Package: bbqr (via r-universe)

October 10, 2026

Type Package

Title Bayesian Quantile Regression with Lasso and Adaptive Lasso

Version 0.2.0

Description Markov chain Monte Carlo samplers for Bayesian quantile regression, based on the asymmetric Laplace distribution and the location-scale mixture representation of Kozumi and Kobayashi (2011) <doi:10.1080/00949655.2010.496117>. A binary response and an observed continuous response are both supported, each with three penalty layers behind one interface: no penalty, following Benoit and Van den Poel (2012) <doi:10.1002/jae.1216>; the Bayesian lasso, following Benoit, Al-Hamzawi and Yu (2013) <doi:10.1007/s00180-013-0439-0>; and the Bayesian adaptive lasso of Rubio Garcia (2023) <https://soar.wichita.edu/entities/publication/a2f86232-4704-4ec2-b685-751e7b04ec42>. In the binary family each is available as published and in a corrected form, the default, in which every improper prior component is replaced by a proper one so that the posterior exists unconditionally; the continuous family ships the corrected form only. The continuous adaptive-lasso layer at its default reproduces the penalty of Alhamzawi, Yu and Benoit (2012) <doi:10.1177/1471082X1101200304>. A binary threshold model identifies the coefficient vector only up to a positive scale, so the binary samplers expose the identification anchor as an explicit argument, allowing fixing the scale of the error distribution, fixing a single coefficient, and constraining the norm of the coefficient vector to be compared directly; an observed response identifies the scale, so the continuous samplers have no anchor and draw it every sweep. The MCMC cores are written in Fortran and called from R.

License GPL (>= 2)

Copyright See inst/COPYRIGHTS for the attribution of third-party components inherited from the 'bayesQR' package.

Encoding UTF-8

Depends R (>= 4.2)

Imports graphics, stats, utils

Suggests testthat (>= 3.0.0), knitr, rmarkdown, quantreg, lpSolve

Config/testthat/edition 3

VignetteBuilder knitr

NeedsCompilation yes

RoxygenNote 7.3.3

URL https://github.com/fernandorubiogarcia/bbqr

BugReports https://github.com/fernandorubiogarcia/bbqr/issues

Author Fernando Rubio Garcia [aut, cre] (Wichita State University), Mai Dao [ths] (Wichita State University), Dries F. Benoit [ctb, cph] (Author of 'bayesQR', from which the Fortran RNG wrapper and package layout are derived), Rahim Al-Hamzawi [ctb], Keming Yu [ctb], Dirk Van den Poel [ctb]

Maintainer Fernando Rubio Garcia <j332v755@wichita.edu>

Repository https://cran.r-universe.dev

Date/Publication 2026-10-10 00:10:09 UTC

RemoteUrl https://github.com/cran/bbqr

RemoteRef HEAD

RemoteSha 7d651ea7141c746c04298ab3291e0acb54c6709f

Contents

bbqr . . . 3
bbqr_alasso . . . 8
bbqr_lasso . . . 12
bbqr_none . . . 14
cbqr . . . 16
cbqr_alasso . . . 18
cbqr_lasso . . . 20
cbqr_none . . . 22
coef.bbqr . . . 24
plot.bbqr . . . 25
plot.cbqr . . . 25
predict.bbqr . . . 26
predict.cbqr . . . 28
prior . . . 28
summary.bbqr . . . 32
summary.cbqr . . . 33

Index **34**

bbqr *Bayesian binary quantile regression*

Description

Fits a quantile regression model to a binary response by Markov chain Monte Carlo, using the asymmetric Laplace distribution (ALD) and the location-scale mixture representation of Kozumi and Kobayashi (2011). Three penalty layers are available through the `penalty` argument, the identification anchor is exposed through `anchor`, and the prior hierarchy is chosen by `model` on the `prior()` object. The defaults – `model = "v5"` at `anchor = "sigma1"` – sample a posterior that is proper unconditionally and identified by construction.

Usage

```
bbqr(
  formula,
  data = NULL,
  quantile = 0.5,
  penalty = c("alasso", "lasso", "none"),
  ndraw = 10000,
  burn = NULL,
  keep = 1,
  prior = NULL,
  anchor = NULL,
  sigma_fixed = 1,
  standardize = FALSE,
  use_boundaries = FALSE,
  clip = NULL,
  sampler = NULL,
  beta_init = NULL,
  beta0_init = 0,
  check_separation = getOption("bbqr.check_separation", TRUE)
)
```

Arguments

`formula`	A model formula. The response must be binary: 0/1, logical, or a two-level factor (the second level is treated as the success).
`data`	A data frame in which to interpret `formula`.
`quantile`	The quantile level(s) to fit, each strictly between 0 and 1. If more than one is given, the samplers are run once per level and an object of class `"bbqr.list"` is returned.
`penalty`	The shrinkage layer: `"alasso"` (adaptive lasso, the default), `"lasso"`, or `"none"`.
`ndraw`	Total number of MCMC iterations.
`burn`	Number of initial iterations to discard. Also the window over which the Metropolis proposal for the penalty hyperparameter adapts. Defaults to `ndraw / 10`.

keep — Thinning interval; every keep-th draw is retained. ndraw must be divisible by keep.

prior — A prior specification from prior(). Defaults to prior() for this penalty, the fully proper model = "v5" hierarchy. Pass prior(penalty, model = "v3") to reproduce the published specification.

anchor — The identification anchor; see the Identification section of bbqr(). Defaults to "sigma1" for every penalty. Every anchor is available for every penalty, with one exception: penalty = "none" with anchor = "free" is not identified and is rejected.

sigma_fixed — The value the ALD inverse-scale is held at when anchor = "sigma1".

standardize — If TRUE, centre and scale the covariates before sampling and map the draws back to the original units afterwards. **Strongly recommended whenever the covariates are on different scales.** Note that this is not a mere reparameterization: the prior and the penalty apply to the standardized coefficients, so the posterior genuinely changes. That is the point — a single shared penalty is not comparable across covariates measured in different units, and leaving them unscaled can pull the estimated direction badly off. See the Scaling section of bbqr().

use_boundaries — If TRUE, clamp the latent quantities to a wide numerical range. Off by default; turn it on only if a chain is producing non-finite values.

clip — Which numerical clamps to apply, available for penalty = "alasso" only; see bbqr_alasso() for the eighteen sites and the accepted forms, including the "none", "v7clip" and "v4" presets. Refused under model = "v4" and "v5" unless every site is off.

sampler — Which routine draws the latents z and s, available for penalty = "alasso" only; "gig_half" (default) or "v4_invgaus". See bbqr_alasso(). Only "gig_half" is accepted under model = "v4" and "v5". Supplying it for the other penalties is an error rather than a silent no-op, because their kernels draw with one fixed routine.

beta_init, beta0_init — Starting values for the slopes and the intercept. beta_init may be named, in which case it is matched to the columns of the design matrix.

check_separation — If TRUE, test before sampling whether the classes are separated through the origin, record the answer on the fit, and warn where that leaves the coefficient posterior without a mean; see the Separation section of bbqr(). Defaults to the option bbqr.check_separation, or TRUE when that is unset. The draws do not depend on it.

Value

For a single quantile, an object of class "bbqr": a list with components

beta — Matrix of retained slope draws, one column per covariate.

beta0 — Vector of retained intercept draws.

sigma — Vector of retained ALD inverse-scale draws. Constant at sigma_fixed under anchor = "sigma1"; under the norm anchors it varies, because the rescaling step absorbs the scale into it.

