Package 'cdfquantreg'

November 22, 2016

Type Package

Title Quantile Regression for Random Variables on the Unit Interval

Version 1.1.0

Date 2016-11-20

Description Employs a two-parameter family of distributions for modelling random variables on the (0, 1) interval by applying the cumulative distribution function (cdf) of one parent distribution to the quantile function of another.

BugReports https://cloudyshou.wordpress.com/cdfquantreg-bugs-report

Depends R (>= 3.1.0)

License GPL-3

Imports pracma (>= 1.8), Formula (>= 1.2), stats, MASS

Suggests knitr, R2OpenBUGS

VignetteBuilder knitr

LazyData true

RoxygenNote 5.0.1

NeedsCompilation no

Author Yiyun Shou [aut, cre], Michael Smithson [aut]

Maintainer Yiyun Shou <yiyun.shou@anu.edu.au>

Repository CRAN

Date/Publication 2016-11-22 09:07:02

R topics documented:

`cdfquantreg-package` *Quantile Regression for Random Variables on the Unit Interval*

Description

Employs a two-parameter family of distributions for modelling random variables on the (0, 1) interval by applying the cumulative distribution function (cdf) of one parent distribution to the quantile function of another.

Details

Package:	cdfquantreg
Type:	Package
Version:	1.1.0
Date:	2016-09-10
License:	GPL-3

The cdfquantreg package includes 36 members of a two-parameter family of distributions for modelling random variables on the (0, 1) interval (see cdfqrFamily). This family has explicit pdfs, cdfs, and quantile functions. The two parameters consist of a location parameter and a dispersion parameter. The location parameter models the median and the dispersion parameter models the spread of other quantiles around the median (see Smithson and Shou, 2016, for details about the distribution family and the models). Separate submodels may be specified for the location and for the dispersion parameters, permitting different or overlapping sets of predictors in each.

The package offers maximum likelihood (see cdfquantreg), Bayesian MCMC (see bugsModel), and bootstrap (see qrBoot) estimation methods. All model functions return S3 objects. The bugsModel() function runs OpenBUGS in R. In addition to the usual goodness of fit information, the package provides root-mean-squared errors in both the raw and logit scales, and the gradient. Model diagnostics include raw, Pearson, and deviance residuals (see residuals.cdfqr), and dfbetas (see influence.cdfqr).

For each distribution, the package provides evaluations of the pdf (dq), cdf (pq), and quantile (qq), as well as random samples from any of them (rq). Evaluations of skew and kurtosis (qrPwlm) also are available using probability-weighted L-moments.

Author(s)

Yiyun Shou (<yiyun.shou@@anu.edu.au>) and Michael Smithson (<Michael.Smithson@@anu.edu.au>)

Maintainer: Yiyun Shou

References

Smithson, M. and Shou, Y. (2016). CDF-Quantile Distributions for Modeling Random Variables on the Unit Interval. Unpublished Manuscript, The Australian National University, Canberra, Australia.

See Also

cdfqrFamily

---

anova.cdfqr *Model comparison test for fitted cdfqr models*

Description

Likelihood Ratio Tests for fitted cdfqr Objects.

Usage

```
## S3 method for class 'cdfqr'
anova(object, ..., test = "LRT")
```

Arguments

object	The fitted cdfqr model.
...	One or more cdfqr model objects for model comparison.
test	The model comparision test, currently only 'LRT' is implemented.

Examples

```
data(cdfqrExampleData)
fit_null <- cdfquantreg(crc99 ~ 1 | 1, 't2','t2', data = JurorData)
fit_mod1 <- cdfquantreg(crc99 ~ vert | confl, 't2','t2', data = JurorData)
anova(fit_null, fit_mod1)
```

`AnxStrData` *Stress-Anxiety data*

Description

A second example is a data from a study that investigates the relationship between stress and anxiety..

Usage

```
AnxStrData
```

Format

A data frame with 166 rows and 2 variables:

Anxiety Scores on Anxiety subscale

Stress Scores on Stress subscale

Source

https://dl.dropboxusercontent.com/u/1857674/betareg/betareg.html

---

`bugsLikelihood` *Likelihood Functions for Generating OpenBUGS Model File*

Description

Likelihood functions for generating OpenBUGS model file.

