Package 'tauBayesW'

September 8, 2025

Title Bayesian Weighted Quantile Regression with EM and MCMC Algorithm

Version 0.1.0

Description Implements Bayesian quantile regression approaches using the EM algorithm and several MCMC methods with observation weights for complex survey designs. Includes fast C++ implementations using 'Rcpp', 'RcppArmadillo', and 'RcppEigen'.

License MIT + file LICENSE

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.2

LinkingTo Rcpp,
RcppArmadillo,
RcppEigen

Imports Rcpp,
stats,
graphics,
methods,
plotly,
geometry,
pracma,
ggplot2,
rlang

Suggests MASS,
knitr,
rmarkdown,
grDevices,
testthat (>= 3.0.0)

VignetteBuilder knitr

Config/testthat/edition 3

SystemRequirements BLAS, LAPACK

URL https://github.com/torodriguezt/tauBayesW

BugReports https://github.com/torodriguezt/tauBayesW/issues

Contents

`as_bqr_prior` *Coerce to a* `bqr_prior` *object (unified)*

Description

Allows passing legacy list priors to `bqr.svy`. A valid list must contain at least `b0` and `B0`. Scalars/vectors are expanded as in `prior_default`. Optional fields `c0`, `C0` are relevant for `method = "ald"` and ignored otherwise.

Usage

```
as_bqr_prior(x, p, names = NULL, method = c("ald", "score", "approximate"))
```

Arguments

`x`	A `bqr_prior` or a list with components `b0`, `B0`, and optionally `c0`, `C0`.
`p`	Number of coefficients.
`names`	Optional coefficient names.
`method`	One of `"ald"`, `"score"`, `"approximate"`.

Value

A bqr_prior object.

---

as_mo_bqr_prior *Coerce to a* mo_bqr_prior *object*

Description

This function ensures that the input is converted to a valid mo_bqr_prior object. It accepts an existing mo_bqr_prior, or a list with the necessary components, and performs validation and expansion of scalar or vector inputs.

Usage

```
as_mo_bqr_prior(x, p, names = NULL)
```

Arguments

x	An object to coerce. Can be: • A valid mo_bqr_prior object • A list with components beta_mean, beta_cov, sigma_shape, and sigma_rate
p	Integer. Number of regression coefficients (including intercept). Used to validate and expand inputs.
names	Optional character vector of coefficient names. If provided, they are attached to beta_mean and beta_cov.

Value

A mo_bqr_prior object with fields:

- beta_mean: Numeric vector of prior means
- beta_cov: Prior covariance matrix
- sigma_shape: Shape parameter of inverse-gamma prior
- sigma_rate: Rate parameter of inverse-gamma prior

Examples

```
# From an existing mo_bqr_prior
prior1 <- mo_prior_default(p = 3)
as_mo_bqr_prior(prior1, p = 3)

# From a list
prior_list <- list(
  beta_mean   = c(0, 0, 0),
  beta_cov    = diag(1e6, 3),
  sigma_shape = 0.001,
  sigma_rate  = 0.001
)
as_mo_bqr_prior(prior_list, p = 3, names = c("(Intercept)", "x1", "x2"))
```

`bqr.svy` *Bayesian Weighted Quantile Regression (Survey Design)*

Description

Fits a Bayesian quantile regression model with survey weights using one of three MCMC kernels implemented in C++:

- `.MCMC_BWQR_AL` – Asymmetric Laplace Distribution
- `.MCMC_BWQR_SL` – Score likelihood
- `.MCMC_BWQR_AP` – Approximate likelihood

One or more quantiles can be estimated, depending on the input.

Usage

```
bqr.svy(
  formula,
  weights = NULL,
  data = NULL,
  quantile = 0.5,
  method = c("ald", "score", "approximate"),
  prior = NULL,
  niter = 50000,
  burnin = 10000,
  thin = 1,
  ...
)
```

Arguments

`formula`	A `formula` specifying the model.
`weights`	Optional survey weights (numeric vector or one-sided formula). Weights are passed directly to the underlying C++ algorithms without any preprocessing like scaling.
`data`	Optional `data.frame` containing the variables used in the model.
`quantile`	Numeric scalar or vector in (0, 1): target quantile(s) τ. Duplicates are automatically removed.
`method`	One of `"ald"`, `"score"`, `"approximate"`.
`prior`	Prior specification. Can be: • A `bqr_prior` object from `prior_default` • A list with components `b0`, `B0`, and optionally `c0`, `C0` • `NULL` (uses default vague priors) For `"ald"`: uses `b0`, `B0`, `c0`, `C0`. For `"score"` and `"approximate"`: uses `b0`, `B0` only.
`niter`	Integer. Number of MCMC iterations.
`burnin`	Integer. Number of burn-in iterations.
`thin`	Integer. Thinning interval.
`...`	Additional arguments passed to underlying functions (reserved for future use).