`accept` Final Metropolis acceptance rate for the penalty hyperparameter, or `NA` when the sampler has no Metropolis step.

`penalty`, `anchor`, `quantile` The settings used.

`prior` The prior actually applied.

`model` Which hierarchy was sampled: `"v3"`, `"v4"` or `"v5"`.

`derived` `TRUE` when the chain targets a posterior one of the derivations writes down; `FALSE` when a projection anchor was combined with `"v4"` or `"v5"`.

`separated_origin` `TRUE` when the classes are separated through the origin in the covariates as sampled, `FALSE` when they are not, and `NA` when the test was switched off or could not decide at its tolerance. See the Separation section.

`b0_prior` The intercept prior actually used, as `mean`, `var` and `prec`. Recorded because the `"v5"` reference depends on `quantile`, so it cannot be reconstructed from the prior object alone.

Penalty-specific draws are also returned: `lambdasq`, `omega` and `delta` for the adaptive lasso; `lambdasq`, `tauhyper` and `delta` for the lasso. For several quantiles, an object of class `"bbqr.list"` holding one such object per level.

Derivation version

Each penalty has a published specification and a corrected one, selected by `prior(penalty, model = )`. `"v3"` is the published hierarchy: for the adaptive lasso and the lasso its joint posterior does not exist, because the log-uniform hyperprior on the global shrinkage scale (ω or τ_h) is not integrable along the ray where the scale, the λ_j^2 and the latent s_j vanish together while the likelihood stays bounded below. `"v5"`, the default, replaces every improper component with a proper one – $\mathrm{Gamma}(2, 2)$ on the global scale, and a τ-calibrated normal prior on the intercept – so that the posterior is proper with no condition on the design, on the number of covariates, or on the anchor. `"v4"` is the intermediate step for the adaptive lasso. The fitted object records which hierarchy was sampled in `model`, and whether the chain targets a derived posterior in `derived`. Under `"v4"` and `"v5"` the numerical clamps, argument floors and the historical sampler routine are refused, because the derivations say none of them is part of the target; `"v3"` keeps them reachable so that earlier results can be reproduced. See `prior()` for the hierarchies in full.

Choosing a penalty

The three penalties share the same latent-variable augmentation and differ only in what sits on top of the slopes:

`"none"` A vague $N(0, \sigma_\beta^2)$ prior on each slope and no shrinkage, following Benoit and Van den Poel (2012).

`"lasso"` One shrinkage parameter shared by all slopes, following Benoit, Al-Hamzawi and Yu (2013).

`"alasso"` A separate shrinkage parameter per slope, so that coefficients are penalised individually rather than uniformly, following Rubio Garcia (2023).

Identification

A binary threshold model identifies the coefficient vector only up to a positive scale: $\mathrm{sign}(x'\beta + \varepsilon)$ is unchanged if β and the error scale are multiplied by the same positive constant. Some anchor must therefore be imposed, and the choice is not innocuous. The available anchors are:

`"sigma1"` Fix the ALD inverse-scale at `sigma_fixed` (default 1) and never sample it. The default for every penalty. It closes the scale ray outright, so the posterior summaries of β are on a stated scale, and it is the convention under which the τ-calibrated intercept prior of `model = "v5"` is exact. In the simulation study behind this package its point estimates matched `"free"` while its credible intervals were markedly narrower, because a sampled σ lets the unidentified scale inflate them.

`"free"` Impose nothing beyond the prior: sample σ from its full conditional and let the prior alone settle the scale. This is the sampled-σ convention of the adaptive-lasso and lasso derivations, and it is a derived posterior under every `model`; it is simply not identified by the likelihood, so `||beta||` is a prior artefact. Not available for `penalty = "none"`: with neither a penalty nor an anchor the likelihood is flat along the scale ray, so nothing identifies the model and only the vague prior stops the chain drifting.

`"beta1"` Hold the first slope at 1 and never sample it. The first column of the design matrix must be a covariate with a genuinely non-zero coefficient for this to be sensible.

`"norm1"` Rescale so that $\|(\beta_0, \beta)\| = 1$ after every sweep.

`"normslopes"` Rescale so that $\|\beta\| = 1$ over the slopes alone, the convention used in the maximum-score literature.

The two norm anchors share one projection step and differ only in how the scaling constant is chosen. In both cases β_0 and the ALD inverse-scale σ are rescaled alongside the slopes. At the observed-data level the model is invariant under $(\beta_0, \beta, \sigma) \mapsto (c\beta_0, c\beta, \sigma/c)$. These projections are identification experiments and are not steps in any of the derivations.

Note, however, that this invariance does not extend to the *penalised posterior*: the shrinkage layer is stated on a fixed scale, so repeatedly renormalising the coefficients moves them relative to the penalty. Anchors that constrain the norm of the coefficient vector should therefore be expected to behave differently from `"sigma1"`, which leaves the coefficients where the sampler put them. Under `model = "v4"` or `"v5"` the three projection anchors (`"beta1"`, `"norm1"`, `"normslopes"`) warn and set `derived = FALSE` on the fit: the projection is an exact reparameterisation of the likelihood but not of the prior, so the chain targets no written-down posterior. They remain available because the comparison is the point of exposing the anchor at all.

Separation

The classes are *separated through the origin* when some direction w puts every observation strictly on its own side without the intercept: $x_i'w > 0$ for every success and $x_i'w < 0$ for every failure, in the covariates as the sampler sees them (after standardization when `standardize = TRUE`). Along such a direction the likelihood does not decay, so the posterior keeps the tail the slope prior has there.

For the default lasso and adaptive lasso (`model = "v5"`) with the inverse scale held fixed (`anchor = "sigma1"`) that tail is heavy: the shape δ of the shrinkage prior has prior mass near zero, and the slope prior has no first moment there. The posterior is then proper but the posterior mean of the coefficients, and with it the standard deviation, does not exist; a sample average of the draws

then estimates nothing and need not settle as the chain grows. Ghosh, Li and Mitra (2018) describe the same phenomenon for logistic regression under Cauchy priors. The unpenalised model, whose slope prior is normal, is not affected.

`bbqr()` and the three fitters test for this before sampling. The result is stored on the fit as `separated_origin`, and in the configuration just described a warning of class `"bbqr_separation_warning"` is raised, once per call, with a matching note in `print()` and `summary()`. The test is exact up to a numerical tolerance on the margin and uses no random numbers, so the draws are identical with or without it. It can be switched off with `check_separation = FALSE` or `options(bbqr.check_separation = FALSE)`, which is the thing to do in a simulation study that fits many separated samples on purpose. The absence of a warning is not a statement that the posterior mean exists: nothing is claimed for the other anchors or hierarchies.

What remains defined in that case: posterior medians and quantile intervals (`coef(fit, statistic = "median")`, `summary()`), the predictions built on the medians (`predict(fit, estimator = "plug_in_median")`), and the averaged probability (`predict(fit, estimator = "draw_average")`), which is bounded and so has a posterior expectation whenever the posterior is proper. Being defined is not the same as having settled: under separation the chain moves slowly through the tail, so check any of these across seeds or chain lengths before relying on it.

Scaling

The prior variances and the penalty are stated on the scale of the design matrix as supplied. If one covariate is measured in units a hundred times larger than another, a shared penalty shrinks them by very different relative amounts, and the recovered direction can be badly wrong. Setting `standardize = TRUE` centres and scales the covariates before sampling and maps the draws back afterwards; the back-transform preserves the linear predictor exactly, but the posterior itself differs because the penalty now acts on comparable coefficients. Unless the covariates are already on a common scale, prefer `standardize = TRUE`.