Usage

```
bugsLikelihood(fd, sd)
```

Arguments

`fd`	A string that specifies the parent distribution.
`sd`	A string that specifies the sub-family distribution.

Value

A string to be written in the BUGS model file.

See Also

qrBugs

Examples

```
bugsLikelihood('t2','t2')
```

`bugsModel` *Generating OpenBUGS Model File*

Description

Generating OpenBUGS model file

Usage

```
bugsModel(formula, fd, sd, random = NULL, modelname = "bugmodel",
  wd = getwd())
```

Arguments

`formula`	A formula object, with the DV on the left of an ~ operator, and predictors on the right. For the part on the right of '~', the specification of the submodels can be separated by 'l'. So `y ~ X1 | X2` means the DV is `y`,`X1` is the term in the mean submodel, and `X2` is the term in the dispersion submodel.
`fd`	A string that specifies the parent distribution (see cdfqrFamily).
`sd`	A string that specifies the sub-family distribution.
`random`	Character or vector of characters that indicates the random effect factors.
`modelname`	The name of the model file; optional.
`wd`	The working directory in which OpenBUGS will work (i.e., generate the model files and chain information).

Value

A model '.txt' file is generated in the specified working directory. The function also returns a list of values:

init1,init2 Default initial values for MCMC two chain procedure.

vars A list of variables that are included in the estimation.

nodes_sample a list of characters that specify the nodes to be monitored.

See Also

qrBugs

Examples

```
## Not run:
# Need write access in the working directory before executing the code.
# No random component
bugsModel(y ~ x1 | x2, 't2','t2', random = NULL)
# Random component as subject ID
bugsModel(y ~ x1 | x2, 't2','t2', random = 'ID')

## End(Not run)
```

`cdfqr.control` *Control Optimization Parameters for CDF-Quantile Probability Distributions*

Description

Control Optimization Parameters for CDF-Quantile Probability Distributions.

Usage

```
cdfqr.control(method = "BFGS", maxit = 5000, hessian = TRUE,
  trace = FALSE)
```

Arguments

`method`	Characters string specifying the method argument passed to optim.
`maxit`	Integer specifying the maxit argument (maximal number of iterations) passed to optim.
`hessian`	logical value, which indicates whether the numerical Hessian matrix from the optim output be used for estimation of the covariance matrix. Currently, only 'TRUE' is allowed, as only the analytical solution is employed.
`trace`	Logical or integer controlling whether tracing information on the progress of the optimization should be produced

Value

A list with the arguments specified.

Examples

```
data(cdfqrExampleData)
fit <- cdfquantreg(crc99 ~ vert | confl, 't2', 't2',
data = JurorData,control = cdfqr.control(trace = TRUE))
```

cdfqrFamily *Overview of the family of distributions*

Description

The cdfquantreg family consists of the currently available distributions that can be used to fit quantile regression models via the cdfquantreg() function.

Usage

```
cdfqrFamily(shape = "all")
```

Arguments

shape	To show all distributions or the set of distribution for a specific type of shape. Can be BM, TM,LL or FT for Bimodal, Trimodal, Logit-logistic or Finite-tailed shapes, respectively.

Details

The cdfquantreg package includes a two-parameter family of distributions for modeling random variables on the (0, 1) interval by applying the cumulative distribution function (cdf) of one "parent" distribution to the quantile function of another.
The naming of these distributions is "parent - child" or "fd - sd", where "fd" is the parent distribution, and "sd" is the child distribution.
The distributions have four characteristic shapes: Logit-logistic, bimodal, trimodal, and finite-tailed. Here is the list of currently available distributions.