Details

Survey weights are handled differently by each method:

- `"ald"` and `"score"`: weights are normalized (divided by their mean).
- `"approximate"`: weights are used as provided (raw weights).

Prior Specification:

The prior can be specified in several ways:

1. Using `prior_default` (recommended).
2. As a list with `b0`, `B0`, and optionally `c0`, `C0`.
3. As `NULL`, in which case vague priors are used.

Multiple quantiles can be fitted in a single call. The returned object adapts its class accordingly (`"bwqr_fit"` for one quantile, `"bwqr_fit_multi"` for several).

Value

An object of class `"bqr.svy"`, containing:

`beta`	Posterior mean estimates of regression coefficients.
`draws`	Posterior draws from the MCMC sampler.
`accept_rate`	Average acceptance rate (if available).
`quantile`	The quantile(s) fitted.
`prior`	Prior specification used.
`formula`, `terms`, `model`	Model specification details.
`runtime`	Elapsed runtime in seconds.

Examples

```
# Simulate data
sim <- simulate_bqr_data(n = 100, betas = c(2, 1.5, -0.8))

# Basic usage with default priors
fit1 <- bqr.svy(y ~ x1 + x2, data = sim$data, weights = sim$weights)

# With informative priors
prior <- prior_default(
  p  = 3,
  b0 = c(2, 1.5, -0.8),
  B0 = diag(c(0.25, 0.25, 0.25)),
  c0 = 3, C0 = 2
)
fit2 <- bqr.svy(y ~ x1 + x2, data = sim$data, weights = sim$weights,
                method = "ald", prior = prior)

# Compare methods
fit_score <- bqr.svy(y ~ x1 + x2, data = sim$data, weights = sim$weights,
                     method = "score")
fit_approx <- bqr.svy(y ~ x1 + x2, data = sim$data, weights = sim$weights,
                      method = "approximate")
```

convergence_check *Check convergence diagnostics for tauBayesW model objects*

Description

This is a generic function that dispatches to specific methods depending on the class of the object (e.g., `bqr.svy`, `mo.bqr.svy`).

Usage

```
convergence_check(
  object,
  rhat_threshold = 1.1,
  ess_ratio_threshold = 0.1,
  verbose = TRUE,
  ...
)
```

Arguments

`object`	An object of class `"bqr.svy"` or `"mo.bqr.svy"`.
`rhat_threshold`	Numeric scalar. The threshold for the Gelman-Rubin $\hat{R}$ statistic above which a parameter is considered not converged. Default is `1.1`.
`ess_ratio_threshold`	Numeric scalar. The threshold for the ratio of Effective Sample Size (ESS) to total draws below which a parameter is considered to have low sampling efficiency. Default is `0.1`.
`verbose`	Logical; if `TRUE`, prints a summary of convergence diagnostics to the console. Default is `TRUE`.
`...`	Additional arguments passed to specific methods.

Value

A list containing:

`rhat` Named numeric vector of $\hat{R}$ values for each parameter (if available).

`ess_ratio` Named numeric vector of ESS ratios for each parameter (if available).

`not_converged` Character vector of parameter names failing $\hat{R}$ threshold.

`low_ess` Character vector of parameter names failing ESS ratio threshold.

Examples

```
## Not run:
fit <- bqr.svy(y ~ x, data = mydata, quantile = 0.5, method = "ald")
convergence_check(fit)

## End(Not run)
```

convergence_check.bqr.svy
Convergence diagnostics for bqr.svy *objects*

Description

Computes the rank-normalized $\hat{R}$ (Gelman Rubin) and Effective Sample Size (ESS) for parameters from MCMC output produced by bqr.svy. Works with both single-quantile fits (one τ) and multi-quantile fits.

Usage

```
## S3 method for class 'bqr.svy'
convergence_check(
  object,
  rhat_threshold = 1.1,
  ess_ratio_threshold = 0.1,
  verbose = TRUE,
  ...
)
```

Arguments

object	An object of class "bqr.svy" or "mo.bqr.svy".
rhat_threshold	Numeric scalar. The threshold for the Gelman-Rubin $\hat{R}$ statistic above which a parameter is considered not converged. Default is 1.1.
ess_ratio_threshold	Numeric scalar. The threshold for the ratio of Effective Sample Size (ESS) to total draws below which a parameter is considered to have low sampling efficiency. Default is 0.1.
verbose	Logical; if TRUE, prints a summary of convergence diagnostics to the console. Default is TRUE.
...	Additional arguments passed to specific methods.

Details

For single-quantile fits, object$draws must be a numeric matrix or data frame with rows = iterations and columns = parameters. For multi-quantile fits, object$draws must be a list of such matrices (one per τ). Diagnostics are computed via summarise_draws_custom following Vehtari et al. (2021).

Value

If a *single* τ is present: a list with components rhat, neff, ess_ratio, not_converged, low_ess, and converged. If *multiple* τ values are present: a named list where each element corresponds to a τ and has the same structure as above.

