output for the Wald-type test containing the value of the test statistic, degrees-of-freedom and p-value.

Author(s)

Yuri S. Maluf (<yurimaluf@gmail.com>), Francisco F. Queiroz (<ffelipeq@outlook.com>) and Silvia L. P. Ferrari.

References

Maluf, Y.S., Ferrari, S.L.P., and Queiroz, F.F. (2022). Robust beta regression through the logit transformation. *Metrika*:61–81.

Basu, A., Ghosh, A., Martin, N., and Pardo, L. (2018). Robust Wald-type tests for non-homogeneous observations based on the minimum density power divergence estimator. *Metrika*, 81:493–522.

Ribeiro, K. A. T. and Ferrari, S. L. P. (2022). Robust estimation in beta regression via maximum Lq-likelihood. *Statistical Papers*.

See Also

`robustbetareg`

Examples

```
# generating a dataset
set.seed(2022)
n <- 40
beta.coef <- c(-1, -2)
gamma.coef <- c(5)
X <- cbind(rep(1, n), x <- runif(n))
```

```
mu <- exp(X%*%beta.coef)/(1 + exp(X%*%beta.coef))
phi <- exp(gamma.coef) #Inverse Log Link Function
y <- rbeta(n, mu*phi, (1 - mu)*phi)
y[26] <- rbeta(1, ((1 + mu[26])/2)*phi, (1 - ((1 + mu[26])/2))*phi)
SimData <- as.data.frame(cbind(y, x))
colnames(SimData) <- c("y", "x")

# Fitting the MLE and the LSMLE
fit.mle <- robustbetareg(y ~ x | 1, data = SimData, alpha = 0)
fit.lsmle <- robustbetareg(y ~ x | 1, data = SimData)

# Hypothesis to be tested: (beta_1, beta_2) = c(-1, -2) against a two
# sided alternative
h0 <- function(theta){theta[1:2] - c(-1, -2)}
waldtypetest(fit.mle, h0)
waldtypetest(fit.lsmle, h0)
# Alternative way:
h0 <- function(theta, B){theta[1:2] - B}
waldtypetest(fit.mle, h0, B = c(-1, -2))
waldtypetest(fit.lsmle, h0, B = c(-1, -2))
```

Index