# Package: Mqrcm (via r-universe)

September 10, 2024

**Type** Package

**Title** M-Quantile Regression Coefficients Modeling

**Version** 1.3

**Date** 2024-02-12

**Author** Paolo Frumento <paolo.frumento@unipi.it>

**Maintainer** Paolo Frumento <paolo.frumento@unipi.it>

**Description** Parametric modeling of M-quantile regression coefficient functions.

**Imports** stats, utils, graphics, Hmisc

**Depends** pch (>= 2.1)

**License** GPL-2

**RoxygenNote** 7.2.3

**NeedsCompilation** no

**Repository** CRAN

**Date/Publication** 2024-02-12 20:30:02 UTC

## Contents

Mqrcm-package . . . . . 2
iMqr . . . . . 3
plf . . . . . 6
plot.iMqr . . . . . 7
predict.iMqr . . . . . 8
psi . . . . . 10
slp . . . . . 11
summary.iMqr . . . . . 12

**Index** **15**

`Mqrcm-package` *M-Quantile Regression Coefficients Modeling*

## Description

This package implements Frumento and Salvati (2020) method for M-quantile regression coefficients modeling (Mqrcm), in which M-quantile regression coefficients are described by (flexible) parametric functions of the order of the quantile. This permits modeling the entire conditional M-quantile function of a response variable.

## Details

| | |
|---|---|
| Package: | Mqrcm |
| Type: | Package |
| Version: | 1.3 |
| Date: | 2024-02-12 |
| License: | GPL-2 |

The function `iMqr` permits specifying the regression model. Two special functions, `slp` and `plf`, are provided to facilitate model building. The auxiliary functions `summary.iMqr`, `predict.iMqr`, and `plot.iMqr` can be used to extract information from the fitted model.

## Author(s)

Paolo Frumento

Maintainer: Paolo Frumento <paolo.frumento@unipi.it>

## References

Frumento, P., Salvati, N. (2020). *Parametric modeling of M-quantile regression coefficient functions with application to small area estimation*, Journal of the Royal Statistical Society, Series A, 183(1), p. 229-250.

## Examples

```
# use simulated data

n <- 250
x <- rexp(n)
y <- runif(n, 0, 1 + x)
model <- iMqr(y ~ x, formula.p = ~ p + I(p^2))
summary(model)
summary(model, p = c(0.1,0.2,0.3))
predict(model, type = "beta", p = c(0.1,0.2,0.3))
predict(model, type = "CDF", newdata = data.frame(x = c(1,2,3), y = c(0.5,1,2)))
predict(model, type = "QF", p = c(0.1,0.2,0.3), newdata = data.frame(x = c(1,2,3)))
```

```
predict(model, type = "sim", newdata = data.frame(x = c(1,2,3)))
par(mfrow = c(1,2)); plot(model, ask = FALSE)
```

---

iMqr *M-Quantile Regression Coefficients Modeling*

---

## Description

This function implements Frumento and Salvati's (2020) method for M-quantile regression coefficients modeling (Mqrcm). M-quantile regression coefficients are described by parametric functions of the order of the quantile.

## Usage

```
iMqr(formula, formula.p = ~ slp(p,3), weights, data, s,
  psi = "Huber", plim = c(0,1), tol = 1e-6, maxit)
```

## Arguments

| | |
|---|---|
| `formula` | a two-sided formula of the form `y ~ x1 + x2 + ...`: a symbolic description of the M-quantile regression model. |
| `formula.p` | a one-sided formula of the form `~ b1(p, ...) + b2(p, ...) + ...`, describing how M-quantile regression coefficients depend on `p`, the order of the quantile. |
| `weights` | an optional vector of weights to be used in the fitting process. The weights will always be normalized to sum to the sample size. This implies that, for example, using double weights will *not* halve the standard errors. |
| `data` | an optional data frame, list or environment containing the variables in `formula`. |
| `s` | an optional 0/1 matrix that permits excluding some model coefficients (see 'Examples'). |
| `psi` | a character string indicating the 'psi' function. Currently, only '`Huber`' is implemented. |
| `plim` | the extremes of the estimation interval. You may want to model the M-quantile regression coefficients in an interval, say, `(a,b)` instead of `(0,1)`. |
| `tol` | convergence criterion for numerical optimization. |
| `maxit` | maximum number of iterations. |