References

Benoit, D. F. and Van den Poel, D. (2012). Binary quantile regression: a Bayesian approach based on the asymmetric Laplace distribution. *Journal of Applied Econometrics*, 27(7), 1174–1188. doi:10.1002/jae.1216

Benoit, D. F., Al-Hamzawi, R. and Yu, K. (2013). Bayesian lasso binary quantile regression. *Computational Statistics*, 28(6), 2861–2873. doi:10.1007/s0018001304390

Ghosh, J., Li, Y. and Mitra, R. (2018). On the use of Cauchy prior distributions for Bayesian logistic regression. *Bayesian Analysis*, 13(2), 359–383. doi:10.1214/17BA1051

Kozumi, H. and Kobayashi, G. (2011). Gibbs sampling methods for Bayesian quantile regression. *Journal of Statistical Computation and Simulation*, 81(11), 1565–1578. doi:10.1080/00949655.2010.496117

Rubio Garcia, F. (2023). Bayesian adaptive lasso binary quantile regression with hybrid resampling for classification of imbalanced data. M.S. thesis, Wichita State University, Dept. of Mathematics, Statistics, and Physics. https://soar.wichita.edu/entities/publication/a2f86232-4704-4ec2-b685-751e7b04ec

See Also

`prior()` for hyperparameters, `summary.bbqr()` for posterior summaries, `plot.bbqr()` for trace and density plots, and `predict.bbqr()` for fitted probabilities. `cbqr()` fits the same three penalties when the response is observed rather than thresholded.

Examples

```
set.seed(42)
n <- 200
X <- matrix(rnorm(n * 4), n, 4, dimnames = list(NULL, paste0("x", 1:4)))
y <- as.numeric(X %*% c(1.5, -1, 0, 0) + rnorm(n) > 0)
dat <- data.frame(y = y, X)

fit <- bbqr(y ~ x1 + x2 + x3 + x4, data = dat, quantile = 0.5,
            ndraw = 1000, burn = 200)
fit
summary(fit)

# Compare the three penalties at the same quantile
fits <- lapply(c("none", "lasso", "alasso"), function(p)
  bbqr(y ~ x1 + x2 + x3 + x4, data = dat, penalty = p,
       ndraw = 1000, burn = 200))
sapply(fits, coef)

# The published adaptive-lasso hierarchy, whose posterior does not exist,
# is still available for reproducing earlier results
old <- bbqr(y ~ x1 + x2 + x3 + x4, data = dat, ndraw = 1000, burn = 200,
            prior = prior("alasso", model = "v3"))
c(fit$model, old$model)
```

`bbqr_alasso` *Adaptive-lasso binary quantile regression*

Description

Low-level sampler for binary quantile regression with an adaptive-lasso penalty, in which each slope receives its own shrinkage parameter. Most users should call `bbqr()` with `penalty = "alasso"` instead; this function is exported for direct access to the sampler. With the defaults – `model = "v5"` on the prior, `anchor = "sigma1"`, no clamps – every transition implements the corresponding full conditional in the V5 appendix: the ALD inverse-scale is held at `sigma_fixed`, the `omega` draw is `Gamma(alpha_omega + k delta, beta_omega + sum(1 / lambdasq))`, the intercept carries its τ-calibrated normal prior, and `delta` is updated by a random-walk Metropolis step on the log scale under a proper `Gamma(alpha_delta, beta_delta)` prior. That last prior is required rather than optional: see `prior()`. `anchor = "free"` samples the inverse-scale from its full conditional instead, which the appendix also covers.

Usage

```
bbqr_alasso(
  formula,
  data = NULL,
  quantile = 0.5,
  ndraw = 10000,
```

```
  burn = NULL,
  keep = 1,
  prior = NULL,
  anchor = c("sigma1", "free", "beta1", "norm1", "normslopes"),
  sigma_fixed = 1,
  standardize = FALSE,
  use_boundaries = FALSE,
  clip = NULL,
  sampler = c("gig_half", "v4_invgaus"),
  kappa_lam = 0,
  beta_init = NULL,
  beta0_init = 0,
  check_separation = getOption("bbqr.check_separation", TRUE)
)
```

Arguments

formula	A model formula. The response must be binary: 0/1, logical, or a two-level factor (the second level is treated as the success).
data	A data frame in which to interpret formula.
quantile	The quantile level(s) to fit, each strictly between 0 and 1. If more than one is given, the samplers are run once per level and an object of class "bbqr.list" is returned.
ndraw	Total number of MCMC iterations.
burn	Number of initial iterations to discard. Also the window over which the Metropolis proposal for the penalty hyperparameter adapts. Defaults to ndraw / 10.
keep	Thinning interval; every keep-th draw is retained. ndraw must be divisible by keep.
prior	A prior specification from prior(). Defaults to prior() for this penalty, the fully proper model = "v5" hierarchy. Pass prior(penalty, model = "v3") to reproduce the published specification.
anchor	The identification anchor; see the Identification section of bbqr(). Defaults to "sigma1" for every penalty. Every anchor is available for every penalty, with one exception: penalty = "none" with anchor = "free" is not identified and is rejected.
sigma_fixed	The value the ALD inverse-scale is held at when anchor = "sigma1".
standardize	If TRUE, centre and scale the covariates before sampling and map the draws back to the original units afterwards. **Strongly recommended whenever the covariates are on different scales.** Note that this is not a mere reparameterization: the prior and the penalty apply to the standardized coefficients, so the posterior genuinely changes. That is the point — a single shared penalty is not comparable across covariates measured in different units, and leaving them unscaled can pull the estimated direction badly off. See the Scaling section of bbqr().
use_boundaries	If TRUE, clamp the latent quantities to a wide numerical range. Off by default; turn it on only if a chain is producing non-finite values.

clip

Which numerical clamps to apply, for isolating which individual clamp a result depends on. There are eighteen sites: the eight the single use_boundaries flag used to gate together ("ystar", "z", "s", "sigma", "lambdasq", "omega", "delta", "mh_sd") and ten that restore guards the Experiment v4 build applied unconditionally and later builds removed ("v4_sd", "v4_z_psi", "v4_z_chi", "v4_zbound", "v4_s_lam", "v4_s_psi", "v4_s_chi", "v4_sbound", "v4_rate_sig", "v4_rate_om").

use_boundaries addresses the first eight sites and only those: the v4 guards were applied unconditionally rather than gated by it, so use_boundaries = TRUE does not switch them on. Reaching them requires an explicit clip.

Accepts three forms. A **named logical** vector overrides individual sites on top of use_boundaries, so clip = c(delta = FALSE) with use_boundaries = TRUE means every gated clamp except delta. A **character** vector names exactly the sites that are on and ignores use_boundaries entirely, which is what makes a named arm reproducible; the preset names "none" (nothing; "v6" is a synonym naming the build that first shipped it), "v7clip" (the eight gated clamps) and "v4" (the ten v4 guards, gated clamps off) are accepted here. An **unnamed logical** of length 1 or 18 is taken in site order; a length-1 value behaves exactly like use_boundaries, so it too reaches only the eight gated sites. NULL, the default, is use_boundaries throughout.

clip = "v4" reproduces v4's clamp configuration, not a v4 rerun: v4 also drew z and s with a hand-rolled inverse-Gaussian rather than the current GIG(1/2) routine, and ran an effectively flat delta prior. Those are a sampler and a prior, not clamps.

sampler

Which routine draws the latents z and s. Both full conditionals are GIG(1/2) and two routines implement that draw, which are not numerically equivalent:

"gig_half" The default. rgig_half, whose q is rationalized to avoid cancellation, with a limiting branch at chi == 0.

