Package 'Qest'

October 29, 2026

Type Package

Title Quantile-Based Estimators

Version 2.0.0

Description Quantile-based estimators (Q-estimators) fit any parametric distribution through its quantile function, by minimizing the integrated loss of quantile regression. They are usually more robust than maximum likelihood estimators, and allow for censored and truncated data. The package implements user-defined quantile functions, built-in families (normal, gamma, continuous Poisson, uniform), linear models and proportional hazards models. The method is described in Sottile and Frumento (2022) <doi:10.1016/j.csda.2022.107471>.

License GPL (>= 2)

URL https://github.com/gianluca-sottile/Qest

BugReports https://github.com/gianluca-sottile/Qest/issues

Depends R (>= 4.1.0)

Imports graphics, matrixStats, stats, survival, utils

Suggests knitr, rmarkdown, testthat (>= 3.0.0)

VignetteBuilder knitr

Config/testthat/edition 3

Encoding UTF-8

Language en-US

Config/roxygen2/version 8.1.1

NeedsCompilation no

Author Gianluca Sottile [aut, cre] (ORCID:
<https://orcid.org/0000-0001-9347-7251>),
Paolo Frumento [aut]

Maintainer Gianluca Sottile <gianluca.sottile@unipa.it>

Repository CRAN

Date/Publication 2026-10-29 18:50:02 UTC

Contents

basehaz.Qcoxph . . . 2
invQ . . . 3
plot.Qest . . . 4
predict.Qcoxph . . . 5
predict.Qest . . . 6
predict.Qlm . . . 7
Qcoxph . . . 8
Qest . . . 10
Qest.control . . . 13
Qfamily . . . 14
Qlm . . . 15
residuals.Qcoxph . . . 17
summary.Qcoxph . . . 18
summary.Qest . . . 19
survfit.Qcoxph . . . 20
wtrunc . . . 21

Index **23**

`basehaz.Qcoxph` *Baseline Cumulative Hazard of a Qcoxph Model*

Description

Baseline Cumulative Hazard of a Qcoxph Model

Usage

```
basehaz.Qcoxph(fit, centered = TRUE, se.fit = FALSE, times)
```

Arguments

`fit`	an object of class `"Qcoxph"`.
`centered`	logical: if `TRUE`, the cumulative hazard at the mean of the covariates; otherwise at $x = 0$.
`se.fit`	logical: compute the variance of the estimates (delta method)?
`times`	the times at which H_0 is computed (by default, the observed times).

Details

The baseline cumulative hazard of the fitted model is piecewise linear, $H_0(t) = \sum_l \exp(\gamma_l) b_l(t)$. Unlike `survival::basehaz()`, this is not a step function.

Value

A data frame with columns `hazard` (the cumulative hazard), `time` and, if `se.fit = TRUE`, `varhaz`.

See Also

`Qcoxph()`, `survfit.Qcoxph()`

Examples

```
library(survival)
set.seed(1)
x <- runif(200)
t <- rexp(200, exp(x))
cens <- runif(200, 0, 3)
fit <- Qcoxph(Surv(pmin(t, cens), t <= cens) ~ x)
head(basehaz.Qcoxph(fit, se.fit = TRUE))
```

`invQ` *Inverse of a Quantile Function*

Description

Given a quantile function $Q(\tau \mid x, \theta)$ and a value of θ, computes $F(y_i \mid x_i, \theta)$ by bisection.

Usage

```
invQ(Q, theta, y, data, n.it = 17)
```

Arguments

`Q`	a quantile function `Q(theta, tau, data)` (see `Qest()`).
`theta`	the parameters.
`y`	the response: one value for each row of `data`.
`data`	a data frame.
`n.it`	the number of iterations of the bisection algorithm (the accuracy is $2^{-n.it}$).

Value

A vector with $F(y_i \mid x_i, \theta)$.

See Also

`Qest()`

Examples

```
Qlinmod <- function(theta, tau, data) {
  qnorm(tau, theta[2] + theta[3] * data$x, exp(theta[1]))
}
set.seed(1)
dat <- data.frame(x = runif(10))
U <- runif(10)
y <- Qlinmod(c(0, 1, 2), U, dat)
cbind(U, invQ(Qlinmod, c(0, 1, 2), y, dat))
```

`plot.Qest` *Diagnostic Plot of Q-Estimators*

Description

If the model is correctly specified, $F(y_i \mid x_i)$ are a sample from a $U(0, 1)$ distribution. The plot compares their empirical distribution (the Kaplan-Meier estimator, with censored or truncated data) with the uniform distribution function.