## Details

A linear model is used to describe the p-th conditional M-quantile:

$$M(p|x) = \beta_0(p) + \beta_1(p)x_1 + \beta_2(p)x_2 + \dots.$$

Assume that each M-quantile regression coefficient can be expressed as a parametric function of $p$ of the form:

$$\beta(p|\theta) = \theta_0 + \theta_1 b_1(p) + \theta_2 b_2(p) + \dots$$

where $b_1(p), b_2(p, \ldots)$ are known functions of $p$. If $q$ is the dimension of $x = (1, x_1, x_2, \ldots)$ and $k$ is that of $b(p) = (1, b_1(p), b_2(p), \ldots)$, the entire M-conditional quantile function is described by a $q \times k$ matrix $\theta$ of model parameters.

Users are required to specify two formulas: `formula` describes the regression model, while `formula.p` identifies the 'basis' $b(p)$. By default, `formula.p = ~ slp(p, k = 3)`, a 3rd-degree shifted Legendre polynomial (see `slp`). Any user-defined function $b(p, \ldots)$ can be used, see 'Examples'.

Estimation of $\theta$ is carried out by minimizing an integrated loss function, corresponding to the integral, over $p$, of the loss function of standard M-quantile regression. This motivates the acronym `iMqr` (integrated M-quantile regression). The scale parameter `sigma` is estimated as the minimizer of the log-likelihood of a Generalized Asymmetric Least Informative distribution (Bianchi et al 2017), and is "modeled" as a piecewise-constant function of the order of the quantile.

## Value

An object of class "`iMqr`", a list containing the following items:

| | |
|---|---|
| `coefficients` | a matrix of estimated model parameters describing the fitted M-quantile function. |
| `plim` | a vector of two elements indicating the range of estimation. |
| `call` | the matched call. |
| `converged` | logical. The convergence status. |
| `n.it` | the number of iterations. |
| `obj.function` | the value of the minimized integrated loss function. |
| `s` | the used '`s`' matrix. |
| `psi` | the used '`psi`' function. |
| `covar` | the estimated covariance matrix. |
| `mf` | the model frame used. |
| `PDF`, `CDF` | the fitted values of the conditional probability density function (PDF) and cumulative distribution function (CDF). The CDF value should be interpreted as the order of the M-quantile that corresponds to the observed `y` variable, while the PDF is just the first derivative of the CDF. |

Use `summary.iMqr`, `plot.iMqr`, and `predict.iMqr` for summary information, plotting, and predictions from the fitted model. The generic accessory functions `coefficients`, `formula`, `terms`, `model.matrix`, `vcov` are available to extract information from the fitted model.

## Author(s)

Paolo Frumento <paolo.frumento@unipi.it>

## References

Frumento, P., Salvati, N. (2020). *Parametric modeling of M-quantile regression coefficient functions with application to small area estimation*, Journal of the Royal Statistical Society, Series A, 183(1), p. 229-250.

Bianchi, A., et al. (2018). *Estimation and testing in M-quantile regression with application to small area estimation*, International Statistical Review, 0(0), p. 1-30.

## See Also

summary.iMqr, plot.iMqr, predict.iMqr, for summary, plotting, and prediction, and plf and slp that may be used to define $b(p)$ to be a piecewise linear function and a shifted Legendre polynomial basis, respectively.