Examples

```
## Not run:
set.seed(1)
dat <- data.frame(y = rnorm(50), x = rnorm(50))
fit1 <- bqr.svy(y ~ x, data = dat, quantile = 0.5, method = "ald")
convergence_check(fit1)

fitk <- bqr.svy(y ~ x, data = dat, quantile = c(0.25, 0.5, 0.75), method = "ald")
convergence_check(fitk)

## End(Not run)
```

---

convergence_check.default

Default convergence check

Description

Allows `convergence_check()` to work on generic matrices or data frames containing MCMC draws (each column = parameter, each row = draw).

Usage

```
## Default S3 method:
convergence_check(
  object,
  rhat_threshold = 1.01,
  ess_ratio_threshold = 0.1,
  ...
)
```

Arguments

`object`	A numeric matrix or data frame of draws.
`rhat_threshold`	Numeric, threshold for R-hat diagnostic (default 1.01).
`ess_ratio_threshold`	Numeric, threshold for effective sample size ratio (default 0.10).
`...`	Not used.

Value

A list with components `rhat`, `neff`, and `converged`.

convergence_check.mo.bqr.svy *Convergence diagnostics for mo.bqr.svy objects*

Description

Checks convergence for EM algorithm fits produced by `mo.bqr.svy`. Since EM is deterministic, this method reports the number of iterations and whether convergence was achieved for each quantile.

Usage

```
## S3 method for class 'mo.bqr.svy'
convergence_check(
  object,
  rhat_threshold = 1.1,
  ess_ratio_threshold = 0.1,
  verbose = TRUE,
  ...
)
```

Arguments

`object`	An object of class `"bqr.svy"` or `"mo.bqr.svy"`.
`rhat_threshold`	Numeric scalar. The threshold for the Gelman-Rubin $\hat{R}$ statistic above which a parameter is considered not converged. Default is `1.1`.
`ess_ratio_threshold`	Numeric scalar. The threshold for the ratio of Effective Sample Size (ESS) to total draws below which a parameter is considered to have low sampling efficiency. Default is `0.1`.
`verbose`	Logical; if `TRUE`, prints a summary of convergence diagnostics to the console. Default is `TRUE`.
`...`	Additional arguments passed to specific methods.

Details

For EM-based `mo.bqr.svy` fits, no MCMC draws are available, so this function reports iteration counts and convergence flags instead of R-hat/ESS statistics.

drawQuantile1D *Draw univariate quantile curves/bands for mo.bqr.svy (d = 1)*

Description

Draw univariate quantile curves/bands for mo.bqr.svy (d = 1)

Usage

```
drawQuantile1D(
  fit,
  datafile = NULL,
  response = "Y",
  x_var = NULL,
  x_grid = NULL,
  xValue = NULL,
  paintedArea = TRUE,
  band_choice = c("minmax", "symmetric"),
  print_plot = TRUE,
  show_data = !is.null(datafile)
)
```

Arguments

fit	mo.bqr.svy object with response_dim = 1
datafile	optional data.frame to overlay observed points
response	name of Y column in datafile (character, length 1)
x_var	which predictor to map to the horizontal axis (character, length 1). If NULL, uses the first term label in fit$terms.
x_grid	numeric vector of x values for x_var; if NULL, deduced from datafile or set to seq(-2,2,len=100)
xValue	data.frame or list with fixed values for other predictors (one or more rows). If multiple rows are provided, curves are colored by row (comparison).
paintedArea	if TRUE and there are at least 2 taus, fills ribbon between two taus (see band_choice)
band_choice	character: "minmax" (default) or "symmetric". - "minmax": banda entre min(taus) y max(taus) - "symmetric": toma el mayor tau < 0.5 y el menor tau > 0.5 (si existen)
print_plot	if TRUE returns ggplot; if FALSE returns a data.frame with predictions
show_data	if TRUE and datafile/response/x_var available, shows observed points

Value

ggplot object or data.frame with columns: xid, x, tau, yhat

---

drawQuantileRegion *Draw quantile regions (2D) for mo.bqr.svy using convex hulls*

Description

This function visualizes bivariate quantile regions from a fitted `mo.bqr.svy` object. For each requested quantile level, a convex hull is computed from the directional points and drawn as a closed polygon. The function can overlay observed data, compare multiple predictor values, and facet the plot so each quantile level is displayed in its own panel.