Usage

```
## S3 method for class 'Qest'
plot(x, conf.int = TRUE, ...)
```

Arguments

`x`	an object of class `"Qest"` (or `"Qlm"`).
`conf.int`	logical: add pointwise 95% confidence bands (Greenwood)?
`...`	additional graphical parameters passed to `graphics::plot()`.

Value

Invisibly, a data frame with the Kaplan-Meier estimate of the distribution of $F(y_i \mid x_i)$.

See Also

`Qest()`

Examples

```
set.seed(1)
y <- rgamma(300, shape = 2, scale = 3)
plot(Qest(y ~ 1, Q = Qgamma))   # good fit
plot(Qest(y ~ 1, Q = Qnorm))    # poor fit
```

predict.Qcoxph *Predictions from a Qcoxph Model*

Description

Predictions from a Qcoxph Model

Usage

```
## S3 method for class 'Qcoxph'
predict(
  object,
  newdata,
  type = c("lp", "risk", "expected", "survival", "quantile"),
  se.fit = FALSE,
  p = c(0.25, 0.5, 0.75),
  ...
)
```

Arguments

object	an object of class "Qcoxph".
newdata	an optional data frame. For type = "expected" and type = "survival" it must contain the response variables.
type	the type of prediction: the linear predictor ("lp", centered at the mean of the covariates, as in survival::coxph()), the relative risk ("risk", $\exp(lp)$), the expected number of events $H(y \mid x)$ ("expected", minus $H(z \mid x)$ with truncation), the survival probability $\exp(-expected)$ ("survival"), or the quantiles of the time-to-event ("quantile").
se.fit	logical: compute standard errors (delta method)? Not available for type = "quantile".
p	the quantile orders for type = "quantile".
...	not used.

Value

A vector, or a list with items fit and se.fit; for type = "quantile", a matrix with one column for each value of p.

See Also

Qcoxph(), survfit.Qcoxph()

Examples

```
library(survival)
set.seed(1)
dat <- data.frame(x = runif(200))
t <- rexp(200, exp(dat$x)); cens <- runif(200, 0, 3)
dat$y <- pmin(t, cens); dat$d <- as.numeric(t <= cens)
fit <- Qcoxph(Surv(y, d) ~ x, data = dat)
predict(fit, newdata = data.frame(x = c(0, 0.5, 1)), type = "risk", se.fit = TRUE)
predict(fit, newdata = data.frame(x = c(0, 1)), type = "quantile", p = c(0.25, 0.5))
```

`predict.Qest` *Predictions from Q-Estimators*

Description

Computes the fitted quantiles $Q(p \mid x)$, or the fitted distribution function $F(y \mid x)$ at the observed (or new) values of the response.

Usage

```
## S3 method for class 'Qest'
predict(
  object,
  newdata,
  type = c("quantile", "CDF"),
  p = c(0.1, 0.25, 0.5, 0.75, 0.9),
  ...
)
```

Arguments

`object`	an object of class `"Qest"`.
`newdata`	an optional data frame. For `type = "CDF"`, it must contain the response.
`type`	`"quantile"` (default) or `"CDF"`.
`p`	the quantile orders for `type = "quantile"`.
`...`	additional arguments of a user-defined `Q`, if they differ from those used to fit the model.

Details

For a user-defined `Q`, `newdata` is passed as the argument `data` of `Q`. For `type = "CDF"`, $F(y \mid x)$ is computed by inverting Q (see `invQ()`). For censored data, the CDF is evaluated at the observed times.

Value

For `type = "quantile"`, a matrix with one row for each observation and one column for each value of `p`; for `type = "CDF"`, a vector.

See Also

Qest(), invQ()

Examples

```
set.seed(1)
dat <- data.frame(x = runif(200))
dat$y <- rgamma(200, shape = 2, scale = exp(1 + dat$x))
fit <- Qest(y ~ x, Q = Qgamma, data = dat)
predict(fit, newdata = data.frame(x = c(0, 1)), p = c(0.1, 0.5, 0.9))
head(predict(fit, type = "CDF"))
```

---

predict.Qlm *Predictions from Qlm Models*

Description

Predictions from Qlm Models

Usage

```
## S3 method for class 'Qlm'
predict(
  object,
  newdata,
  type = c("response", "quantile"),
  se.fit = FALSE,
  p = c(0.1, 0.25, 0.5, 0.75, 0.9),
  na.action = stats::na.pass,
  ...
)
```

Arguments

object	an object of class "Qlm".
newdata	an optional data frame.
type	"response" (the fitted mean, i.e. median, $x^\top\beta$) or "quantile" ($x^\top\beta+\sigma\Phi^{-1}(p)$).
se.fit	logical: return the standard errors of the fitted mean?
p	the quantile orders for type = "quantile".
na.action	how to treat missing values in newdata.
...	not used.