## Examples

```
##### Using simulated data in all examples
##### NOTE 1: the true quantile and M-quantile functions do not generally coincide
##### NOTE 2: the true M-quantile function is usually unknown, even with simulated data


##### Example 1

n <- 250
x <- runif(n)
y <- rnorm(n, 1 + x, 1 + x)
# true quantile function: Q(p | x) = beta0(p) + beta1(p)*x, with
  # beta0(p) = beta1(p) = 1 + qnorm(p)

# fit the 'true' model: b(p) = (1 , qnorm(p))
m1 <- iMqr(y ~ x, formula.p = ~ I(qnorm(p)))
# the fitted M-quantile regression coefficient functions are
  # beta0(p) = m1$coef[1,1] + m1$coef[1,2]*qnorm(p)
  # beta1(p) = m1$coef[2,1] + m1$coef[2,2]*qnorm(p)

# a basis b(p) = (1, p), i.e., beta(p) is assumed to be a linear function of p
m2 <- iMqr(y ~ x, formula.p = ~ p)

# a 'rich' basis b(p) = (1, p, p^2, log(p), log(1 - p))
m3 <- iMqr(y ~ x, formula.p = ~ p + I(p^2) + I(log(p)) + I(log(1 - p)))

# 'slp' creates an orthogonal spline basis using shifted Legendre polynomials
m4 <- iMqr(y ~ x, formula.p = ~ slp(p, k = 3)) # note that this is the default

# 'plf' creates the basis of a piecewise linear function
m5 <- iMqr(y ~ x, formula.p = ~ plf(p, knots = c(0.1,0.9)))


summary(m1)
summary(m1, p = c(0.25,0.5,0.75))
par(mfrow = c(1,2)); plot(m1, ask = FALSE)
# see the documentation for 'summary.iMqr' and 'plot.iMqr'




##### Example 2 ### excluding coefficients

n <- 250
x <- runif(n)
qy <- function(p,x){(1 + qnorm(p)) + (1 + log(p))*x}
```

```
# true quantile function: Q(p | x) = beta0(p) + beta1(p)*x, with
  # beta0(p) = 1 + qnorm(p)
  # beta1(p) = 1 + log(p)

y <- qy(runif(n), x) # to generate y, plug uniform p in qy(p,x)
iMqr(y ~ x, formula.p = ~ I(qnorm(p)) + I(log(p)))

# I would like to exclude log(p) from beta0(p), and qnorm(p) from beta1(p)
# I set to 0 the corresponding entries of 's'

s <- rbind(c(1,1,0),c(1,0,1))
iMqr(y ~ x, formula.p = ~ I(qnorm(p)) + I(log(p)), s = s)
```

---

plf *Basis of a Piecewise Linear Function*

---

## Description

Generates $b_1(p), b_2(p), \ldots$ such that, for `0 < p < 1`,

$$\theta_1 * b_1(p) + \theta_2 * b_2(p) + \ldots$$

is a piecewise linear function with slopes $(\theta_1, \theta_2, \ldots)$.

## Usage

```
plf(p, knots)
```

## Arguments

| | |
|---|---|
| p | a numeric vector of values between 0 and 1. |
| knots | a set of *internal* knots between 0 and 1. It can be NULL for no internal knots. |

## Details

This function permits computing a piecewise linear function on the unit interval. A different slope holds between each pair of knots, and the function is continuous at the knots.

## Value

A matrix with one row for each element of p, and `length(knots) + 1` columns. The knots are returned as `attr(, "knots")`. Any linear combination of the basis matrix is a piecewise linear function where each coefficient represents the slope in the corresponding sub-interval (see 'Examples').

## Note

This function is typically used within a call to iMqr. A piecewise linear function can be used to describe how M-quantile regression coefficients depend on the order of the quantile.

## Author(s)

Paolo Frumento <paolo.frumento@unipi.it>

## See Also

slp, for shifted Legendre polynomials.

## Examples

```
p <- seq(0,1, 0.1)

a1 <- plf(p, knots = NULL) # returns p

a2 <- plf(p, knots = c(0.2,0.7))
plot(p, 3 + 1*a2[,1] - 1*a2[,2] + 2*a2[,3], type = "l")
  # intercept = 3; slopes = (1,-1,2)
```

---

# plot.iMqr *Plot M-Quantile Regression Coefficients*

---

## Description

Plots M-quantile regression coefficients $\beta(p)$ as a function of $p$, based on a fitted model of class "iMqr".