Usage

```
drawQuantileRegion(
  fit,
  datafile = NULL,
  response = c("Y1", "Y2"),
  xValue = NULL,
  paintedArea = FALSE,
  comparison = FALSE,
  print_plot = TRUE,
  show_data = !is.null(datafile),
  facet_by_tau = TRUE,
  facet_nrow = 1,
  facet_ncol = NULL,
  facet_scales = "fixed",
  round_digits = 10
)
```

Arguments

fit	mo.bqr.svy object with response_dim = 2.
datafile	Optional data.frame with observed responses to overlay.
response	Character vector of length 2 naming the Y columns in datafile, e.g. c("Y1","Y2").
xValue	Predictor values at which to evaluate the region. Can be: • NULL: uses the mean design vector (all zeros). • data.frame: one or more rows specifying predictor settings. • list of data.frames: each element is one row of predictors.
paintedArea	Logical; if TRUE, fills the polygon; if FALSE, only draws the outline.
comparison	Logical; if TRUE and multiple xValues are provided, colors polygons by each xValue.
print_plot	Logical; if TRUE returns a **ggplot** object; if FALSE returns a data.frame with polygon coordinates.
show_data	Logical; if TRUE and valid datafile/response are provided, overlays observed points.
facet_by_tau	Logical; if TRUE, facets the plot by quantile level, creating one panel per τ.
facet_nrow	Number of rows in the facet grid.
facet_ncol	Number of columns in the facet grid (if NULL, computed automatically).
facet_scales	Scales option passed to facet_wrap ("fixed", "free", etc.).
round_digits	Integer, number of digits to round projected points before computing convex hulls (helps remove duplicates).

Details

Internally, the function computes directional projections of the fitted model via .collect_points_for_tau, applies a convex hull with grDevices::chull, and then arranges the vertices to form closed polygons. This ensures smooth and convex quantile regions. Multiple quantiles can be visualized simultaneously using faceting.

Value

Either a **ggplot2** object (if `print_plot = TRUE`) showing the quantile regions, or a `data.frame` with polygon coordinates (if `print_plot = FALSE`).

Examples

```
## Not run:
# Fit a bivariate model
fit2 <- mo.bqr.svy(cbind(Y1,Y2) ~ x, data=df2,
                   weights=df2$w, quantile=c(0.5,0.8))

# Draw quantile regions with observed points
drawQuantileRegion(fit2, datafile=df2, response=c("Y1","Y2"),
                   xValue=data.frame(x=c(-1,0,1)),
                   paintedArea=FALSE, comparison=TRUE,
                   facet_by_tau=TRUE)

## End(Not run)
```

---

drawQuantileRegion_3D *Draw quantile regions (3D) for mo.bqr.svy using convex hulls*

Description

This function visualizes trivariate quantile regions from a fitted `mo.bqr.svy` object using **plotly**. For each requested quantile level, a convex hull is computed from the directional points and plotted as a 3D mesh. When multiple quantiles are present, each one is displayed in its own subplot (grid of 3D panels).

Usage

```
drawQuantileRegion_3D(
  fit,
  xValue = NULL,
  opacity = 0.5,
  datafile = NULL,
  response = c("Y1", "Y2", "Y3"),
  show_points = FALSE,
  point_opacity = 0.25,
  point_size = 2,
  nrows = 1,
  ncols = NULL,
  show_titles = TRUE,
  round_digits = 10
)
```

Arguments

`fit`	`mo.bqr.svy` object with `response_dim = 3`.
`xValue`	Predictor values at which to evaluate the region. Can be: • `NULL`: uses the mean design vector (all zeros).

- `data.frame`: one or more rows specifying predictor settings.
- `list` of data.frames: each element is one row of predictors.

Works in the same way as in `drawQuantileRegion` (2D).

`opacity`	Opacity, between 0 and 1, of each quantile body.
`datafile`	Optional `data.frame` with observed responses to overlay.
`response`	Character vector of length 3 naming the Y columns in `datafile`, e.g. `c("Y1","Y2","Y3")`.
`show_points`	Logical; if `TRUE` and valid `datafile`/ `response` are provided, overlays observed points as a 3D scatter.
`point_opacity`	Opacity of observed points.
`point_size`	Size of observed points.
`nrows`	Number of rows in the grid of subplots (when multiple quantiles).
`ncols`	Number of columns in the grid of subplots. If `NULL`, computed automatically from the number of quantiles and `nrows`.
`show_titles`	Logical; if `TRUE`, adds facet titles (e.g. tau = 0.5) above each subplot.
`round_digits`	Integer, number of digits to round projected points before computing convex hulls (helps remove duplicates).

Details

Internally, the function computes directional projections of the fitted model via `.collect_points_for_tau`, applies a convex hull with `geometry::convhulln`, and then maps triangular faces to a mesh object. The resulting quantile bodies are plotted in interactive 3D. When multiple quantiles are present, they are arranged in a grid of subplots so each level can be inspected separately.

Value

A **plotly** object with one 3D scene per quantile, each containing a convex-hull mesh for every `xValue`, plus (optionally) the observed data points.