"v4_invgaus" The routine the v4 build shipped: draw the reciprocal of the latent as inverse Gaussian from the textbook q expression and invert. That expression subtracts two nearly equal quantities when chi is small, so it can return a latent at zero or Inf – the failure v4's guards were written to contain.

The switch exists so that the v4-to-v6 difference can be attributed. clip settles the guards and sampler settles the routine; holding one fixed while moving the other is what separates the two mechanisms. Both routines consume one normal and one uniform per draw, so two fits from the same seed that differ only in sampler stay aligned in the RNG stream and their paired difference is the routine alone. Reproducing v4's z and s draws takes sampler = "v4_invgaus" together with clip = "v4", since v4 applied those argument floors unconditionally; "v4_invgaus" with the floors off is accepted but is not a configuration v4 ever ran.

kappa_lam

Coefficient of an $\exp(-\kappa_\lambda/\lambda_j^2)$ tilt on each λ_j^2 conditional. The tilt is conjugate, so it only shifts the inverse-Gamma scale to $\sigma s_j/2 + \omega + \kappa_\lambda$; it was tried as a barrier keeping λ_j^2 away from zero. It is not part of any derived target and does not repair the ω defect – the normalising constant travels with the scale, which removes the very factor that made the origin integrable – so it must be 0 (the

default) under model = "v4" and "v5". Under model = "v3" a positive value is accepted, for reproducing the runs that used it.

beta_init, beta0_init
: Starting values for the slopes and the intercept. beta_init may be named, in which case it is matched to the columns of the design matrix.

check_separation
: If TRUE, test before sampling whether the classes are separated through the origin, record the answer on the fit, and warn where that leaves the coefficient posterior without a mean; see the Separation section of bbqr(). Defaults to the option bbqr.check_separation, or TRUE when that is unset. The draws do not depend on it.

Value

An object of class "bbqr", as documented in bbqr(), with two extra components: clip, the resolved logical(18) of switches, and clip_hits, the number of times each site's clamp actually bound. clip_hits is counted whether or not the switch was on, so a fit with every clamp off still reports how often each clamp would have fired. Denominators differ by site. ystar, z, v4_zarg and v4_zbound are out of ndraw * n; s, lambdasq, v4_sarg and v4_sbound out of ndraw * p; mh_sd out of burn; delta out of the accepted Metropolis proposals; omega and v4_rate_om out of ndraw. Multiply by the number of guards in a group where it holds more than one: v4_zarg has two, v4_sarg three, each rate group two. v4_sd is out of n alone, since v4 floored that standard deviation in the initialisation sweep only and left the main-loop assignment bare. sigma and v4_rate_sig stay at zero under anchor = "sigma1", where nothing ever reaches them.

Derivation version

Which of the three adaptive-lasso hierarchies is sampled is set by model on the prior() object, not by an argument here. "v4" and "v5" are derivation-faithful: their appendices state that no clamp, no sampler-argument floor, and no exp(-kappa_lam / lambda_j^2) tilt is part of the target, so a fit that names them while enabling one of those devices is refused rather than recorded. Concretely, under model = "v4" or "v5" this function requires clip = "none" (the default), kappa_lam = 0 and sampler = "gig_half". model = "v3" leaves all three switches reachable, because reproducing what the earlier builds actually ran needs them.

One further check is a warning rather than an error. Under model = "v4" with a free sigma, a flat intercept is admissible only if a > 1, and the historical default a = 1 sits exactly on the divergent boundary. That configuration is what Experiments v4 and v6 ran, so it stays available and warns instead of failing.

References

Rubio Garcia, F. (2023). Bayesian adaptive lasso binary quantile regression with hybrid resampling for classification of imbalanced data. M.S. thesis, Wichita State University, Dept. of Mathematics, Statistics, and Physics. https://soar.wichita.edu/entities/publication/a2f86232-4704-4ec2-b685-751e7b04ec

See Also

bbqr(), and cbqr_alasso() for the same penalty layer fitted to an observed continuous response

Examples

```
set.seed(1)
n <- 150
X <- matrix(rnorm(n * 3), n, 3, dimnames = list(NULL, c("x1", "x2", "x3")))
y <- as.numeric(X %*% c(1, -1, 0) + rnorm(n) > 0)
d <- data.frame(y = y, X)
fit <- bbqr_alasso(y ~ x1 + x2 + x3, data = d, ndraw = 400, burn = 100)
coef(fit)
```

`bbqr_lasso` *Lasso binary quantile regression*

Description

Low-level sampler for binary quantile regression with a standard lasso penalty: one shrinkage parameter shared by all slopes. This implements the hierarchical sampler of Benoit, Al-Hamzawi and Yu (2013). Most users should call `bbqr()` with `penalty = "lasso"` instead; this function is exported for direct access to the sampler.

Usage

```
bbqr_lasso(
  formula,
  data = NULL,
  quantile = 0.5,
  ndraw = 10000,
  burn = NULL,
  keep = 1,
  prior = NULL,
  anchor = c("sigma1", "free", "beta1", "norm1", "normslopes"),
  sigma_fixed = 1,
  standardize = FALSE,
  use_boundaries = FALSE,
  beta_init = NULL,
  beta0_init = 0,
  check_separation = getOption("bbqr.check_separation", TRUE)
)
```

Arguments

`formula`	A model formula. The response must be binary: 0/1, logical, or a two-level factor (the second level is treated as the success).
`data`	A data frame in which to interpret `formula`.
`quantile`	The quantile level(s) to fit, each strictly between 0 and 1. If more than one is given, the samplers are run once per level and an object of class `"bbqr.list"` is returned.

`ndraw` Total number of MCMC iterations.

`burn` Number of initial iterations to discard. Also the window over which the Metropolis proposal for the penalty hyperparameter adapts. Defaults to `ndraw / 10`.

`keep` Thinning interval; every `keep`-th draw is retained. `ndraw` must be divisible by `keep`.

`prior` A prior specification from `prior()`. Defaults to `prior()` for this penalty, the fully proper `model = "v5"` hierarchy. Pass `prior(penalty, model = "v3")` to reproduce the published specification.

`anchor` The identification anchor; see the Identification section of `bbqr()`. Defaults to `"sigma1"` for every penalty. Every anchor is available for every penalty, with one exception: `penalty = "none"` with `anchor = "free"` is not identified and is rejected.

`sigma_fixed` The value the ALD inverse-scale is held at when `anchor = "sigma1"`.

`standardize` If `TRUE`, centre and scale the covariates before sampling and map the draws back to the original units afterwards. **Strongly recommended whenever the covariates are on different scales.** Note that this is not a mere reparameterization: the prior and the penalty apply to the standardized coefficients, so the posterior genuinely changes. That is the point — a single shared penalty is not comparable across covariates measured in different units, and leaving them unscaled can pull the estimated direction badly off. See the Scaling section of `bbqr()`.

`use_boundaries` If `TRUE`, clamp the latent quantities to a wide numerical range. Off by default; turn it on only if a chain is producing non-finite values.

`beta_init`, `beta0_init` Starting values for the slopes and the intercept. `beta_init` may be named, in which case it is matched to the columns of the design matrix.

`check_separation` If `TRUE`, test before sampling whether the classes are separated through the origin, record the answer on the fit, and warn where that leaves the coefficient posterior without a mean; see the Separation section of `bbqr()`. Defaults to the option `bbqr.check_separation`, or `TRUE` when that is unset. The draws do not depend on it.