## Usage

```
## S3 method for class 'iMqr'
plot(x, conf.int = TRUE, polygon = TRUE, which = NULL, ask = TRUE, ...)
```

## Arguments

| | |
|---|---|
| x | an object of class "iMqr", typically the result of a call to iMqr. |
| conf.int | logical. If TRUE, asymptotic 95% confidence intervals are added to the plot. |
| polygon | logical. If TRUE, confidence intervals are represented by shaded areas via polygon. Otherwise, dashed lines are used. |
| which | an optional numerical vector indicating which coefficient(s) to plot. If which = NULL, all coefficients are plotted. |
| ask | logical. If which = NULL and ask = TRUE (the default), you will be asked interactively which coefficients to plot. |
| ... | additional graphical parameters, that can include xlim, ylim, xlab, ylab, col, lwd, cex.lab, cex.axis, axes, frame.plot. See par. |

## Details

Using iMqr, each M-quantile regression coefficient $\beta(p)$ is described by a linear combination of known parametric functions of $p$. With this command, a plot of $\beta(p)$ versus $p$ is created.

## Author(s)

Paolo Frumento <paolo.frumento@unipi.it>

## See Also

iMqr for model fitting; summary.iMqr and predict.iMqr for model summary and prediction.

## Examples

```
# using simulated data

n <- 250
x <- runif(n)
qy <- function(p,x){p^2 + x*log(p)}
# true quantile function: Q(p | x) = beta0(p) + beta1(p)*x, with
   # beta0(p) = p^2
   # beta1(p) = log(p)
y <- qy(runif(n), x) # to generate y, plug uniform p in qy(p,x)

par(mfrow = c(1,2))
plot(iMqr(y ~ x, formula.p = ~ slp(p,3)), ask = FALSE)
# flexible fit with shifted Legendre polynomials
```

---

# predict.iMqr *Prediction After M-Quantile Regression Coefficients Modeling*

---

## Description

Predictions from an object of class "iMqr".

## Usage

```
## S3 method for class 'iMqr'
predict(object, type = c("beta", "CDF", "QF", "sim"), newdata, p, se = TRUE, ...)
```

## Arguments

| | |
|---|---|
| object | an object of class "iMqr", the result of a call to iMqr. |
| type | a character string specifying the type of prediction. See 'Details'. |
| newdata | an optional data frame in which to look for variables with which to predict. If omitted, the data are used. For type = "CDF", it must include the response variable. Ignored if type = "beta". |
| p | a numeric vector indicating the order(s) of the quantile to predict. Only used if type = "beta" or type = "QF". |
| se | logical. If TRUE (the default), standard errors of the prediction will be computed. Only used if type = "beta" or type = "QF". |
| ... | for future methods. |

## Details

Using iMqr, M-quantile regression coefficients $\beta(p)$ are modeled as parametric functions of $p$, the order of the quantile. This implies that the model parameter is *not* $\beta(p)$ itself. The function predict.iqr permits computing $\beta(p)$ and other quantities of interest, as detailed below.

- if type = "beta" (the default), $\beta(p)$ is returned at the supplied value(s) of p. If p is missing, a default p = (0.01, ..., 0.99) is used.
- if type = "CDF", the value of the fitted CDF (cumulative distribution function) and PDF (probability density function) are computed. The CDF value should be interpreted as the order of the M-quantile that corresponds to the observed y values, while the PDF is just the first derivative of the CDF.
- if type = "QF", the fitted values $x'\beta(p)$, corresponding to the conditional M-quantile function, are computed at the supplied values of p.
- if type = "sim", data are simulated from the fitted model. To simulate the data, the fitted conditional M-quantile function is computed at randomly generated p following a Uniform(0,1) distribution. CAUTION: this generates data assuming that the model describes the *quantile* function, while in practice it describes M-quantiles.

## Value

- if type = "beta" a list with one item for each covariate in the model. Each element of the list is a data frame with columns (p, beta, se, low, up) reporting $\beta(p)$, its estimated standard error, and the corresponding 95% confidence interval. If se = FALSE, the last three columns are not computed.
- if type = "CDF", a two-columns data frame (CDF,PDF).
- if type = "QF" and se = FALSE, a data frame with one row for each observation, and one column for each value of p. If se = TRUE, a list of two data frames, fit (predictions) and se.fit (standard errors).
- if type = "sim", a vector of simulated data.