Value

A vector, a list with items fit and se.fit, or (for type = "quantile") a matrix with one column for each value of p.

See Also

`Qlm()`

Examples

```
set.seed(1)
x <- runif(100)
y <- 1 + 2 * x + rnorm(100)
fit <- Qlm(y ~ x)
predict(fit, newdata = data.frame(x = c(0, 0.5)), se.fit = TRUE)
predict(fit, newdata = data.frame(x = c(0, 0.5)), type = "quantile", p = c(0.1, 0.9))
```

Qcoxph *Q-Estimation of Proportional Hazards Regression Models*

Description

Fits a proportional hazards model with the Q-estimator of Sottile and Frumento (2022). The baseline cumulative hazard $H_0(t)$ is piecewise linear, so that the model is fully parametric, and $Q(\tau \mid x) = H_0^{-1}(-\exp(-x^\top \beta)\log(1-\tau))$.

Usage

```
Qcoxph(
  formula,
  weights,
  start,
  data,
  subset,
  na.action,
  knots,
  wtau = NULL,
  control = Qcoxph.control(),
  ...
)
```

Arguments

`formula`	a formula with a `survival::Surv()` response: `Surv(time, event)` for right-censored data, or `Surv(start, stop, event)` for left-truncated and right-censored data. Times must be positive.
`weights`	an optional vector of non-negative (sampling) weights.
`start`	optional starting values for the regression coefficients β.
`data`	an optional data frame.
`subset, na.action`	as in `stats::lm()`.

knots	the knots of $H_0(t)$: a number (the number of internal knots, placed at the quantiles of the observed event times) or a vector with the positions of all knots, including the boundaries (the first knot must not be larger than the smallest time, the last not smaller than the largest time). By default, `max(1, min(floor(n.events / 30), 3))` internal knots are used.
wtau	an optional weighting function $w(\tau)$, e.g. `wtrunc()`.
control	a list created by `Qcoxph.control()`.
...	additional arguments of `wtau`.

Details

The model is $h(t \mid x) = h_0(t) \exp(x^\top \beta)$, with $H_0(t) = \sum_l \exp(\gamma_l)\, b_l(t)$, where b_l is a piecewise linear basis. All quantities of the estimating equations have closed forms (Appendix D2 of the paper). The coefficients have the same interpretation as in `survival::coxph()`; the estimates are typically very close, but more robust. Starting points are obtained from the partial likelihood and the Breslow estimator of H_0.

Methods are available to compute the fitted survival curves (`survfit.Qcoxph()`, which returns an object of class `"survfit"`), predictions (`predict.Qcoxph()`), residuals (`residuals.Qcoxph()`) and the baseline cumulative hazard (`basehaz.Qcoxph()`). All of them are based on the fitted parametric model.

Value

An object of class `c("Qcoxph", "Qest")`, a list with items

coefficients, var	the estimated β and their covariance matrix.
obj.function	the loss at the solution.
iter, converged	number of iterations and convergence.
linear.predictors	$x_i^\top \beta - \bar{x}^\top \beta$.
residuals	martingale residuals.
means, n, nevent	means of the covariates, number of observations and of events.
concordance	Harrell's concordance index (from `survival::concordancefit()`).
internal	the knots, the coefficients γ of H_0, the full covariance matrix (`covar`), $F(y_i \mid x_i)$ (`CDF`), $f(y_i \mid x_i)$ (`PDF`), and other quantities used by the methods.

and `terms`, `assign`, `formula`, `call`, `y`, `x`, `model`, `xlevels`, `na.action`.

References

Sottile, G. and Frumento, P. (2022). Robust estimation and regression with parametric quantile functions. *Computational Statistics and Data Analysis*, 171, 107471. doi:10.1016/j.csda.2022.107471

See Also

`survfit.Qcoxph()`, `predict.Qcoxph()`, `residuals.Qcoxph()`, `basehaz.Qcoxph()`, `Qest()`

Examples

```
library(survival)
set.seed(1)
n <- 200
x <- runif(n, 0, 3)
t <- rweibull(n, shape = 2, scale = (1 / exp(2 + 2 * x))^(1 / 2))
cens <- runif(n, 0, 1)
dat <- data.frame(y = pmin(t, cens), d = as.numeric(t <= cens), x = x)

m1 <- coxph(Surv(y, d) ~ x, data = dat)
m2 <- Qcoxph(Surv(y, d) ~ x, data = dat)
summary(m2)
cbind(coxph = coef(m1), Qcoxph = coef(m2))
sf <- survfit(m2, newdata = data.frame(x = c(0.5, 1.5)))
plot(sf, col = 1:2)
```

Qest *Quantile-Based Estimators (Q-Estimators)*

Description

Fits a parametric model by minimizing the integrated loss of quantile regression (Sottile and Frumento, 2022). The model is specified through its quantile function $Q(\tau \mid x, \theta)$, either written by the user or chosen among the built-in families (see Qfamily). Censored and truncated data are allowed.