Examples

```
## Not run:
# Fit a trivariate model
fit3 <- mo.bqr.svy(cbind(Y1,Y2,Y3) ~ x, data=df3,
                   weights=df3$w, quantile=c(0.5,0.8), U=U)

# Visualize quantile bodies at x = -1, 0, 1
drawQuantileRegion_3D(fit3,
                      xValue=data.frame(x=c(-1,0,1)),
                      datafile=df3, response=c("Y1","Y2","Y3"),
                      show_points=TRUE)

## End(Not run)
```

`mo.bqr.svy` *Multiple-Output Bayesian Quantile Regression for Complex Surveys (Directional EM)*

Description

Fits Bayesian quantile regression models for multivariate responses using the EM algorithm and a directional approach. The method projects the response into random unit vectors (directions) and their orthogonal complements, and then fits univariate Bayesian quantile regression models along each projection. The collection of fitted directions defines the multivariate quantile region.

Usage

```
mo.bqr.svy(
  formula,
  weights = NULL,
  data,
  quantile = 0.5,
  algorithm = "em",
  prior = NULL,
  n_dir = 1,
  epsilon = 1e-06,
  max_iter = 1000,
  verbose = FALSE,
  gamma_prior_var = 1e+06,
  ...
)
```

Arguments

`formula`	A formula object specifying the model.
`weights`	Optional vector of sampling weights. If `NULL`, equal weights are used.
`data`	A data frame containing the variables in the model.
`quantile`	Numeric vector of quantile levels (between 0 and 1, exclusive).
`algorithm`	Character string specifying the algorithm. Currently only `"em"` is supported.
`prior`	Prior specification. Can be: • `NULL`: Default priors are used for all quantiles • A single `mo_bqr_prior` object: Recycled for all quantiles • A list of `mo_bqr_prior` objects: One prior per quantile • A function `f(tau, p, names)`: Generates quantile-specific priors
`n_dir`	Integer. Number of projection directions (if directions `U` are not supplied).
`epsilon`	Convergence tolerance for the EM algorithm.
`max_iter`	Maximum number of EM iterations.
`verbose`	Logical indicating whether to print progress messages.
`gamma_prior_var`	Numeric. Prior variance for the gamma coefficients associated with orthogonal complements.

`...` Additional arguments for direction specification:

U Optional user-specified matrix of directions ($d \times K$). If not provided, `n_dir` random unit vectors are generated automatically.

Details

The algorithm works by drawing or receiving as input a set of unit directions $u_k \in \mathbb{R}^d$. For each direction, an orthonormal basis of its orthogonal complement Γ_k is computed using `pracma::nullspace`. The response Y is then projected into the pair (u_k, Γ_k), and a Bayesian quantile regression is fitted along that direction using the EM algorithm. Results across all directions can be combined to approximate the multivariate quantile region.

Prior distributions can be specified globally or quantile-specific. When a list of priors is provided, elements can be named using either `"q0.1"` format or `"0.1"` format to match specific quantiles. When a function is provided, it will be called with `(tau, p, names)` for each quantile level.

Value

An object of class `"mo.bqr.svy"` containing:

`call`	The matched call
`formula`	The model formula
`terms`	The terms object
`quantile`	Vector of fitted quantiles
`algorithm`	Algorithm used
`prior`	List of priors used for each quantile
`fit`	List of fitted results for each quantile, each containing one sub-list per direction
`coefficients`	Coefficients from the first quantile
`n_dir`	Number of directions
`U`	Matrix of projection directions ($d \times K$)
`Gamma_list`	List of orthogonal complement bases, one per direction
`n_obs`	Number of observations
`n_vars`	Number of covariates
`response_dim`	Dimension of the response d

Examples

```
# Datos simulados para el ejemplo
set.seed(1)
n  <- 150
x1 <- runif(n,-1,1)
x2 <- rnorm(n)
y  <- 1 + 2*x1 + 0.5*x2 + rnorm(n)
mydata <- data.frame(y, x1, x2)

# Basic usage with default priors
fit1 <- mo.bqr.svy(y ~ x1 + x2, data = mydata,
                   quantile = c(0.1, 0.5, 0.9))

# Using quantile-specific priors via function
prior_fn <- function(tau, p, names) {
```

```
  variance <- ifelse(tau < 0.2 | tau > 0.8, 0.1, 1.0)
  mo_prior_default(p = p, beta_cov = diag(variance, p), names = names)
}
fit2 <- mo.bqr.svy(y ~ x1 + x2, data = mydata,
                   quantile = c(0.1, 0.5, 0.9), prior = prior_fn)

# Explicit control of directions
set.seed(1)
y1 <- 1 + 2*x1 + 0.5*x2 + rnorm(n)
y2 <- -1 + 1.5*x1 + rnorm(n)
y3 <- 0.5 - x2 + rnorm(n)
mydata <- data.frame(y1, y2, y3, x1, x2)
U <- matrix(rnorm(9), nrow = 3)  # d=3, K=2
U <- apply(U, 2, function(v) v / sqrt(sum(v^2))) # normalize
fit3 <- mo.bqr.svy(cbind(y1,y2,y3) ~ x1 + x2, data = mydata,
                   quantile = 0.5, U = U)
```

`mo_prior_default` *Default prior for Multiple-Output BQR (EM)*

Description

Creates a prior object (class `"mo_bqr_prior"`) for `mo.bqr.svy`. The regression coefficients have a multivariate normal prior, and the noise variance has an inverse-gamma prior specified via shape/rate. This function can be used to create individual priors that can then be assigned to specific quantiles, allowing for quantile-specific prior distributions.