## Note

Prediction may generate quantile crossing if the support of the new covariates values supplied in newdata is different from that of the observed data.

## Author(s)

Paolo Frumento <paolo.frumento@unipi.it>

## See Also

iMqr, for model fitting; summary.iMqr and plot.iMqr, for summarizing and plotting iMqr objects.

## Examples

```
# using simulated data

n <- 250
x <- runif(n)
y <- rlogis(n, 1 + x, 1 + x)
# true quantile function: Q(p | x) = beta0(p) + beta1(p)*x, with
  # beta0(p) = beta1(p) = 1 + log(p/(1 - p))

model <- iMqr(y ~ x, formula.p = ~ I(log(p)) + I(log(1 - p)))
# (fit asymmetric logistic distribution)


# predict beta(0.25), beta(0.5), beta(0.75)
predict(model, type = "beta", p = c(0.25,0.5, 0.75))

# predict the CDF and the PDF at new values of x and y
predict(model, type = "CDF", newdata = data.frame(x = c(.1,.2,.3), y = c(1,2,3)))

# computes the quantile function at new x, for p = (0.25,0.5,0.75)
predict(model, type = "QF", p = c(0.25,0.5,0.75), newdata = data.frame(x = c(.1,.2,.3)))

# simulate data from the fitted model
ysim <- predict(model, type = "sim") # 'newdata' can be supplied
# NOTE: data are generated using the fitted M-quantile function as if
  # it was a quantile function. This means that the simulated data will
  # have quantiles (and not M-quantiles) described by the fitted model.
  # There is no easy way to generate data with a desired M-quantile function.
```

---

`psi` *Generate Various Influence Functions for M-Quantiles*

---

## Description

Influence function to be passed to `iMqr`.

## Usage

```
Huber(c = 1.345)
```

## Arguments

| | |
|---|---|
| `c` | tuning parameter for Huber's influence function. |

## Details

These functions are only meant to be used used within a call to `iMqr`.

## Value

A list with the following items:

`psi`, `psi_tau`, `psi1_tau`, `rho_tau`
: define the influence function.

`par`
: the parameters of the influence function, e.g., the value of `c` in Huber's function.

`name`
: a character string indicating the name of the influence function.

## References

Huber, P. J. (1981). "Robust Statistics", John Wiley and Sons, New York.

## See Also

`iMqr`

## Examples

```
# The following are identical:
# iMqr(y ~ x, psi = "Huber")
# iMqr(y ~ x, psi = Huber)
# iMqr(y ~ x, psi = Huber(c = 1.345))
```

---

# slp *Shifted Legendre Polynomials*

---

## Description

Computes shifted Legendre polynomials.

## Usage

```
slp(p, k = 3, intercept = FALSE)
```

## Arguments

`p`
: the variable for which to compute the polynomials. Must be `0 <= p <= 1`.

`k`
: the degree of the polynomial.

`intercept`
: logical. If `TRUE`, the polynomials include the constant term.

## Details

Shifted Legendre polynomials (SLP) are orthogonal polynomial functions in (0,1) that can be used to build a spline basis, typically within a call to `iMqr`. The constant term is omitted unless `intercept = TRUE`: for example, the first two SLP are `(2*p - 1, 6*p^2 - 6*p + 1)`, but `slp(p, k = 2)` will only return `(2*p, 6*p^2 - 6*p)`.

## Value

An object of class "slp", i.e., a matrix with the same number of rows as p, and with k columns named slp1, slp2, ... containing the SLP of the corresponding orders. The value of k is reported as attribute.

## Note

The default for iMqr is formula.p = ~ slp(p, k = 3).

## Author(s)

Paolo Frumento <paolo.frumento@unipi.it>

## References

Refaat El Attar (2009), *Legendre Polynomials and Functions*, CreateSpace, ISBN 978-1-4414-9012-4.

## See Also

plf, for piecewise linear functions in the unit interval.