Bimodal Shape Distributions

Distribution	R input	Alternative Input	Shape
Burr VII-ArcSinh	`fd = "burr7", sd = "arcsinh"`	`family = "burr7-arcsinh"`	Bimodal
Burr VII-Cauchy	`fd = "burr7", sd = "cauchy"`	`family = "burr7-cauchy"`	Bimodal
Burr VII-T2	`fd = "burr7", sd = "t2"`	`family = "burr7-t2"`	Bimodal
Burr VIII-ArcSinh	`fd = "burr8", sd = "arcsinh"`	`family = "burr8-arcsinh"`	Bimodal
Burr VIII-Cauchy	`fd = "burr8", sd = "cauchy"`	`family = "burr8-cauchy"`	Bimodal
Burr VIII-T2	`fd = "burr8", sd = "t2"`	`family = "burr8-t2"`	Bimodal
Logit-ArcSinh	`fd = "logit", sd = "arcsinh"`	`family = "logit-arcsinh"`	Bimodal
Logit-Cauchy	`fd = "logit", sd = "cauchy"`	`family = "logit-cauchy"`	Bimodal
Logit-T2	`fd = "logit", sd = "t2"`	`family = "logit-t2"`	Bimodal
T2-ArcSinh	`fd = "t2", sd = "arcsinh"`	`family = "t2-arcsinh"`	Bimodal
T2-Cauchy	`fd = "t2", sd = "cauchy"`	`family = "t2-cauchy"`	Bimodal

Trimodal Shape Distributions

Distribution	R input	Alternative Input	Shape
ArcSinh-Burr VII	`fd = "arcsinh", sd = "burr7"`	`family = "arcsinh-burr7"`	Trimodal

ArcSinh-Burr VIII	`fd = "arcsinh", sd = "burr8"`	`family = "arcsinh-burr8"`	Trimodal
ArcSinh-Logistic	`fd = "arcsinh", sd = "logistic"`	`family = "arcsinh-logistic"`	Trimodal
ArcSinh-T2	`fd = "arcsinh", sd = "t2"`	`family = "arcsinh-t2"`	Trimodal
Cauchit-Burr VII	`fd = "cauchit", sd = "burr7"`	`family = "cauchit-burr7"`	Trimodal
Cauchit-Burr VIII	`fd = "cauchit", sd = "burr8"`	`family = "cauchit-burr8"`	Trimodal
Cauchit-Logistic	`fd = "cauchit", sd = "logistic"`	`family = "cauchit-logistic"`	Trimodal
Cauchit-T2	`fd = "cauchit", sd = "t2"`	`family = "cauchit-t2"`	Trimodal
T2-Burr VII	`fd = "t2", sd = "burr7"`	`family = "t2-burr7"`	Trimodal
T2-Burr VIII	`fd = "t2", sd = "burr8"`	`family = "t2-burr8"`	Trimodal
T2-Logistic	`fd = "t2", sd = "logistic"`	`family = "t2-logistic"`	Trimodal

Logit-logistic Shape Distributions

Distribution	R input	Alternative Input	Shape
Burr VII-Burr VII	`fd = "burr7", sd = "burr7"`	`family = "burr7-burr7"`	Logit-logistic
Burr VII-Burr VIII	`fd = "burr7", sd = "burr8"`	`family = "burr7-burr8"`	Logit-logistic
Burr VII-Logistic	`fd = "burr7", sd = "logistic"`	`family = "burr7-logistic"`	Logit-logistic
Burr VIII-Burr VII	`fd = "burr8", sd = "burr7"`	`family = "burr8-burr7"`	Logit-logistic
Burr VIII-Burr VIII	`fd = "burr8", sd = "burr8"`	`family = "burr8-burr8"`	Logit-logistic
Burr VIII-Logistic	`fd = "burr8", sd = "logistic"`	`family = "burr8-logistic"`	Bimodal
Logit-Burr VII	`fd = "logit", sd = "burr7"`	`family = "logit-burr7"`	Logit-logistic
Logit-Burr VIII	`fd = "logit", sd = "burr8"`	`family = "logit-burr8"`	Logit-logistic
Logit-Logistic	`fd = "logit", sd = "logistic"`	`family = "logit-logistic"`	Logit-logistic