Details

The default `anchor = "sigma1"` holds the ALD inverse-scale at `sigma_fixed`; `anchor = "free"` samples it from its full conditional, which is what the published algorithm does. Which hierarchy is sampled – the published one, whose posterior does not exist, or the corrected `"v5"` default – is set by `model` on the `prior()` object.

Value

An object of class `"bbqr"`, as documented in `bbqr()`.

References

Benoit, D. F., Al-Hamzawi, R. and Yu, K. (2013). Bayesian lasso binary quantile regression. *Computational Statistics*, 28(6), 2861–2873. doi:10.1007/s0018001304390

See Also

bbqr(), and cbqr_lasso() for the same penalty layer fitted to an observed continuous response

Examples

```
set.seed(1)
n <- 150
X <- matrix(rnorm(n * 3), n, 3, dimnames = list(NULL, c("x1", "x2", "x3")))
y <- as.numeric(X %*% c(1, -1, 0) + rnorm(n) > 0)
d <- data.frame(y = y, X)
fit <- bbqr_lasso(y ~ x1 + x2 + x3, data = d, ndraw = 400, burn = 100)
coef(fit)
```

---

bbqr_none *Unpenalised binary quantile regression*

Description

Low-level sampler for binary quantile regression with a vague normal prior on the slopes and no shrinkage layer, implementing Benoit and Van den Poel (2012). Most users should call bbqr() with penalty = "none" instead.

Usage

```
bbqr_none(
  formula,
  data = NULL,
  quantile = 0.5,
  ndraw = 10000,
  burn = NULL,
  keep = 1,
  prior = NULL,
  anchor = c("sigma1", "beta1", "norm1", "normslopes"),
  standardize = FALSE,
  use_boundaries = FALSE,
  beta_init = NULL,
  beta0_init = 0,
  check_separation = getOption("bbqr.check_separation", TRUE)
)
```

Arguments

formula	A model formula. The response must be binary: 0/1, logical, or a two-level factor (the second level is treated as the success).
data	A data frame in which to interpret formula.

`quantile` The quantile level(s) to fit, each strictly between 0 and 1. If more than one is given, the samplers are run once per level and an object of class `"bbqr.list"` is returned.

`ndraw` Total number of MCMC iterations.

`burn` Number of initial iterations to discard. Defaults to `ndraw / 10`. This sampler has no penalty hyperparameter and no Metropolis step, so nothing adapts over it.

`keep` Thinning interval; every `keep`-th draw is retained. `ndraw` must be divisible by `keep`.

`prior` A prior specification from `prior()`. Defaults to `prior()` for this penalty, the fully proper `model = "v5"` hierarchy. Pass `prior(penalty, model = "v3")` to reproduce the published specification.

`anchor` The identification anchor; see the Identification section of `bbqr()`. Defaults to `"sigma1"` for every penalty. Every anchor is available for every penalty, with one exception: `penalty = "none"` with `anchor = "free"` is not identified and is rejected.

`standardize` If `TRUE`, centre and scale the covariates before sampling and map the draws back to the original units afterwards. **Strongly recommended whenever the covariates are on different scales.** Note that this is not a mere reparameterization: the prior and the penalty apply to the standardized coefficients, so the posterior genuinely changes. That is the point — a single shared penalty is not comparable across covariates measured in different units, and leaving them unscaled can pull the estimated direction badly off. See the Scaling section of `bbqr()`.

`use_boundaries` If `TRUE`, clamp the latent quantities to a wide numerical range. Off by default; turn it on only if a chain is producing non-finite values.

`beta_init`, `beta0_init`
Starting values for the slopes and the intercept. `beta_init` may be named, in which case it is matched to the columns of the design matrix.

`check_separation`
If `TRUE`, test before sampling whether the classes are separated through the origin, record the answer on the fit, and warn where that leaves the coefficient posterior without a mean; see the Separation section of `bbqr()`. Defaults to the option `bbqr.check_separation`, or `TRUE` when that is unset. The draws do not depend on it.

Details

The default `anchor = "sigma1"` reproduces the published algorithm, which holds the ALD scale at 1. `"free"` is deliberately absent from the choices here, unlike the penalised fitters: with neither a penalty nor an anchor the likelihood is flat along the scale ray, so the model is not identified and only the vague prior stops the chain drifting.

Value

An object of class `"bbqr"`, as documented in `bbqr()`.

References

Benoit, D. F. and Van den Poel, D. (2012). Binary quantile regression: a Bayesian approach based on the asymmetric Laplace distribution. *Journal of Applied Econometrics*, 27(7), 1174–1188. doi:10.1002/jae.1216

See Also

`bbqr()`, and `cbqr_none()` for the same penalty layer fitted to an observed continuous response

Examples

```
set.seed(1)
n <- 150
X <- matrix(rnorm(n * 3), n, 3, dimnames = list(NULL, c("x1", "x2", "x3")))
y <- as.numeric(X %*% c(1, -1, 0) + rnorm(n) > 0)
d <- data.frame(y = y, X)
fit <- bbqr_none(y ~ x1 + x2 + x3, data = d, ndraw = 400, burn = 100)
coef(fit)
```

---

cbqr *Bayesian quantile regression for a continuous response*

Description

User-facing entry point for the three continuous penalty layers. The binary threshold model is fitted by `bbqr()` instead; the two share no arguments that refer to identification, because a continuous response identifies the scale and a binary one does not.

Usage

```
cbqr(
  formula,
  data = NULL,
  quantile = 0.5,
  penalty = c("alasso", "lasso", "none"),
  ndraw = 10000,
  burn = NULL,
  keep = 1,
  prior = NULL,
  q = NULL,
  standardize = TRUE,
  use_boundaries = FALSE,
  clip = NULL,
  sampler = NULL,
  beta_init = NULL,
  beta0_init = 0
)
```

Arguments

formula	A two-sided formula. The intercept is split off and never penalised, so it must not appear among the penalised columns.
data	Optional data frame.
quantile	Quantile level(s) in (0, 1).
penalty	One of "alasso", "lasso", "none".
ndraw, burn, keep	MCMC controls. ndraw must be divisible by keep.
prior	Optional named list overriding the continuous prior defaults. Accepted names are a, b, b0_mean, b0_var, mh_sd0, target_acc, sigma_init, delta_init, alpha_delta, beta_delta, (note mh_sd0 defaults to 0.5 here, not the binary 0.05: delta mixes over a wider range once sigma is live) and, by penalty, omega_init/alpha_omega/beta_omega ("alasso"), tau_init/alpha_tau/beta_tau ("lasso") or beta_var ("none"). This is **not** a prior() object: that constructor builds the binary hierarchy and is rejected here.
q	Exponent on sigma in the local-scale prior. Layer-specific default: 1 for "alasso", 0 for "lasso". Supplying it with penalty = "none" is an error: there is no penalty layer for the scale to enter.
standardize	Centre and scale **both** X and y. Defaults to TRUE, which is not the binary default: no proper hierarchy is exactly response-scale equivariant, so fixing the scale is part of the specification rather than a preprocessing convenience. Draws are mapped back to the original units exactly.
use_boundaries	Apply the kernel's numerical clamps. "lasso" and "none" only; the adaptive-lasso kernel takes its clamps through the per-site clip argument instead and ignores this one.
clip	Per-site clamp configuration; "alasso" only.
sampler	Latent-draw routine; "alasso" only.
beta_init, beta0_init	Starting values.