Usage

```
mo_prior_default(
  p,
  beta_mean = rep(0, p),
  beta_cov = diag(1e+06, p),
  sigma_shape = 0.001,
  sigma_rate = 0.001,
  names = NULL
)
```

Arguments

`p`	Number of regression coefficients (including the intercept).
`beta_mean`	Numeric vector of length `p`. A scalar is expanded to length `p`.
`beta_cov`	Prior covariance for coefficients. May be: • a `p` x `p` matrix, • a scalar (expanded to `diag(scalar, p)`), or • a length-`p` vector (expanded to `diag(vector)`).
`sigma_shape`	Positive scalar (shape of the inverse-gamma prior on σ^2).
`sigma_rate`	Positive scalar (rate of the inverse-gamma prior on σ^2).
`names`	Optional coefficient names to attach to `beta_mean` and `beta_cov`.

Details

When used with mo.bqr.svy, you can specify different priors for each quantile by providing a list of mo_bqr_prior objects, a function that takes (tau, p, names) as arguments, or use a single prior that will be recycled across all quantiles.

Value

An object of class "mo_bqr_prior" with fields beta_mean, beta_cov, sigma_shape, and sigma_rate.

Examples

```
# Create a single prior (will be recycled for all quantiles)
prior1 <- mo_prior_default(p = 3, beta_mean = c(0, 1, -0.5))

# Create quantile-specific priors using a list
priors_list <- list(
  q0.1 = mo_prior_default(p = 3, beta_mean = c(0, 0.8, -0.3)),
  q0.5 = mo_prior_default(p = 3, beta_mean = c(0, 1.0, -0.5)),
  q0.9 = mo_prior_default(p = 3, beta_mean = c(0, 1.2, -0.7))
)

# Create quantile-specific priors using a function
prior_fn <- function(tau, p, names) {
  # More informative priors for extreme quantiles
  variance <- ifelse(tau < 0.2 | tau > 0.8, 0.5, 1.0)
  mo_prior_default(p = p, beta_cov = diag(variance, p), names = names)
}
```

---

plot.bqr.svy *Plot method for* bqr.svy *objects*

Description

Plot method for bqr.svy objects

Usage

```
## S3 method for class 'bqr.svy'
plot(x, type = c("trace", "intervals", "quantiles"), ...)
```

Arguments

x	An object of class bqr.svy.
type	Type of plot: "trace", "intervals", or "quantiles".
...	Additional plotting arguments.

`plot_quantile.bqr.svy` *Plot predicted quantile regression curve for bqr.svy objects*

Description

This function plots the predicted quantile regression curve from a fitted `bqr.svy` model. It can handle both numeric and categorical predictors and optionally overlays the curve on an existing plot.

Usage

```
plot_quantile.bqr.svy(
  object,
  data,
  predictor,
  grid_length = 100,
  fixed_values = NULL,
  add = FALSE,
  line_col = "red",
  line_lwd = 2,
  line_type = 1,
  point_pch = 19,
  point_cex = 1.2,
  prefer_predict = FALSE,
  main = NULL,
  ...
)
```

Arguments

`object`	An object of class `bqr.svy`, typically the result of a call to `bqr.svy`.
`data`	A `data.frame` containing the variables used in the model. Must include the predictor specified in `predictor` and any covariates in the fitted model.
`predictor`	A character string giving the name of the predictor variable to plot on the x-axis.
`grid_length`	Integer; number of grid points to generate for continuous predictors. Ignored for categorical predictors.
`fixed_values`	Optional named list giving fixed values for covariates other than `predictor`. If not supplied, numeric covariates are fixed at their median and factors at their most frequent level.
`add`	Logical; if `TRUE`, adds the curve to an existing plot instead of creating a new one.
`line_col`	Color for the regression line.
`line_lwd`	Line width for the regression line.
`line_type`	Line type for the regression line.
`point_pch`	Plotting symbol to use for points (categorical predictors).
`point_cex`	Size of the plotting symbols for points (categorical predictors).
`prefer_predict`	Logical; if `TRUE`, attempts to use the `predict()` method for the fitted object first.
`main`	Main title for the plot. If `NULL`, a default is constructed.
`...`	Additional graphical parameters passed to `plot`, `points`, or `lines`.

Value

Invisibly returns a data.frame with:

predictor	The sequence or factor levels of the predictor.
predicted	The predicted quantile values.
quantile	The tau value used.

