Package ‘mqqr’

June 1, 2026

Type Package

Title Multivariate Quantile-on-Quantile Regression

Version 1.0.0

Description Implements Multivariate Quantile-on-Quantile Regression (m-QQR) of Sinha, Ghosh, Hussain, Nguyen and Das (2023) <doi:10.1016/j.eneco.2023.107021>, extending the bivariate Quantile-on-Quantile regression of Sim and Zhou (2015) <doi:10.1016/j.jbankfin.2015.01.013> to include exogenous moderators and controls with optional interaction terms. For each pair of quantile levels (theta of the response and tau of the regressor) the package fits a locally-weighted quantile regression of y on the principal regressor x, a lagged dependent variable, moderators Z and the x*Z interaction terms, using Gaussian kernel weights on the empirical cumulative distribution function (CDF) distance. Bootstrap standard errors and Koenker-Machado pseudo R-squared are reported. Visualisations include 'MATLAB'-style 'Parula' and 'Jet' 3D surfaces, heatmaps and contour plots through 'plotly'.

License GPL-3

Encoding UTF-8

RoxygenNote 7.3.1

Depends R (>= 3.5.0)

Imports quantreg (>= 5.0), plotly (>= 4.0.0), stats, utils, grDevices

Suggests knitr, rmarkdown, testthat (>= 3.0.0)

URL `https://github.com/merwanroudane/multiqqr`

BugReports `https://github.com/merwanroudane/multiqqr/issues`

Config/testthat/edition 3

NeedsCompilation no

Author Merwan Roudane [aut, cre, cph],
Avik Sinha [ctb] (Original m-QQR methodology),
Nicholas Sim [ctb] (Original QQ regression methodology),
Hongtao Zhou [ctb] (Original QQ regression methodology)

Maintainer Merwan Roudane <merwanroudane920@gmail.com>

Repository CRAN

Date/Publication 2026-06-01 09:00:26 UTC

Contents

---

mqqr-package *mqqr: Multivariate Quantile-on-Quantile Regression*

Description

Implements the multivariate Quantile-on-Quantile regression of Sinha et al. (2023) extending the bivariate QQR of Sim and Zhou (2015) with exogenous moderators and optional interaction terms.

Main functions

- `mqq_regression` – the multivariate QQR estimator.
- `plot_mqq_3d`, `plot_mqq_heatmap`, `plot_mqq_contour`, `plot_mqq_interaction` – visualisations.
- `mqq_to_matrix`, `mqq_export`, `mqq_statistics` – helpers.
- `parula_colors`, `mqqr_colorscales` – colour palettes.

Author(s)

Dr Merwan Roudane <merwanroudane920@gmail.com>
GitHub: https://github.com/merwanroudane/multiqqr

mqq_regression *Multivariate Quantile-on-Quantile Regression*

Description

Fits a multivariate Quantile-on-Quantile regression with optional moderators and interaction terms.

Usage

```
mqq_regression(y, x, moderators = list(),
               y_quantiles = seq(0.05, 0.95, by = 0.05),
               x_quantiles = seq(0.05, 0.95, by = 0.05),
               bandwidth = 0.05, include_lag = TRUE,
               interactions = TRUE, se = c("bootstrap", "none"),
               n_boot = 200, cdf_based_kernel = TRUE,
               x_name = "X", y_name = "Y",
               verbose = TRUE, seed = 42)
```

Arguments

`y`	Numeric vector. Dependent variable.
`x`	Numeric vector. Principal regressor.
`moderators`	Named list of numeric vectors.
`y_quantiles`	Numeric vector of theta in (0, 1).
`x_quantiles`	Numeric vector of tau in (0, 1).
`bandwidth`	Kernel bandwidth on the empirical-CDF scale.
`include_lag`	Include y_{t-1} as a control.
`interactions`	Include x*Z cross-terms.
`se`	Standard errors: `"bootstrap"` or `"none"`.
`n_boot`	Bootstrap replicates.
`cdf_based_kernel`	Use CDF-distance kernel.
`x_name, y_name`	Variable names for printing.
`verbose`	Print progress.
`seed`	RNG seed for bootstrap.

Value

An object of class `mqq_regression`.

References

Sinha, A., Ghosh, V., Hussain, N., Nguyen, D.K., Das, N. (2023). *Energy Economics*, 126, 107021.

Sim, N., Zhou, H. (2015). *Journal of Banking and Finance*, 55, 1-12.

Examples

```
set.seed(1); n <- 200
x <- rnorm(n); z <- rnorm(n)
y <- 0.3*x + 0.2*z + 0.1*x*z + rnorm(n, sd=0.4)
fit <- mqq_regression(y, x, list(Z = z), n_boot = 30, verbose = FALSE)
print(fit)
```

---

mqq_to_matrix *Helpers for m-QQR results*

Description

Pivot, export and summarise m-QQR results.

Usage

```
mqq_to_matrix(mqq_result, value = "beta1", moderator = NULL)
mqq_export(mqq_result, prefix, digits = 4)
mqq_statistics(mqq_result, alpha = 0.05)
```

Arguments

mqq_result	An mqq_regression object.
value	Column to pivot: "beta1", "se", "t_value", "p_value", "r_squared", "gamma", "alpha".
moderator	Moderator name when value is gamma or alpha.
prefix	Output-file prefix (three CSVs written).
digits	Rounding digits.
alpha	Significance threshold.