Value

A cbqr object, or a cbqr.list when quantile has length > 1. The components are:

family, penalty, q, quantile what was fitted.

beta, beta0, sigma the kept draws: a matrix of slopes, and vectors for the intercept and the ALD inverse-scale. Unlike a binary fit, sigma is always a live parameter here.

lambdasq, omega, delta the shrinkage layer, "alasso" only: the per-coefficient scales, the global scale, and its shape.

lambdasq, tau_hyper, delta the shrinkage layer, "lasso" only: one global scale, its hyperparameter, and the shape. Absent for "none", which has no shrinkage layer.

accept, acc Metropolis acceptance rate for delta, and its running trace. NA when there is no Metropolis step.

ndraw, burn, keep, ndraw_kept the MCMC controls as used.

`n`, `nvar`, `names`, `terms`, `call` the data and the model frame the fit came from.

`prior` the resolved prior list, including any defaults filled in.

`standardized` the centring and scaling applied to `X` and `y`, or `NULL`. Draws are already mapped back, so this is a record rather than something to undo.

`b0_prior` the intercept prior actually used.

`clip`, `clip_hits`, `sampler` the clamp configuration, how often each site fired, and the latent-draw routine. `"alasso"` only.

References

Alhamzawi, R., Yu, K. and Benoit, D. F. (2012). Bayesian adaptive lasso quantile regression. *Statistical Modelling*, 12(3), 279–297. doi:10.1177/1471082X1101200304

Benoit, D. F., Al-Hamzawi, R. and Yu, K. (2013). Bayesian lasso binary quantile regression. *Computational Statistics*, 28(6), 2861–2873. doi:10.1007/s0018001304390

Kozumi, H. and Kobayashi, G. (2011). Gibbs sampling methods for Bayesian quantile regression. *Journal of Statistical Computation and Simulation*, 81(11), 1565–1578. doi:10.1080/00949655.2010.496117

See Also

`bbqr()` for the binary threshold model; `cbqr_alasso()`, `cbqr_lasso()` and `cbqr_none()` for the direct forms; `predict.cbqr()`, `summary.cbqr()` and `plot.cbqr()` for the fitted object. `prior()` does **not** configure these fitters.

Examples

```
set.seed(1)
d <- data.frame(x1 = rnorm(120), x2 = rnorm(120))
d$y <- 1 + 2 * d$x1 - d$x2 + rnorm(120)
fit <- cbqr(y ~ x1 + x2, d, ndraw = 400, burn = 100)
round(coef(fit), 2)
```

---

cbqr_alasso *Continuous quantile regression with the adaptive lasso*

Description

Fits the continuous-response adaptive-lasso hierarchy for an observed response. The response is observed, so there is no latent *response* draw and no identification anchor: sigma is identified by the data and is sampled every sweep.

Usage

```
cbqr_alasso(
  formula,
  data = NULL,
  quantile = 0.5,
  ndraw = 10000,
  burn = NULL,
  keep = 1,
  prior = NULL,
  q = NULL,
  standardize = TRUE,
  clip = NULL,
  sampler = NULL,
  beta_init = NULL,
  beta0_init = 0
)
```

Arguments

formula	A two-sided formula. The intercept is split off and never penalised, so it must not appear among the penalised columns.
data	Optional data frame.
quantile	Quantile level(s) in (0, 1).
ndraw, burn, keep	MCMC controls. ndraw must be divisible by keep.
prior	Optional named list overriding the continuous prior defaults. Accepted names are a, b, b0_mean, b0_var, mh_sd0, target_acc, sigma_init, delta_init, alpha_delta, beta_delta, (note mh_sd0 defaults to 0.5 here, not the binary 0.05: delta mixes over a wider range once sigma is live) plus omega_init, alpha_omega and beta_omega. This is **not** a prior() object: that constructor builds the binary hierarchy and is rejected here.
q	Exponent on sigma in the local-scale prior rate sigma^q / (2 lambda_j^2). Defaults to 1, reproducing Alhamzawi, Yu & Benoit (2012). q = 2 is response-scale equivariant in the penalty block – no proper hierarchy can be made exactly invariant to rescaling y – and draws sigma by slice sampling rather than conjugately. See vignette("bbqr").
standardize	Centre and scale **both** X and y. Defaults to TRUE, which is not the binary default: no proper hierarchy is exactly response-scale equivariant, so fixing the scale is part of the specification rather than a preprocessing convenience. Draws are mapped back to the original units exactly.
clip	Optional clamp configuration: a single logical, a named logical vector over the sixteen continuous clamp sites, or a character vector of site names. **Not** the binary eighteen-site contract.
sampler	"gig_half" (default) or "v4_invgaus".
beta_init, beta0_init	Starting values.

Value

An object of class `cbqr`.

References

Alhamzawi, R., Yu, K. and Benoit, D. F. (2012). Bayesian adaptive lasso quantile regression. *Statistical Modelling*, 12(3), 279–297. doi:10.1177/1471082X1101200304

Benoit, D. F., Al-Hamzawi, R. and Yu, K. (2013). Bayesian lasso binary quantile regression. *Computational Statistics*, 28(6), 2861–2873. doi:10.1007/s0018001304390

Kozumi, H. and Kobayashi, G. (2011). Gibbs sampling methods for Bayesian quantile regression. *Journal of Statistical Computation and Simulation*, 81(11), 1565–1578. doi:10.1080/00949655.2010.496117

See Also

`cbqr()` for the common interface, `bbqr_alasso()` for the binary counterpart. `prior()` configures that one, not this one.

Examples

```
set.seed(1)
d <- data.frame(x1 = rnorm(120), x2 = rnorm(120))
d$y <- 1 + 2 * d$x1 - d$x2 + rnorm(120)
fit <- cbqr_alasso(y ~ x1 + x2, d, ndraw = 400, burn = 100)
round(coef(fit), 2)
```

---

cbqr_lasso *Continuous quantile regression with the lasso*

Description

Fits the continuous-response lasso hierarchy for an observed response: a single shared `lambda^2` carried as a rate with a Gamma prior and a `tau_h` hyperlayer. The prior rate here is `sigma^q * lambda^2 / 2`, with `lambda^2` a single global scale multiplying; the adaptive-lasso rate is `sigma^q / (2 * lambda_j^2)`, one scale per coefficient and dividing. The two forms are different by construction, not a typo.

Usage

```
cbqr_lasso(
  formula,
  data = NULL,
  quantile = 0.5,
  ndraw = 10000,
  burn = NULL,
  keep = 1,
  prior = NULL,
  q = NULL,
```

```
  standardize = TRUE,
  use_boundaries = FALSE,
  beta_init = NULL,
  beta0_init = 0
)
```

Arguments

formula	A two-sided formula. The intercept is split off and never penalised, so it must not appear among the penalised columns.
data	Optional data frame.
quantile	Quantile level(s) in (0, 1).
ndraw, burn, keep	MCMC controls. ndraw must be divisible by keep.
prior	Optional named list overriding the continuous prior defaults. Accepted names are a, b, b0_mean, b0_var, mh_sd0, target_acc, sigma_init, delta_init, alpha_delta, beta_delta, (note mh_sd0 defaults to 0.5 here, not the binary 0.05: delta mixes over a wider range once sigma is live) plus tau_init, alpha_tau and beta_tau. This is **not** a prior() object: that constructor builds the binary hierarchy and is rejected here.
q	Exponent on sigma in the local-scale prior rate sigma^q lambda^2 / 2. Defaults to **0**, reproducing the published sigma-free penalty lambda|beta_j|. This differs from the adaptive layer's default of 1 because the two published specifications differ.
standardize	Centre and scale **both** X and y. Defaults to TRUE, which is not the binary default: no proper hierarchy is exactly response-scale equivariant, so fixing the scale is part of the specification rather than a preprocessing convenience. Draws are mapped back to the original units exactly.
use_boundaries	Apply the kernel's numerical clamps.
beta_init, beta0_init	Starting values.