Finite-tailed Shape Distributions

Distribution	R input	Alternative Input	Shape
ArcSinh-ArcSinh	`fd = "arcsinh", sd = "arcsinh"`	`family = "arcsinh-arcsinh"`	Finite-tailed
ArcSinh-Cauchy	`fd = "arcsinh", sd = "cauchy"`	`family = "arcsinh-cauchy"`	Finite-tailed
Cauchit-ArcSinh	`fd = "cauchit", sd = "arcsinh"`	`family = "cauchit-arcsinh"`	Finite-tailed
Cauchit-Cauchy	`fd = "cauchit", sd = "cauchy"`	`family = "cauchit-cauchy"`	Finite-tailed
T2-T2	`fd = "t2", sd = "t2"`	`family = "t2-t2"`	Finite-tailed

Kumaraswamy Distribution

Distribution	R input	Alternative Input	Shape
Kumaraswamy	`fd = "", sd = ""`	`family = "-"`	

Value

A list of distributions that are available in the current version of package.

Examples

```
cdfqrFamily()
```

`cdfquantreg` *CDF-Quantile Probability Distributions*

Description

`cdfquantreg` is the main function to fit a cdf quantile regression with a variety of distributions.

Usage

```
cdfquantreg(formula, fd = NULL, sd = NULL, data, family = NULL,
  start = NULL, control = cdfqr.control(...), ...)
```

Arguments

`formula`	A formula object, with the dependent variable (DV) on the left of an ~ operator, and predictors on the right. For the part on the right of '~', the specification of the location and dispersion submodels can be seperated by '|'. So `y ~ X1	X2` specifies that the DV is `y`, `X1` is the predictor in the location submodel, and `X2` is the predictor in the dispersion submodel.
`fd`	A string that specifies the parent distribution.
`sd`	A string that specifies the child distribution.
`data`	The data in a data.frame format
`family`	If 'fd' and 'sd' are not provided, the name of a member of the family of distributions can be provided (See `cdfqrFamily` for details of family functions)
`start`	The starting values for model fitting. If not provided, default values will be used.
`control`	Control optimization parameters (See `cdfqr.control`))
`...`	Currently ignored.

Details

The cdfquantreg function fits a quantile regression model with a distributions from the cdf-quantile family selected by the user (Smithson and Shou, 2015). The model is specified in a two-part formula, one part containing the predictors of the location parameter, and the second part containing the predictors of the dispersion parameter. The models are fitted in two stages, the first of which uses the Nelder-Mead algorithm and the second of which takes the estimates from the first stage and applies the BFGS algorithm to refine the estimates.

Value

An object of class `cdfquantreg` will be returned. Generic functions such as summary,print (e.g., print.cdfqr) and coef can be used to extract output (see summary.cdfqr for more details about the generic functions that can be used). Class of object is a list with the following output:

coefficients A named vector of coefficients.

residuals Raw residuals, the difference between the fitted values and the data.

fitted The fitted values, including full model fitted values, fitted values for the mean component, and fitted values for the dispersion component.

rmse The model root mean squared errors

rmseLogit The root mean squared errors between the logit of the fitted values, and the logit of the response values.

vcov The variance-covariance matrix of the coefficient estimates.

AIC, BIC Akaike's Information Criterion and Bayesian Information Criterion.

deviance The deviance for the model.

Examples

```
data(cdfqrExampleData)
fit <- cdfquantreg(crc99 ~ vert | confl, fd ='t2',sd ='t2', data = JurorData)

summary(fit)
```

`influence.cdfqr` *Influence Diagnosis For Fitted Cdfqr Object*

Description

Influence Diagnosis (dfbetas) For Fitted Cdfqr Object

Usage

```
## S3 method for class 'cdfqr'
influence(model, method = "dfbeta", type = c("full",
  "location", "dispersion"), what = "full", plot = FALSE, ...)

## S3 method for class 'cdfqr'
dfbeta(model, type = c("full", "location", "dispersion"),
  what = "full", ...)

## S3 method for class 'cdfqr'
dfbetas(model, type = c("full", "location", "dispersion"),
  what = "full", ...)
```

Arguments

model	A cdfqr model object
method	Currently only 'dfbeta' method is available.
type	A string that indicates whether the results for all paramters are to be returned, or only the location/dispersion submodel's parameters returned.
what	for influence statistics based on coefficient values, indicate the predictor variables that needs to be tested.
plot	if plot is needed.
...	currently ignored.s

Value

A matrix, each row of which contains the estimated influence on parameters when that row's observation is removed from the sample.