Value

Matrix, NULL, or data frame.

Examples

```
set.seed(1); n <- 60
x <- rnorm(n); z <- rnorm(n)
y <- 0.3 * x + 0.2 * z + rnorm(n, sd = 0.4)
fit <- mqq_regression(y, x, list(Z = z),
                      y_quantiles = c(0.25, 0.5, 0.75),
                      x_quantiles = c(0.25, 0.5, 0.75),
                      n_boot = 5, verbose = FALSE)
M <- mqq_to_matrix(fit, "beta1")
mqq_statistics(fit)
mqq_export(fit, file.path(tempdir(), "mqq"))
```

`parula_colors` *MATLAB-style and Sinha-paper colour palettes*

Description

Colour palettes used by mqqr plotting functions. The default colour scale across all mqqr 3D and heatmap functions is MATLAB Parula (R2014b default), reproduced exactly from MathWorks' RGB stops.

Usage

```
parula_colors(n = 256)
matlab_jet_colors(n = 256)
turbo_colors(n = 256)
bluered_colors(n = 256)
sinha_colors(n = 256)
mqqr_palette(cols, n_breaks = 32)
resolve_colorscale(name = "Parula", n_breaks = 32)
mqqr_colorscales(show_preview = TRUE)
```

Arguments

`n`	Number of interpolated colours.
`cols`	Character vector of hex colours.
`n_breaks`	Stops used in the plotly list.
`name`	Scale name (Parula, Jet, Turbo, BlueRed, Sinha, Viridis, ...).
`show_preview`	Print the available scales.

Value

Character vector or list, depending on the function.

Examples

```
parula_colors(8)
matlab_jet_colors(8)
turbo_colors(8)
bluered_colors(8)
sinha_colors(8)
mqqr_colorscales(show_preview = FALSE)
```

plot_mqq_3d *Visualisations for m-QQR Results*

Description

3D surface, heatmap, contour and moderator-interaction plots for results of mqq_regression, defaulting to MATLAB Parula.

Usage

```
plot_mqq_3d(mqq_result, value = "beta1", moderator = NULL,
            colorscale = "Parula", show_contour = TRUE,
            x_label = "X Quantile (tau)",
            y_label = "Y Quantile (theta)",
            title = NULL)

plot_mqq_heatmap(mqq_result, value = "beta1", moderator = NULL,
                 colorscale = "Parula",
                 x_label = "X Quantile (tau)",
                 y_label = "Y Quantile (theta)",
                 title = NULL, show_stars = FALSE)

plot_mqq_contour(mqq_result, value = "beta1", moderator = NULL,
                 colorscale = "Parula",
                 x_label = "X Quantile (tau)",
                 y_label = "Y Quantile (theta)",
                 title = NULL)

plot_mqq_interaction(mqq_result, moderator,
                     value = c("gamma", "alpha"),
                     colorscale = "Parula",
                     kind = c("3d", "heatmap", "contour"), ...)
```

Arguments

- `mqq_result` An `mqq_regression` object.
- `value` Quantity to plot.
- `moderator` Moderator name (for gamma / alpha).
- `colorscale` Default "Parula".
- `show_contour`, `show_stars`, `x_label`, `y_label`, `title`, `kind`, `...` See details.

Value

A plotly object.

Examples

```
## Small toy example -- auto-tested. Plot objects are constructed but
## not rendered when run non-interactively.
set.seed(1); n <- 60
x <- rnorm(n); z <- rnorm(n)
y <- 0.4 * x + 0.2 * z + rnorm(n, sd = 0.4)
fit <- mqq_regression(y, x, list(Z = z),
                      y_quantiles = c(0.25, 0.5, 0.75),
                      x_quantiles = c(0.25, 0.5, 0.75),
                      n_boot = 5, verbose = FALSE)
p1 <- plot_mqq_3d(fit, colorscale = "Parula")
p2 <- plot_mqq_heatmap(fit, value = "beta1", show_stars = TRUE)
p3 <- plot_mqq_contour(fit, value = "beta1")
p4 <- plot_mqq_interaction(fit, "Z", value = "gamma", kind = "heatmap")
```

`qq_weights` *Numerical building blocks for QQ regression*

Description

Low-level utilities exposed for advanced users: Gaussian kernel weights on the empirical-CDF scale, and weighted quantile regression via `quantreg::rq.wfit`.

Usage

```
qq_weights(x, tau, h = 0.05, cdf_based = TRUE)
gaussian_kernel(u)
weighted_qr(y, X, tau, weights = NULL)
```

Arguments

`x, y`	Numeric vectors.
`X`	Numeric design matrix (intercept included).
`tau`	Quantile in (0, 1).
`h`	Bandwidth.
`cdf_based`	Use empirical-CDF distance kernel.
`weights`	Optional numeric weights.
`u`	Numeric vector.

Value

Numeric vector (qq_weights, gaussian_kernel) or list (weighted_qr).

Examples

```
set.seed(1)
x <- rnorm(100)
w <- qq_weights(x, tau = 0.3)
sum(w)            # weights sum to length(x)
k <- gaussian_kernel(seq(-3, 3, by = 1))
X <- cbind(1, x)
y <- 0.5 * x + rnorm(100, sd = 0.3)
fit <- weighted_qr(y, X, tau = 0.5, weights = w)
fit$coef
```

Index