Details

This is Benoit, Al-Hamzawi & Yu (2013)'s hierarchy, **not** the adaptive one with k = 1, and the two are not reachable from one another.

Value

An object of class cbqr.

References

Alhamzawi, R., Yu, K. and Benoit, D. F. (2012). Bayesian adaptive lasso quantile regression. *Statistical Modelling*, 12(3), 279–297. doi:10.1177/1471082X1101200304

Benoit, D. F., Al-Hamzawi, R. and Yu, K. (2013). Bayesian lasso binary quantile regression. *Computational Statistics*, 28(6), 2861–2873. doi:10.1007/s0018001304390

Kozumi, H. and Kobayashi, G. (2011). Gibbs sampling methods for Bayesian quantile regression. *Journal of Statistical Computation and Simulation*, 81(11), 1565–1578. doi:10.1080/00949655.2010.496117

See Also

`cbqr()` for the common interface, `bbqr_lasso()` for the binary counterpart.

Examples

```
set.seed(1)
d <- data.frame(x1 = rnorm(120), x2 = rnorm(120))
d$y <- 1 + 2 * d$x1 - d$x2 + rnorm(120)
fit <- cbqr_lasso(y ~ x1 + x2, d, ndraw = 400, burn = 100)
round(coef(fit), 2)
```

---

cbqr_none *Continuous quantile regression, unpenalised*

Description

Fits the unpenalised hierarchy for an observed response. Unlike its binary counterpart, which holds sigma at the identification anchor and never samples it, this kernel draws sigma every sweep from its Gamma full conditional given the latents, with shape `a + 3n/2`.

Usage

```
cbqr_none(
  formula,
  data = NULL,
  quantile = 0.5,
  ndraw = 10000,
  burn = NULL,
  keep = 1,
  prior = NULL,
  standardize = TRUE,
  use_boundaries = FALSE,
  beta_init = NULL,
  beta0_init = 0
)
```

Arguments

`formula`	A two-sided formula. The intercept is split off and never penalised, so it must not appear among the penalised columns.
`data`	Optional data frame.
`quantile`	Quantile level(s) in (0, 1).

`ndraw`, `burn`, `keep`
: MCMC controls. `ndraw` must be divisible by `keep`.

`prior`
: Optional named list overriding the continuous prior defaults. Accepted names are `a`, `b`, `b0_mean`, `b0_var`, `mh_sd0`, `target_acc`, `sigma_init`, `delta_init`, `alpha_delta`, `beta_delta`, (note `mh_sd0` defaults to 0.5 here, not the binary 0.05: `delta` mixes over a wider range once `sigma` is live) plus `beta_var`. This is **not** a `prior()` object: that constructor builds the binary hierarchy and is rejected here.

`standardize`
: Centre and scale **both** `X` and `y`. Defaults to `TRUE`, which is not the binary default: no proper hierarchy is exactly response-scale equivariant, so fixing the scale is part of the specification rather than a preprocessing convenience. Draws are mapped back to the original units exactly.

`use_boundaries`
: Apply the kernel's numerical clamps.

`beta_init`, `beta0_init`
: Starting values.

Value

An object of class `cbqr`.

References

Alhamzawi, R., Yu, K. and Benoit, D. F. (2012). Bayesian adaptive lasso quantile regression. *Statistical Modelling*, 12(3), 279–297. doi:10.1177/1471082X1101200304

Benoit, D. F., Al-Hamzawi, R. and Yu, K. (2013). Bayesian lasso binary quantile regression. *Computational Statistics*, 28(6), 2861–2873. doi:10.1007/s0018001304390

Kozumi, H. and Kobayashi, G. (2011). Gibbs sampling methods for Bayesian quantile regression. *Journal of Statistical Computation and Simulation*, 81(11), 1565–1578. doi:10.1080/00949655.2010.496117

See Also

`cbqr()` for the common interface, `bbqr_none()` for the binary counterpart.

Examples

```
set.seed(1)
d <- data.frame(x1 = rnorm(120), x2 = rnorm(120))
d$y <- 1 + 2 * d$x1 - d$x2 + rnorm(120)
fit <- cbqr_none(y ~ x1 + x2, d, ndraw = 400, burn = 100)
round(coef(fit), 2)
```

`coef.bbqr` *Extract posterior coefficient summaries*

Description

The posterior mean of every coefficient by default, or the posterior median on request.

Usage

```
## S3 method for class 'bbqr'
coef(object, statistic = c("mean", "median"), ...)

## S3 method for class 'bbqr.list'
coef(object, statistic = c("mean", "median"), ...)
```

Arguments

`object`	An object of class `"bbqr"`, `"cbqr"`, `"bbqr.list"` or `"cbqr.list"`. Continuous fits inherit from `"bbqr"`, so this method serves them too.
`statistic`	Which summary of the retained draws to return: `"mean"` (the default) or `"median"`. Each is taken coefficient by coefficient, the intercept included.
`...`	Ignored.

Details

For a binary fit whose classes are separated through the origin, the default lasso and adaptive lasso have a proper posterior whose coefficient mean does not exist; see the Separation section of `bbqr()`. The median is defined in that case too. It is not thereby guaranteed to have settled at the chain length in use, so compare it across seeds or chain lengths before relying on it.

Value

A named numeric vector, intercept first. For a `"bbqr.list"`, a matrix with one column per quantile.

Examples

```
set.seed(1)
n <- 150
X <- matrix(rnorm(n * 3), n, 3, dimnames = list(NULL, c("x1", "x2", "x3")))
y <- as.numeric(X %*% c(1, -1, 0) + rnorm(n) > 0)
fit <- bbqr(y ~ x1 + x2 + x3, data = data.frame(y = y, X),
            ndraw = 500, burn = 100)
coef(fit)
coef(fit, statistic = "median")
```

plot.bbqr *Trace and density plots for a fitted binary quantile regression*

Description

Trace and density plots for a fitted binary quantile regression

Usage

```
## S3 method for class 'bbqr'
plot(x, which = NULL, type = c("both", "trace", "density"), ...)
```

Arguments

x	An object of class "bbqr".
which	Which coefficients to plot, by name or index. Index 1 is the intercept. Defaults to all.
type	"trace", "density", or "both" (the default), which draws the two side by side.
...	Passed to the underlying plotting calls.

Value

x, invisibly. Called for its side effect.

Examples

```
set.seed(1)
n <- 150
X <- matrix(rnorm(n * 3), n, 3, dimnames = list(NULL, c("x1", "x2", "x3")))
y <- as.numeric(X %*% c(1, -1, 0) + rnorm(n) > 0)
fit <- bbqr(y ~ x1 + x2 + x3, data = data.frame(y = y, X),
            ndraw = 500, burn = 100)
op <- graphics::par(no.readonly = TRUE)
plot(fit, which = "x1")
graphics::par(op)
```

plot.cbqr *Trace and density plots for a continuous quantile regression*

Description

Same interface as plot.bbqr(), with one addition: sigma is a plottable parameter here. In the binary model it is constant under the default anchor = "sigma1", so there is usually nothing to look at; here it is always a live parameter.