See Also

lm.influence, influence.measures

Examples

```
data(cdfqrExampleData)
fit <- cdfquantreg(crc99 ~ vert | confl, 't2', 't2', data = JurorData)
influcne <- influence(fit)
plot(influcne[,2])

## Not run:
# Same as influence(fit)
dfbetval <- dfbetas(fit)

## End(Not run)
```

IPCC *IPCC data-set*

Description

The IPCC data-set comprises the lower, best, and upper estimates for the phrases "likely" and "unlikely" in six IPCC report sentences.

Usage

```
IPCC
```

Format

A data frame with 4014 rows and 8 variables:

subj Subject ID number

treat Experimental conditions

valence Valence of the sentences

prob raw probability estimates

probm Linear transformed prob into (0, 1) interval

mid Distinguish lower, best and upper estiamtes

high Distinguish lower, best and upper estiamtes

Question IPCC question number

`JurorData` *Juror data*

Description

Juror Judgment Study.

Usage

```
JurorData
```

Format

A data frame with 104 rows and 3 variables:

crc99 The ratings of confidence levels with rescaling into the (0, 1) interval to avoide 1 and 0 values.

vert was the dummy variable for coding the conditions of verdict types, whereas

confl was the dummy variable for coding the conflict conditions

Source

https://dl.dropboxusercontent.com/u/1857674/betareg/betareg.html

plot.cdfqr *Plot Fitted Values/Residuals of A Cdfqr Object or Distribution*

Description

Plot Fitted Values/Residuals of A Cdfqr Object or Distribution

Usage

```
## S3 method for class 'cdfqr'
plot(x, mu = NULL, sigma = NULL, fd = NULL, sd = NULL,
  n = 10000, type = c("fitted"), ...)
```

Arguments

x	If the plot is based on the fitted values, provide a fitted cdfqr object.
mu, sigma,	fd, sd alternatively, mu and sigma, and the distribution can be specified
fd	A string that specifies the parent distribution.
sd	A string that specifies the sub-family distribution.
n	The number of random variates to be generated for user specified plot.
type	Currently only fitted values are available for generating plots.
...	other plot parameters pass onto plot.

Examples

```
data(cdfqrExampleData)
fit <- cdfquantreg(crc99 ~ vert | confl, 't2','t2', data = JurorData)
plot(fit)
```

pq *The Familiy of Distributions*

Description

Density function, distribution function, quantile function, and random generation of variates for a specified cdf-quantile distribution with mean equal to mean and standard deviation equal to sd.

Usage

```
pq(q, mu, sigma, fd, sd)

dq(x, mu, sigma, fd, sd)

rq(n, mu, sigma, fd, sd)

qq(p, mu, sigma, fd, sd)
```

Arguments

q	vector of quantiles.
mu	vector of means.
sigma	vector of standard deviations.
fd	A string that specifies the parent distribution.
sd	A string that specifies the sub-family distribution.
x	vector of quantiles.
n	Number of random samples.
p	vector of probabilities.

Value

dq gives the density, rq generates random variates, qq gives the quantile function, and pq gives the cumulative densitity of specified distribution.

Examples

```
x <- rq(5, mu = 0.5, sigma = 1, 't2','t2'); x

dq(x, mu = 0.5, sigma = 1, 't2','t2')

qtil <- pq(x, mu = 0.5, sigma = 1, 't2','t2');qtil

qq(qtil , mu = 0.5, sigma = 1, 't2','t2')
```

---

predict.cdfqr *Methods for Cdfqr Objects*

Description

Methods for obtaining the fitted/predicted values for a fitted cdfqr object.