## Examples

```
p <- seq(0,1,0.1)
slp(p, k = 1) # = 2*p
slp(p, k = 1, intercept = TRUE) # = 2*p - 1 (this is the true SLP of order 1)
slp(p, k = 3) # a linear combination of (p, p^2, p^3), with slp(0,k) = 0
```

---

# summary.iMqr *Summary After M-Quantile Regression Coefficients Modeling*

---

## Description

Summary of an object of class "iMqr".

## Usage

```
## S3 method for class 'iMqr'
summary(object, p, cov = FALSE, ...)
```

## Arguments

| | |
|---|---|
| object | an object of class "iMqr", the result of a call to iMqr. |
| p | an optional vector of quantiles. |
| cov | logical. If TRUE, the covariance matrix of $\beta(p)$ is reported. Ignored if p is missing. |
| ... | for future methods. |

## Details

If p is missing, a summary of the fitted model is reported. This includes the estimated coefficients, their standard errors, and other summaries (see 'Value'). If p is supplied, the M-quantile regression coefficients of order p are extrapolated and summarized.

## Value

If p is supplied, a standard summary of the estimated M-quantile regression coefficients is returned for each value of p. If `cov = TRUE`, the covariance matrix is also reported.

If p is missing (the default), a list with the following items:

| | |
|---|---|
| `converged` | logical value indicating the convergence status. |
| `n.it` | the number of iterations. |
| `n` | the number of observations. |
| `free.par` | the number of free parameters in the model. |
| `coefficients` | the matrix of estimated coefficients. Each row corresponds to a covariate, while each column corresponds to an element of $b(p)$, the set of functions that describe how M-quantile regression coefficients vary with the order of the quantile. See 'Examples'. |
| `se` | the estimated standard errors. |
| `test.x` | Wald test for the covariates. Each *row* of `coefficients` is tested for nullity. |
| `test.p` | Wald test for the building blocks of the quantile function. Each *column* of `coefficients` is tested for nullity. |
| `obj.function` | the minimized loss function. |
| `call` | the matched call. |

## Author(s)

Paolo Frumento <paolo.frumento@unipi.it>

## See Also

`iMqr`, for model fitting; `predict.iMqr` and `plot.iMqr`, for predicting and plotting objects of class "`iMqr`".

## Examples

```
# using simulated data

set.seed(1234); n <- 250
x1 <- rexp(n)
x2 <- runif(n)
qy <- function(p,x){qnorm(p)*(1 + x)}
# true quantile function: Q(p | x) = beta0(p) + beta1(p)*x, with
   # beta0(p) = beta1(p) = qnorm(p)

y <- qy(runif(n), x1) # to generate y, plug uniform p in qy(p,x)
```

```
                    # note that x2 does not enter

model <- iMqr(y ~ x1 + x2, formula.p = ~ I(qnorm(p)) + p + I(p^2))
# beta(p) is modeled by linear combinations of b(p) = (1, qnorm(p),p,p^2)

summary(model)
# interpretation:
  # beta0(p) = model$coef[1,]*b(p)
  # beta1(p) = model$coef[2,]*b(p); etc.
# x2 and (p, p^2) are not significant


summary(model, p = c(0.25, 0.75)) # summary of beta(p) at selected quantiles
```

# Index

∗ **array**
plf, 6
∗ **methods**
plot.iMqr, 7
predict.iMqr, 8
summary.iMqr, 12
∗ **models**
iMqr, 3
psi, 10
∗ **package**
Mqrcm-package, 2
∗ **regression**
iMqr, 3
∗ **smooth**
slp, 11

Huber *(*psi*)*, 10

iMqr, *2*, 3, *6–13*

Mqrcm-package, 2

par, *7*
plf, *2*, *5*, 6, *12*
plot.iMqr, *2*, *4, 5*, 7, *9*, *13*
predict.iMqr, *2*, *4, 5*, *8*, 8, *13*
psi, 10

slp, *2*, *4, 5*, *7*, 11
summary.iMqr, *2*, *4, 5*, *8, 9*, 12