Usage

```
Qest(
  formula,
  Q,
  weights,
  start,
  data,
  subset,
  na.action,
  ntau = 199,
  wtau = NULL,
  control = Qest.control(),
  ...
)
```

Arguments

formula	a two-sided formula of the form `y ~ x`. The response can be a numeric vector, a right-censored `survival::Surv(time, event)` object, or a left-truncated and right-censored `survival::Surv(start, stop, event)` object. With a user-defined `Q`, the right-hand side of the formula is only used to define the covariates of the starting points: write `y ~ 1` if no covariates are used by `Q`.

`Q`	a quantile function `Q(theta, tau, data)` (see "Details"), or a Qfamily (`Qnorm`, `Qgamma`, `Qpois`, `Qunif`).
`weights`	an optional vector of non-negative weights. They are treated as sampling weights: the (sandwich) standard errors do not depend on their scale.
`start`	the starting points. With a user-defined `Q`, `start` is required, because its length defines the number of parameters: use `NA` for the parameters without a good starting value (with any `NA`, the starting points are estimated, see "Details"). With a `Qfamily`, `start` is optional.
`data`	an optional data frame containing the variables in the model. It is also the argument `data` of `Q`.
`subset`, `na.action`	as in `stats::lm()`.
`ntau`	the number of points of the grid of quantile orders used for numerical integration (default 199).
`wtau`	an optional weighting function $w(\tau)$, e.g. `wtrunc()`. Its first argument must be `tau`; other arguments are passed through `...`.
`control`	a list of operational parameters created by `Qest.control()`.
`...`	additional named arguments of `wtau` or `Q`.

Details

The Q-estimator minimizes

$$L(\theta) = \sum_i \int_0^1 w(\tau)\rho_\tau(y_i - Q(\tau \mid x_i, \theta))\, d\tau,$$

where $\rho_\tau(u) = u(\tau - I(u < 0))$. It is consistent and asymptotically normal, and its robustness can be tuned by choosing $w(\tau)$ (Sottile and Frumento, 2022). For censored and truncated data, the estimating equations of Appendix A of the paper are solved.

User-defined quantile functions. `Q(theta, tau, data)` must have these three arguments, in this order (other arguments can follow and are passed through `...`), and must return, for each row of `data`, $Q(\tau \mid x_i, \theta)$. The argument `tau` is either a matrix with one row for each row of `data` (then `Q` returns a matrix of the same size), or a vector with one value for each row of `data` (then `Q` returns a vector). `data` is the data frame used to fit the model (after removing missing values, `subset` and observations with zero weight). `Q` must be increasing in `tau` and twice differentiable in `theta`; it is evaluated on a grid of `ntau` quantile orders in $[5 \cdot 10^{-4}, 1 - 5 \cdot 10^{-4}]$, and its derivatives with respect to `theta` are computed by central differences.

Starting points. If `start` contains `NA` (or `control$restart = TRUE`), a nonparametric estimate $\hat{F}(y \mid x)$ is obtained from a piecewise-constant hazard model that uses the covariates in `formula`, and the starting points minimize $\sum_i (y_i - Q(\hat{F}(y_i \mid x_i) \mid x_i, \theta))^2$ (Sottile and Frumento, 2022, Algorithm 1), searched from the values in `start` (with `NA` replaced by zero): with poorly scaled data, rough starting values for all parameters are recommended.

Algorithm. The estimating equations are solved by Newton-Raphson with backtracking, switching to Levenberg-Marquardt steps when the Newton step fails. The Jacobian is the exact derivative of the discretized estimating equations. Standard errors are computed with the sandwich formula.

Value

An object of class `"Qest"`, a list with items

`coefficients`	the estimated parameters.
`std.errs`	their standard errors.
`covar`	the estimated covariance matrix.
`obj.function`	the value of the loss at the solution (for censored or truncated data, the loss of Appendix A, which is not the function whose gradient is solved).
`ee`, `jacobian`	the estimating equations (divided by the sample size) and their Jacobian at the solution.
`CDF`, `PDF`	the fitted $F(y_i \mid x_i)$ and $f(y_i \mid x_i)$.
`converged`, `n.it`	convergence of the algorithm and number of iterations.
`internal`	quantities used by the methods.

and the usual `call`, `formula`, `terms`, `data`, `na.action`, `xlevels`, `contrasts`, `control`.