Usage

```
## S3 method for class 'cdfqr'
predict(object, type = c("full", "mu", "sigma"), ...)

## S3 method for class 'cdfqr'
fitted(object, type = c("full", "mu", "sigma"), ...)
```

Arguments

object	A cdfqr model fit object
type	A character that indicates whether the full model prediction/fitted values are needed, or values for the 'mu' and 'sigma' submodel only.
...	currently ignored

Examples

```
data(cdfqrExampleData)
fit <- cdfquantreg(crc99 ~ vert | confl, 't2','t2', data = JurorData)

plot(predict(fit))
```

---

qrBoot *Bootstrapping for cdf quantile regression*

Description

qrBoot provides a simple bootstrapping method for estimating the parameters of a cdf quantile regression model.

Usage

```
qrBoot(object, rn, f = coef, R = 500, ci = 0.95)
```

Arguments

object	The fitted cdfqr model object
rn	The sample size of bootstrap samples
f	A function whose one argument is the name of a cdfqr object that will be applied to the updated cdfqr object to compute the statistics of interest. The default is coef.
R	Number of bootstrap samples.
ci	The confidence interval level to obtain the boostrap confidence intervals

Value

A matrix that includes the original statistics, boostrap means, and boostrap confidence intervals

Examples

```
data(cdfqrExampleData)
fit <- cdfquantreg(crc99 ~ vert | confl, 't2', 't2', data = JurorData)
qrBoot(fit, rn = 50, R = 50)
```

---

qrBugs *Running cdf quantile regression in OpenBUGS*

Description

Function used for running cdf quantile regression in OpenBUGS and performing Bayesian MCMC estimation. In addition, a simple random effects model is allowed to be estimated in this function.

Usage

```
qrBugs(formula, data, fd, sd, bugs = TRUE, random = NULL, nodes = NULL,
  inits = NULL, n.iter = 10000, n.burnin = n.iter/2,
  modelname = "bugmodel", working.directory = NULL, ...)
```

Arguments

formula	A formula object, with the DV on the left of an ~ operator, and predictors on the right. For the part on the right of '~', the specification of the submodels can be seperated by 'l'. So `y ~ X1 | X2` means the DV is y,X1 is the term in the mean submodel, and X2 is the term in the dispersion submodel.
data	The data file. Can be either a data.frame or a list of with specifying the names of variable in the list. The structure of the list and list component (e.g., matrix) should follow BUGS data format (see XXX for more details about preparing for list format of the data)
fd	A string that specifies the parent distribution.
sd	A string that specifies the sub-family distribution.
bugs	A logical value to indicate whether to use JAGS or OpenBUGS (NOTE: currently only OpenBUGS supports the customized likelihood functions; the package will update to allow JAGS as soon as JAGS provides that function).
random	Character or vector of chacters that indicate the random effect factors.
nodes	Character or vector of chacters that indicate the parameters to be estimated. The default nodes are the coefficients in both the mean and dispersion submodel.
inits	A list of values that serve as initial values for MCMC chain procedure.
n.iter	The number of MCMC samples to be drawn from the posterior distribution.

n.burnin	The number of samples to be discarded when summarizing the MCMC simulation results.
modelname	Name of the model (optional).
working.directory	the directory for generating temporary BUGS files.
...	further arguments to bugs.

Value

A bugs object (See more details)

See Also

bugs

Examples

```
data(cdfqrExampleData)
## Not run:
# Need to OpenBUGS has been installed, and R2OpenBUGS has been loaded first.
library(R2OpenBUGS)
bugfit <- qrBugs(crc99 ~ vert | confl, data = JurorData, 't2', 't2',clearWD=TRUE)
bugfit
Inference for Bugs model at "bugmodel.txt",
# Current: 2 chains, each with 10000 iterations (first 5000 discarded)
# Cumulative: n.sims = 10000 iterations saved
# mean  sd  2.5%   25%   50%   75% 97.5% Rhat n.eff
# b_0        0.8 0.1   0.6   0.7   0.8   0.9   1.0    1   830
# d_0       -0.2 0.1  -0.4  -0.3  -0.2  -0.1   0.1    1  1700
# b_vert     0.1 0.1  -0.1   0.0   0.1   0.2   0.3    1   170
# d_confl    0.0 0.1  -0.3  -0.1   0.0   0.0   0.2    1  4000
# deviance -49.3 2.8 -52.8 -51.3 -49.9 -47.9 -42.1    1  7400
#
# For each parameter, n.eff is a crude measure of effective sample size,
# and Rhat is the potential scale reduction factor (at convergence, Rhat=1).
#
# DIC info (using the rule, pD = Dbar-Dhat)
# pD = 4.0 and DIC = -45.3
# DIC is an estimate of expected predictive error (lower deviance is better).

## End(Not run)
```