Examples

```
## Not run:
fit <- bqr.svy(y ~ x, data = mydata, quantile = 0.5)
plot_quantile.bqr.svy(fit, data = mydata, predictor = "x")

## End(Not run)
```

---

plot_quantile_with_points.bqr.svy

Plot observed points and predicted quantile regression curve for bqr.svy

---

Description

Plot observed points and predicted quantile regression curve for bqr.svy

Usage

```
plot_quantile_with_points.bqr.svy(object, data, predictor, main = NULL, ...)
```

Arguments

object	An object of class bqr.svy.
data	A data frame containing the variables used in the model.
predictor	Character string with the name of the predictor variable.
main	Main title for the plot.
...	Additional arguments passed to plot_quantile.bqr.svy.

print.bqr.svy *Print method for bqr.svy objects*

Description

Print method for bqr.svy objects

Usage

```
## S3 method for class 'bqr.svy'
print(x, digits = 3, ...)
```

Arguments

x	An object of class "bqr.svy".
digits	Integer, number of significant digits to print.
...	Additional arguments passed to print.

print.bwqr_fit *Print method for bwqr_fit objects*

Description

Print method for bwqr_fit objects

Usage

```
## S3 method for class 'bwqr_fit'
print(x, digits = 3, ...)
```

Arguments

x	An object of class "bwqr_fit".
digits	Integer, number of significant digits to print.
...	Additional arguments passed to print.

print.mo.bqr.svy *Print method for mo.bqr.svy objects*

Description

Displays the fitted call, quantiles, and estimated coefficients per direction.

Usage

```
## S3 method for class 'mo.bqr.svy'
print(x, ...)
```

Arguments

x	An object of class "mo.bqr.svy".
...	Additional arguments (not used). Print method for mo.bqr.svy objects

print.mo_bqr_prior *Print method for* mo_bqr_prior *objects*

Description

This function defines the print() method for objects of class mo_bqr_prior. It displays a summary of the prior structure, including the length of beta_mean, the dimension of beta_cov, and the values of sigma_shape and sigma_rate.

Usage

```
## S3 method for class 'mo_bqr_prior'
print(x, ...)
```

Arguments

x	An object of class mo_bqr_prior.
...	Additional arguments passed to or from other methods (currently unused).

Value

The object x, invisibly.

Examples

```
prior <- mo_prior_default(p = 3)
print(prior)
```

print.summary.bqr.svy *Print method for summary.bqr.svy objects*

Description

Print method for summary.bqr.svy objects

Usage

```
## S3 method for class 'summary.bqr.svy'
print(x, ...)
```

Arguments

x	An object of class "summary.bqr.svy".
...	Additional arguments passed to print.

print.summary.bwqr_fit *Print method for summary.bwqr_fit objects*

Description

Print method for summary.bwqr_fit objects

Usage

```
## S3 method for class 'summary.bwqr_fit'
print(x, ...)
```

Arguments

x	An object of class "summary.bwqr_fit".
...	Additional arguments passed to print.

print.summary.mo_bqr.svy *Print method for summary.mo_bqr.svy objects*

Description

Print method for summary.mo_bqr.svy objects

Usage

```
## S3 method for class 'summary.mo_bqr.svy'
print(x, ...)
```

Arguments

x	An object of class "summary.mo_bqr.svy".
...	Additional arguments (currently unused).

prior_default *Default Prior for Bayesian Weighted Quantile Regression*

Description

Creates a unified prior object (class "bqr_prior") to be passed to bqr.svy (or mo.bqr.svy). It stores a multivariate normal prior on the regression coefficients and, for the "ald" kernel, optional Inverse-Gamma hyperparameters c0, C0 for σ^2. Methods that do not use some fields simply ignore them.

Usage

```
prior_default(
  p,
  b0 = rep(0, p),
  B0 = diag(1e+06, p),
  c0 = 0.001,
  C0 = 0.001,
  names = NULL
)
```

Arguments

p	Number of regression coefficients (including the intercept).
b0	Numeric vector of prior means (length p). If a scalar is supplied, it is expanded to length p.
B0	Prior covariance. May be a p x p matrix, a scalar (expanded to diag(scalar, p)), or a length-p vector (expanded to diag(vector)).
c0	Shape parameter of the Inverse-Gamma prior for σ^2 (ALD).
C0	Scale parameter of the Inverse-Gamma prior for σ^2 (ALD).
names	Optional coefficient names to attach to b0 and B0.

Value

An object of class `"bqr_prior"` with components `b0`, `B0`, and optionally `c0`, `C0`.

`simulate_bqr_data` *Simulate data for Bayesian Weighted Quantile Regression*

Description

Generates synthetic data compatible with `bqr.svy`, allowing the user to specify true regression coefficients, error scale, and optional survey weights.