References

Sottile, G. and Frumento, P. (2022). Robust estimation and regression with parametric quantile functions. *Computational Statistics and Data Analysis*, 171, 107471. doi:10.1016/j.csda.2022.107471

See Also

Qfamily, `Qlm()`, `Qcoxph()`, `summary.Qest()`, `wtrunc()`, `invQ()`.

Examples

```
# Normal linear model
Qlinmod <- function(theta, tau, data) {
  X <- model.matrix(~ x1 + x2, data = data)
  qnorm(tau, c(X %*% theta[-1]), exp(theta[1]))
}
set.seed(1)
n <- 100
dat <- data.frame(x1 = rnorm(n), x2 = runif(n, 0, 3))
dat$y <- 4 + dat$x1 + 2 * dat$x2 + rnorm(n)
m1 <- Qest(y ~ x1 + x2, Q = Qlinmod, start = rep(NA, 4), data = dat)
m2 <- Qest(y ~ x1 + x2, Q = Qnorm, data = dat)    # the same model, as a Qfamily
m3 <- Qlm(y ~ x1 + x2, data = dat)                # ... or with Qlm (fastest)
summary(m1)
cbind(user = coef(m1), family = coef(m2))


# A Weibull proportional hazards model with censored data
QWeibPH <- function(theta, tau, data) {
  shape <- exp(theta[1])
  X <- model.matrix(~ x, data = data)
  qweibull(tau, shape = shape, scale = exp(-c(X %*% theta[-1]) / shape))
}
```

```
dat <- data.frame(x = runif(n, 0, 3))
t <- QWeibPH(c(log(2), -0.5, 1), runif(n), dat)
cens <- runif(n, 0, 2)
dat$y <- pmin(t, cens); dat$d <- as.numeric(t <= cens)
m4 <- Qest(survival::Surv(y, d) ~ x, Q = QWeibPH, start = rep(NA, 3), data = dat)
summary(m4)

# Gamma model with more weight around the median
y <- rgamma(n, shape = 3, scale = 2)
m5 <- Qest(y ~ 1, Q = Qgamma, wtau = function(tau, h) dnorm((tau - 0.5) / h), h = 0.2)
coef(m5)
```

`Qest.control` *Control Parameters of Qest, Qlm and Qcoxph*

Description

Control Parameters of Qest, Qlm and Qcoxph

Usage

```
Qest.control(
  tol = 1e-08,
  maxit = NULL,
  display = FALSE,
  restart = FALSE,
  deriv.step = 1e-04,
  ...
)

Qcoxph.control(tol = 1e-08, maxit = NULL, display = FALSE, ...)
```

Arguments

`tol`	convergence tolerance on the norm of the estimating equations (divided by the sample size).
`maxit`	maximum number of iterations (default `50 + 25 * npar`).
`display`	logical: print the iterations?
`restart`	logical: if `TRUE`, the starting points of `Qest()` with a user-defined `Q` are re-estimated even if `start` has no missing values.
`deriv.step`	relative step of the central differences used for the derivatives of a user-defined `Q` with respect to `theta`: the step for θ_j is `deriv.step * max(1, abs(theta_j))`.
`...`	arguments of 'Qest' 1.0.x that were removed (`safeit`, `alpha0`): they give an error.

Value

A list with the control parameters.

See Also

`Qest()`, `Qlm()`, `Qcoxph()`

Examples

```
Qest.control(tol = 1e-10, display = TRUE)
```

`Qfamily` *Family Objects for Qest*

Description

Built-in parametric models for `Qest()`, with analytical (or nearly analytical) estimating equations: much faster and more accurate than the equivalent user-defined quantile functions.

Usage

```
Qgamma()

Qnorm()

Qpois(offset = NULL)

Qunif(min = TRUE)
```

Arguments

`offset`	an optional vector of offsets of the log-rate (`Qpois` only). An `offset()` term in the formula can be used instead.
`min`	logical (`Qunif` only): if `TRUE`, fit a $U(a, b)$ model; if `FALSE`, a $U(0, b)$ model.

Details

A family can be passed to `Qest()` by name (`"Qpois"`), as a function (`Qpois`) or as a call (`Qpois()`); the latter allows to pass arguments.