---

qrGrad *Give the Gradient Function for CDF-Quantile Distribution Modles*

Description

Give the Gradient Function for CDF-Quantile Distribution Modles.

Usage

```
qrGrad(fd, sd)
```

Arguments

fd	A string that specifies the parent distribution.
sd	A string that specifies the sub-family distribution.

Value

grad The gradient function of parameter estimates, given a specified cdf-quantile distribution

Examples

```
qrGrad('t2','t2')
```

---

qrLogLik *Log Likelihood for Fitting Cdfquantile Distributions*

Description

Function to give the (negative) log likelihood for fitting cdfquantile distributions.

Usage

```
qrLogLik(y, mu, sigma, fd, sd)
```

Arguments

y	the vector to be evaluated.
mu	mean of the distribution.
sigma	sigma of the distribution.
fd	A string that specifies the parent distribution.
sd	A string that specifies the sub-family distribution.

Value

The negative log likelihood for fitting the data with a cdfquantile distribution.

Examples

```
y <- rbeta(20, 0.5, 0.5)
qrLogLik(y, mu = 0.5, sigma = 1, 't2','t2')
```

qrLogLikFun *Function to Give the Log Likelihood Function*

Description

Function to compute the (negative) log likelihood for fitting cdfquantile models.

Usage

```
qrLogLikFun(fd, sd)
```

Arguments

fd	A string that specifies the parent distribution.
sd	A string that specifies the sub-family distribution.

Value

The log-likelihood calculation function given a specified cdfquantile distribution.

Examples

```
qrLogLikFun('t2','t2')
```

qrPwlm *Probability Weighted L-moment Skewness and Kurtosis*

Description

Calculate the skew and kurtosis statistics based on probability weighted moments, via simulation method.

Usage

```
qrPwlm(x, n = NULL, mu = NULL, sigma = NULL, fd = NULL, sd = NULL)
```

Arguments

x	The vector of values for the calculation of Skewness and Kurtosis.
n	The number of samples drawn in the simulation. The higher this value, the greater accuracy.
mu	vector of means.
sigma	vector of standard deviations.
fd	A string that specifies the parent distribution.
sd	A string that specifies the sub-family distribution.

Details

This function computes the L-moment measures of skew and kurtosis, which may be computed via linear combinations of probability-weighted moments (Greenwood, Landwehr, Matalas and Wallis, 1979).

Value

The tau3(skew) and tau4(kurtosis) values of the L-moment.

References

Greenwood, J. A., Landwehr, J. M., Matalas, N. C., & Wallis, J. R. (1979). Probability weighted moments: definition and relation to parameters of several distributions expressable in inverse form. Water Resources Research, 15(5), 1049-1054.

Examples

```
qrPwlm(n = 1000, mu = 0.5, sigma = 1, fd = 't2', sd = 't2')
```

---

qrStart *Starting Value Genertion for CDF quantile Regressions*

Description

`qrStart` is the function for generating starting values for a cdf-quantile GLM null model.

Usage

```
qrStart(ydata, fd = NULL, sd = NULL)
```

Arguments

`ydata`	The variable to be modelled
`fd`	A string that specifies the parent distribution.
`sd`	A string that specifies the sub-family distribution.

Details

The start values for the location parameter in a null model are the median of the empirical distribtution, and a starting value for the dispersion parameter based on a specific quantile of the empirical distribution, specified according to the theoretical distribution on which the model is based. The start values for all new predictor coefficients in both the location and dispersion submodels are assigned the value 0.1.

Value

A vector that consists initial values for mu and sigma.