Usage

```
simulate_bqr_data(
  n = 100,
  betas = c(1, 2, -0.5),
  sigma = 1,
  weights = NULL,
  seed = NULL
)
```

Arguments

`n`	Number of observations.
`betas`	Numeric vector of true coefficients (first element is intercept).
`sigma`	Standard deviation of the Gaussian error term.
`weights`	Optional numeric vector of survey weights. If `NULL`, generated from Uniform(0.5, 2).
`seed`	Optional integer seed for reproducibility.

Value

A list with components:

`data` `data.frame` with response `y` and predictors.

`weights` Numeric vector of survey weights.

`true_betas` The true coefficients used in data generation.

Examples

```
sim <- simulate_bqr_data(n = 50, betas = c(1, 2, -1), sigma = 0.5)
head(sim$data)
```

`simulate_mo_bqr_data` *Simulate data for Multiple-Output Bayesian Quantile Regression*

Description

Generates synthetic data suitable for testing and demonstrating `mo.bqr.svy`. The function simulates predictors `X`, a response `y`, random weights, and stores the true regression coefficients used for generation.

Usage

```
simulate_mo_bqr_data(n = 100, p = 2, beta_true = NULL, seed = NULL)
```

Arguments

`n`	Integer. Number of observations to simulate.
`p`	Integer. Number of predictors (excluding the intercept).
`beta_true`	Numeric vector of length `p + 1`, giving the true regression coefficients (first element is the intercept, followed by slopes). If `NULL`, defaults to `c(1, 1, 2, ..., p)`.
`seed`	Optional integer for reproducibility.

Value

A list with the following elements:

`data` A `data.frame` with response `y` and predictors `x1, ..., xp`.

`weights` A numeric vector of length `n` with random survey weights drawn from $U(0.5, 2)$.

`true_betas` Matrix with the true regression coefficients used in the simulation (1 row, `p+1` columns).

`quantiles` Default quantile levels `c(0.1, 0.5, 0.9)`.

Examples

```
# Simulate data with 2 predictors and default coefficients
sim <- simulate_mo_bqr_data(n = 50, p = 2, seed = 123)
str(sim$data)

# Use custom true coefficients
sim2 <- simulate_mo_bqr_data(n = 50, p = 3,
                             beta_true = c(1, 2, -1, 0.5),
                             seed = 456)
sim2$true_betas
```

summary.bqr.svy *Summary method for bqr.svy objects*

Description

Summary method for bqr.svy objects

Usage

```
## S3 method for class 'bqr.svy'
summary(object, probs = c(0.025, 0.975), digits = 3, ...)
```

Arguments

object	An object of class "bqr.svy".
probs	Numeric vector of probabilities for summary statistics.
digits	Integer, number of significant digits to display.
...	Additional arguments passed to other methods.

summary.bwqr_fit *Summary method for bwqr_fit objects*

Description

Summary method for bwqr_fit objects

Usage

```
## S3 method for class 'bwqr_fit'
summary(object, probs = c(0.025, 0.975), digits = 3, max_lag = 200, ...)
```

Arguments

object	An object of class "bwqr_fit".
probs	Numeric vector of probabilities for summary statistics.
digits	Integer, number of significant digits to display.
max_lag	Integer, maximum lag for autocorrelation diagnostics.
...	Additional arguments passed to other methods.

summary.mo.bqr.svy *Summary method for Multiple-Output Bayesian Quantile Regression*

Description

Provides a tabular summary of fitted Bayesian quantile regression models estimated with mo.bqr.svy. The output contains, for each quantile and direction, the estimated coefficients, scale parameter, iteration count, and convergence status. In addition, summary attributes at the quantile level (average iterations, global convergence) are stored for use by the print method.

Usage

```
## S3 method for class 'mo.bqr.svy'
summary(object, digits = 3, ...)
```

Arguments

object	An object of class "mo.bqr.svy", typically returned by mo.bqr.svy.
digits	Integer, number of decimal places to use for rounding numeric results. Defaults to 3.
...	Additional arguments (currently unused).

Value

A summary.mo_bqr.svy object, which is a data frame containing:

- quantile: quantile level τ.
- direction: index of the direction used in the EM algorithm.
- coefficient columns: estimated regression parameters for each covariate.
- sigma: estimated scale parameter for that direction.
- iter: number of EM iterations taken for that direction.
- converged: logical indicating convergence for that direction.

Additionally, each block of rows corresponding to a quantile has two attributes:

- "conv_global": TRUE if all directions converged, FALSE otherwise.
- "iter_summary": average number of iterations across directions.

The object has class c("summary.mo_bqr.svy", "data.frame"), and its default print method displays both per-quantile summaries and per-direction details.

See Also

mo.bqr.svy, print.summary.mo_bqr.svy Summary method for Multiple-Output Bayesian Quantile Regression (doc igual que ya tienes)

summary_bqr *Generic summary function for Bayesian Quantile Regression objects*

Description

Generic summary function for Bayesian Quantile Regression objects

Usage

```
summary_bqr(object, ...)
```

Arguments

object	An object containing model fit results.
...	Additional arguments passed to other methods.

Index