- `Qnorm`: normal homoskedastic model, $Q(\tau \mid x) = x^\top \beta + \sigma \Phi^{-1}(\tau)$; parameters `log(sigma)`, `beta`. With complete data, `Qlm()` fits the same model with a richer output.
- `Qgamma`: Gamma model with shape a and log-scale $x^\top \beta$; parameters `log(shape)`, `beta`.
- `Qpois`: Poisson model with log-rate $x^\top \beta$. To obtain a continuous quantile function, `qpois` is replaced by the inverse with respect to y of the regularized upper incomplete gamma function $\Gamma(y, \lambda)/\Gamma(y)$ (Ilienko, 2013). The model should be applied to a jittered response, `y + runif(n)`.

- `Qunif`: Uniform model $U(a(x), b(x))$ with $a(x) = x^\top \alpha$ and $b(x) = x^\top \beta$, or $U(0, b(x))$ if `min = FALSE`.

`Qnorm` and `Qgamma` can be used with censored and truncated data; `Qpois` and `Qunif` cannot. `Qgamma` and `Qpois` require an intercept.

Value

An object of class `"Qfamily"`.

References

Ilienko, A. (2013). Continuous counterparts of Poisson and binomial distributions and their properties. *Annales Univ. Sci. Budapest., Sect. Comp.*, 39, 137-147.

Sottile, G. and Frumento, P. (2022). Robust estimation and regression with parametric quantile functions. *Computational Statistics and Data Analysis*, 171, 107471. doi:10.1016/j.csda.2022.107471

See Also

`Qest()`

Examples

```
set.seed(1)
n <- 250
x <- runif(n)
eta <- 1 + 2 * x

m1 <- Qest(y ~ x, Qnorm, data = data.frame(y = rnorm(n, eta, exp(1)), x = x))
m2 <- Qest(y ~ x, Qgamma, data = data.frame(y = rgamma(n, shape = exp(1), scale = exp(eta)), x = x))
m3 <- Qest(y ~ x, Qpois, data = data.frame(y = rpois(n, exp(eta)) + runif(n), x = x))
m4 <- Qest(y ~ x, Qunif(min = TRUE), data = data.frame(y = runif(n, 0, eta), x = x))
m5 <- Qest(y ~ x, Qunif(min = FALSE), data = data.frame(y = runif(n, 0, eta), x = x))
coef(m2)
```

Qlm *Q-Estimation of Linear Regression Models*

Description

Fits a linear model with normal homoskedastic errors, $Q(\tau \mid x) = x^\top \beta + \sigma \Phi^{-1}(\tau)$, with the Q-estimator of Sottile and Frumento (2022). The estimating equations and the Hessian have closed forms (Appendix D1 of the paper), and the estimator is fast and robust to outliers in the response.

Usage

```
Qlm(
  formula,
  data,
  subset,
  weights,
  na.action,
  start = NULL,
  contrasts = NULL,
  wtau = NULL,
  control = Qest.control(),
  ...
)

Qlm.fit(
  y,
  X,
  w = rep(1, nobs),
  start = NULL,
  wtau = NULL,
  control = Qest.control(),
  ...
)
```

Arguments

`formula`, `data`, `subset`, `na.action`, `contrasts`	as in `stats::lm()`.
`weights`	an optional vector of non-negative (sampling) weights; the standard errors do not depend on their scale.
`start`	optional starting values for the regression coefficients (excluding σ).
`wtau`	an optional weighting function $w(\tau)$, e.g. `wtrunc()`.
`control`	a list created by `Qest.control()`.
`...`	additional arguments of `wtau`.
`y`, `X`, `w`	for `Qlm.fit`: the response, the design matrix and the weights.

Details

`Qlm(formula)` estimates the same model as `Qest(formula, Q = Qnorm)`, with a much richer output. Censored and truncated data are not allowed (use `Qest()` with `Q = Qnorm`). Standard errors are computed with the sandwich formula, and the tests in `summary.Qlm()` are based on the asymptotic normal distribution.

Value

`Qlm` returns an object of class `c("Qlm", "Qest")` with items

`coefficients`, `std.errs`, `covar`
: the estimated regression coefficients, their standard errors and covariance matrix.

`dispersion`
: the estimated σ^2.

`sigma`
: the estimated σ, with attribute `"std.err"` (the standard error of $\log\sigma$).

`residuals`, `fitted.values`
: residuals and fitted values.

`CDF`
: the fitted $F(y_i \mid x_i)$.

`obj.function`
: the minimized loss.

`gradient`, `hessian`
: the estimating equations and the Hessian of the loss at the solution (including $\log\sigma$).

`covar.full`
: the covariance matrix of $(\log\sigma, \beta)$.

and `rank`, `df.residual`, `weights`, `converged`, `n.it`, `call`, `terms`, `model`, `xlevels`, `contrasts`, `na.action`. `Qlm.fit` returns a similar list.