Examples

```
x <- rbeta(100, 1, 2)
qrStart(x, fd='t2', sd='t2')
#[1] -0.5938286  1.3996999
```

`residuals.cdfqr` *Register method for cdfqr object functions*

Description

Register method for cdfqr object functions.

Usage

```
## S3 method for class 'cdfqr'
residuals(object, type = c("raw", "pearson", "deviance"), ...)
```

Arguments

`object`	The cdfqr model project
`type`	The type of residuals to be extracted: `'raw'`, `'pearson'`,`'std.pearson'`, or `'deviance'`,
`...`	currently ignored

Value

residuals of a specified type.

Examples

```
data(cdfqrExampleData)
fit <- cdfquantreg(crc99 ~ vert | confl, 't2','t2', data = JurorData)

residuals(fit, "pearson")
```

scaleTR *Transform Values into (0, 1) Interval*

Description

scaleTR is function that rescales values of a variable into the (0, 1) interval.

Usage

```
scaleTR(y, high = NULL, low = NULL, data = NULL, N = NULL,
  scale = 0.5)
```

Arguments

y	A numeric vector, or a variable in a dataframe.
high	The highest possible value of that variable. The value should be equal or greater than the maximum value of y. If not supplied, the maximum value of y will be used.
low	The lowest possible value of that variable. The value should be equal or smaller than the minimum value of y. If not supplied, the minimum value of y will be used.
data	A dataframe that contains the variable y.
N	A integer, normally is the sample size or the number of values. If not supplied, the length of y will be used.
scale	A compressing parameter that determines the extend to which the boundary values are going to be pushed away from the boundray. See details.

Details

scaleTR used the method suggested by Smithson and Verkuilen (2006) and applies linear transformation to values into the open interval (0, 1). It first transform the values from their original scale by taking $y' = (y - a)/(b - a)$, where a is the lowest possible value of that variable and b is the highest possible value of that variable. Next, it compresses the range to avoid zeros and ones by taking $y" = (y'(N - 1) + c)/N$, where N is the sample size and c is the compressing parameter. The smaller value c is, the boundray values would be more approaching zeros and ones, and have greater impact on the estimation of the dispersion parameters in the cdf quantile model.

See Also

cdfquantreg

Examples

```
y <- rnorm(20, 0, 1)
ynew <- scaleTR(y)
```

summary.cdfqr *S3 Methods for getting output from fitted cdfqr Objects.*

Description

Give the Gradient Function for CDF-Quantile Distribution Modles

Usage

```
## S3 method for class 'cdfqr'
summary(object, ...)

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

## S3 method for class 'cdfqr'
logLik(object, ...)

## S3 method for class 'cdfqr'
deviance(object, ...)

## S3 method for class 'cdfqr'
coef(object, type = c("full", "mean", "sigma"), ...)

## S3 method for class 'cdfqr'
vcov(object, type = c("full", "mean", "sigma"), ...)

## S3 method for class 'cdfqr'
update(object, formula., ..., evaluate = TRUE)

## S3 method for class 'cdfqr'
formula(x, ...)

## S3 method for class 'cdfqr'
confint(object, parm, level = 0.95, submodel = "full", ...)
```

Arguments

object	The fitted cdfqr model.
...	Pass onto other functions or currently ignored
x	The fitted cdfqr model.
digits	Number of digits to be retained in printed output.
type, submodel	The parts of coefficients or variance-covariance matrix to be extracted.Can be "full", "mean",or "sigma".
formula.	Changes to the formula. See update.Formula for details.

evaluate	If true evaluate the new updated model else return the call for the new model.
parm	a specification of which parameters are to be given confidence intervals, either a vector of numbers or a vector of names. If missing, all parameters are considere.
level	the confidence level required.

Examples

```
data(cdfqrExampleData)
fit <- cdfquantreg(crc99 ~ vert | confl, 't2','t2', data = JurorData)

summary(fit)
print(fit)
logLik(fit)
coef(fit)
deviance(fit)
vcov(fit)
confint(fit)

#Update the model
fit2 <- update(fit, crc99 ~ vert*confl | confl)
summary(fit2)
```

Index