References

Sottile, G. and Frumento, P. (2022). Robust estimation and regression with parametric quantile functions. *Computational Statistics and Data Analysis*, 171, 107471. doi:10.1016/j.csda.2022.107471

See Also

`summary.Qlm()`, `predict.Qlm()`, `Qest()`

Examples

```
set.seed(1234)
n <- 100
x1 <- rnorm(n)
x2 <- runif(n, 0, 3)
y <- rnorm(n, 4 + x1 + 2 * x2, 1)
y[1:5] <- y[1:5] + 20      # outliers

m1 <- Qlm(y ~ x1 + x2)
summary(m1)
m2 <- Qlm(y ~ x1 + x2, wtau = wtrunc, delta.left = 0.05, delta.right = 0.05)
cbind(lm = coef(lm(y ~ x1 + x2)), Qlm = coef(m1), Qlm.trunc = coef(m2))
```

---

residuals.Qcoxph *Residuals of a Qcoxph Model*

Description

Residuals of a Qcoxph Model

Usage

```
## S3 method for class 'Qcoxph'
residuals(object, type = c("martingale", "deviance"), ...)
```

Arguments

object	an object of class "Qcoxph".
type	"martingale" ($d_i - H(y_i \mid x_i)$) or "deviance".
...	not used.

Value

A vector of residuals.

See Also

Qcoxph()

---

summary.Qcoxph *Summarizing Qcoxph Fits*

Description

Summarizing Qcoxph Fits

Usage

```
## S3 method for class 'Qcoxph'
summary(object, conf.int = 0.95, scale = 1, ...)

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

Arguments

object	an object of class "Qcoxph".
conf.int	the level of the confidence intervals for the hazard ratios.
scale	a vector of scale factors for the coefficients.
...	not used.
x	an object of class "summary.Qcoxph".
digits, signif.stars	printing options.

Value

An object of class "summary.Qcoxph", with the table of coefficients (coefficients), the confidence intervals of the hazard ratios (conf.int), the concordance index and its standard error (concordance), and the loss (obj.function).

See Also

Qcoxph()

---

summary.Qest *Summarizing Q-Estimators*

Description

Summary methods for objects of class "Qest" (fitted by Qest()) and "Qlm" (fitted by Qlm()).

Usage

```
## S3 method for class 'Qest'
summary(object, covar = FALSE, ...)

## S3 method for class 'summary.Qest'
print(
  x,
  digits = max(3L, getOption("digits") - 3L),
  signif.stars = getOption("show.signif.stars"),
  ...
)

## S3 method for class 'Qlm'
summary(object, correlation = FALSE, ...)

## S3 method for class 'summary.Qlm'
print(
  x,
  digits = max(3L, getOption("digits") - 3L),
  signif.stars = getOption("show.signif.stars"),
  ...
)
```

Arguments

object	an object of class "Qest" or "Qlm".
covar	logical: include the covariance matrix?
...	not used.
x	a summary object.

`digits`, `signif.stars`
: printing options.

`correlation`
: logical: include the correlation matrix of the coefficients (`summary.Qlm`)?

Details

The tests are based on the asymptotic normal distribution of the estimators, with standard errors from the sandwich formula (Sottile and Frumento, 2022).

Value

`summary.Qest` returns an object of class `"summary.Qest"` with the table of coefficients (`coefficients`), the loss (`obj.function`), the numbers of observations and parameters, the convergence status and, optionally, the covariance matrix. `summary.Qlm` returns an object of class `"summary.Qlm"` with the table of coefficients, the estimate of σ and its standard error, the residuals and, optionally, the correlation matrix.

See Also

`Qest()`, `Qlm()`

Examples

```
set.seed(1)
y <- rgamma(200, shape = 3, scale = 2)
fit <- Qest(y ~ 1, Q = Qgamma)
summary(fit, covar = TRUE)
```

---

survfit.Qcoxph *Survival Curves from a Qcoxph Model*

Description

Computes the survival curves $S(t \mid x) = \exp\{-H_0(t)\exp(x^\top \beta)\}$ of the fitted model, with pointwise confidence intervals based on the delta method.

Usage

```
## S3 method for class 'Qcoxph'
survfit(
  formula,
  newdata,
  se.fit = TRUE,
  conf.int = 0.95,
  conf.type = c("log", "log-log", "plain", "none"),
  times,
  ...
)
```

Arguments

formula	an object of class "Qcoxph".
newdata	a data frame with the covariates of the curves. By default, one curve at the mean of the covariates.
se.fit	logical: compute the standard errors?
conf.int	the level of the confidence intervals.
conf.type	the transformation used for the confidence intervals: "log" (default, symmetric on the scale of H), "log-log", "plain" or "none".
times	the times (by default, the observed times).
...	not used.

Value

An object of class "survfit" (see survival::survfit.object): the methods print, plot, lines and summary of 'survival' can be used.

See Also

Qcoxph(), predict.Qcoxph()

Examples

```
library(survival)
set.seed(1)
dat <- data.frame(x = runif(200))
t <- rexp(200, exp(dat$x)); cens <- runif(200, 0, 3)
dat$y <- pmin(t, cens); dat$d <- as.numeric(t <= cens)
fit <- Qcoxph(Surv(y, d) ~ x, data = dat)
sf <- survfit(fit, newdata = data.frame(x = c(0, 1)))
plot(sf, col = 1:2, conf.int = TRUE)
```

---

wtrunc *Weights for Truncated Quantile Integration*

Description

A weighting function $w(\tau)$ that assigns zero weight to the quantiles below delta.left and above 1 - delta.right, to be used as the argument wtau of Qest(), Qlm() and Qcoxph(). Truncated weights make the estimators more robust to outliers (Sottile and Frumento, 2022, Section 3).

Usage

```
wtrunc(
  tau,
  delta.left = 0.05,
  delta.right = 0.05,
  smooth = FALSE,
  sigma = 0.01
)
```

Arguments

tau	a vector of quantile orders in $[0, 1]$.
delta.left, delta.right	the proportions of quantiles with zero weight in the left and right tail.
smooth	logical: if TRUE, a smooth version based on the normal distribution function is used.
sigma	the scale of the smooth version (only used if smooth = TRUE).

Details

With smooth = FALSE, $w(\tau) = I(\delta_L \leq \tau \leq 1 - \delta_R)$. With smooth = TRUE, the indicators are replaced by $\Phi((\tau - \delta_L)/\sigma)$ and $1 - \Phi((\tau - 1 + \delta_R)/\sigma)$. A delta equal to zero means no truncation on that side. The arguments delta.left, delta.right, smooth and sigma are passed through the ... of the fitting functions.

Value

A vector of weights of the same length as tau.

References

Sottile, G. and Frumento, P. (2022). Robust estimation and regression with parametric quantile functions. *Computational Statistics and Data Analysis*, 171, 107471. doi:10.1016/j.csda.2022.107471

Examples

```
taus <- seq(0, 1, length.out = 1000)
plot(taus, wtrunc(taus, delta.left = 0.05, delta.right = 0.1), type = "l")
lines(taus, wtrunc(taus, 0.05, 0.1, smooth = TRUE, sigma = 0.01), col = 2)

# in a model
set.seed(1)
y <- c(rnorm(95), rnorm(5, 10))   # 5% of outliers
fit <- Qlm(y ~ 1, wtau = wtrunc, delta.left = 0.05, delta.right = 0.05)
coef(fit)
```

Index

basehaz.Qcoxph, 2
basehaz.Qcoxph(), *9*

graphics::plot(), *4*

invQ, 3
invQ(), *6, 7*, *12*

plot.Qest, 4
predict.Qcoxph, 5
predict.Qcoxph(), *9*, *21*
predict.Qest, 6
predict.Qlm, 7
predict.Qlm(), *17*
print.summary.Qcoxph *(*summary.Qcoxph*)*, 18
print.summary.Qest *(*summary.Qest*)*, 19
print.summary.Qlm *(*summary.Qest*)*, 19

Qcoxph, 8
Qcoxph(), *3*, *5*, *12*, *14*, *18, 19*, *21*
Qcoxph.control *(*Qest.control*)*, 13
Qcoxph.control(), *9*
Qest, 10
Qest(), *3, 4*, *7*, *9*, *13–17*, *19–21*
Qest.control, 13
Qest.control(), *11*, *16*
Qfamily, *10–12*, 14
Qgamma *(*Qfamily*)*, 14
Qlm, 15
Qlm(), *8*, *12*, *14*, *19–21*
Qnorm *(*Qfamily*)*, 14
Qpois *(*Qfamily*)*, 14
Qunif *(*Qfamily*)*, 14

residuals.Qcoxph, 17
residuals.Qcoxph(), *9*

stats::lm(), *8*, *11*, *16*
summary.Qcoxph, 18
summary.Qest, 19
summary.Qest(), *12*
summary.Qlm *(*summary.Qest*)*, 19
summary.Qlm(), *16, 17*
survfit.Qcoxph, 20
survfit.Qcoxph(), *3*, *5*, *9*
survival::basehaz(), *2*
survival::concordancefit(), *9*
survival::coxph(), *5*, *9*
survival::survfit.object, *21*

wtrunc, 21
wtrunc(), *9*, *11, 12*, *16*