Usage

```
## S3 method for class 'cbqr'
plot(x, which = NULL, type = c("both", "trace", "density"), ...)
```

Arguments

x	An object of class "cbqr".
which	Coefficient names or indices; defaults to all, plus sigma.
type	"both", "trace" or "density".
...	Passed to the underlying plot calls.

Value

x, invisibly. Called for the side effect of drawing.

See Also

plot.bbqr() for the binary counterpart.

---

predict.bbqr *Predicted probabilities and indices from a binary quantile regression*

Description

Predicted probabilities and indices from a binary quantile regression

Usage

```
## S3 method for class 'bbqr'
predict(
  object,
  newdata = NULL,
  type = c("response", "link", "class"),
  estimator = c("plug_in_mean", "plug_in_median", "draw_average"),
  ...
)
```

Arguments

object	An object of class "bbqr".
newdata	A data frame of covariates. Required: fitted objects do not retain the design matrix, so that they stay small enough to hold thousands of fits in memory during a simulation study.
type	"response" returns $P(y = 1 \mid x)$ implied by the ALD at the fitted quantile; "link" returns the latent index $\beta_0 + x'\beta$; "class" returns the 0/1 label given by the sign of that index.

`estimator` How the retained draws are turned into a prediction. `"plug_in_mean"` (the default) evaluates the prediction at the posterior mean coefficients and the mean inverse scale. `"plug_in_median"` does the same at their posterior medians. `"draw_average"` evaluates the probability for every retained draw, with that draw's own intercept, slopes and inverse scale, and averages the probabilities; it is available for `type = "response"` only.

`...` Ignored.

Details

Predictions use the posterior mean coefficients by default. Because the coefficient vector is identified only up to a positive scale, `type = "response"` depends on the identification anchor, whereas `type = "class"` does not.

The probability is $1 - F(-\eta)$ for an asymmetric Laplace error with skewness τ and inverse scale σ, that is $1 - \tau \exp\{-(1-\tau)\sigma\eta\}$ for $\eta > 0$ and $(1-\tau)\exp(\tau\sigma\eta)$ otherwise. It equals $1 - \tau$ at $\eta = 0$, which is where `type = "class"` switches label.

Where the classes are separated through the origin the default lasso and adaptive lasso have no coefficient posterior mean (see the Separation section of `bbqr()`), and the other two estimators do not rely on one. The posterior medians are defined. The averaged probability is bounded, so its posterior expectation exists whenever the posterior does. Neither is thereby guaranteed to have settled at the chain length in use. Three points about `"draw_average"`:

- It is an average of probabilities, so it has no linear predictor and no label of its own; `type = "link"` and `type = "class"` are refused. Anyone who thresholds it should do so at $1 - \tau$, not at 0.5, and should not expect the result to equal `type = "class"`.
- For a fit with `derived = FALSE` it is an average over the draws of that chain and not the predictive probability of a stated model.
- It costs `nrow(newdata)` times the number of retained draws in arithmetic, done in blocks of rows.

There is no `predict()` method for a `"bbqr.list"`; predict from its elements one quantile level at a time.

Value

A numeric vector, one entry per row of `newdata`.

Examples

```
set.seed(1)
n <- 150
X <- matrix(rnorm(n * 3), n, 3, dimnames = list(NULL, c("x1", "x2", "x3")))
y <- as.numeric(X %*% c(1, -1, 0) + rnorm(n) > 0)
d <- data.frame(y = y, X)
fit <- bbqr(y ~ x1 + x2 + x3, data = d, ndraw = 500, burn = 100)
head(predict(fit, newdata = d, type = "response"))
mean(predict(fit, newdata = d, type = "class") == y)
head(predict(fit, newdata = d, estimator = "draw_average"))
```

predict.cbqr *Predictions from a continuous quantile regression*

Description

Returns the fitted conditional quantile of the response, which is the linear predictor. That is a different quantity from what `predict.bbqr()` returns – `P(y = 1 | x)` from the ALD distribution function – which on a continuous fit saturates at 1 and is silently meaningless.

Usage

```
## S3 method for class 'cbqr'
predict(object, newdata = NULL, interval = FALSE, level = 0.95, ...)
```

Arguments

`object`	An object of class `"cbqr"`.
`newdata`	A data frame of predictors. Required: the fitted object does not store the design matrix.
`interval`	Return posterior credible intervals for the fitted quantile as well as the point estimate. Uses the full retained draws, so the cost is `nrow(newdata) * ndraw_kept` doubles.
`level`	Credible level when `interval = TRUE`.
`...`	Ignored.

Value

A numeric vector, or a matrix with columns `fit`, `lower`, `upper` when `interval = TRUE`.

prior *Prior and hyperparameter settings for the binary fitters*

Description

Builds the list of prior hyperparameters and Metropolis tuning constants for one of the three penalty layers.

Usage

```
prior(
  penalty = c("alasso", "lasso", "none"),
  model = c("v5", "v4", "v3"),
  a = NULL,
  b = NULL,
  alpha_delta = NULL,
  beta_delta = NULL,
  alpha_omega = NULL,
  beta_omega = NULL,
  alpha_tau = NULL,
  beta_tau = NULL,
  b0_mean = NULL,
  b0_var = NULL,
  delta_init = 1,
  omega_init = 1,
  tau_init = 1,
  mh_sd0 = 0.05,
  target_acc = 0.3,
  beta_var = 100
)
```

Arguments

`penalty` Which sampler the prior is for: `"alasso"` (adaptive lasso), `"lasso"`, or `"none"`.

`model` Which prior hierarchy to sample. The three are nested corrections, each removing one failure of propriety. For the adaptive lasso they differ only in the prior on ω and the prior on the intercept; for the lasso, in the priors on τ_h, δ and the intercept; for the unpenalised model, in the prior on the intercept alone. `"v3"` and `"v5"` are defined for every penalty; `"v4"` exists only for `"alasso"`, because ω does.

- `"v5"` The default. Every prior component is proper: for `"alasso"`, `"v4"` plus $\beta_0 \sim N(m_{00}, V_{00})$; for `"lasso"`, $\mathrm{Gamma}(2, 2)$ priors on τ_h and δ plus the same intercept prior; for `"none"`, the intercept prior alone. The observed likelihood is bounded by one, so the posterior is proper unconditionally: no condition on the design, on k, on the anchor, or on a beyond positivity.
- `"v4"` $\omega \sim \mathrm{Gamma}(2, 2)$ with a flat intercept; `"alasso"` only. Removes the ω defect. Propriety is then conditional: with a free σ it requires $a > 1$ plus an overlap condition on the design, which is why the default here is `a = b = 2` rather than the historical `a = b = 1`. At `a = 1` the fit warns.
- `"v3"` The published specifications, retained as a baseline. For `"alasso"`: $p(\omega) \propto 1/\omega$ with a flat intercept, the hierarchy Experiments v4 and v6 ran. **Its joint posterior does not exist.** The prior hierarchy is a scale family in ω: on the wedge where ω, λ_j^2 and s_j vanish together with $\beta_j = O(\sqrt{\omega})$ the prior measure is $d\omega/\omega$ while the likelihood is bounded below, so the normalizing integral diverges. That holds for every prior on δ and at either anchor. For `"lasso"`: the hierarchy of Benoit, Al-Hamzawi and Yu (2013),

$p(\tau_h) \propto 1/\tau_h$ and $p(\delta) \propto 1$, which carries the same defect along $\tau_h \to 0$. For `"none"`: the flat intercept of Benoit and Van den Poel (2012), whose posterior is proper if and only if both outcome classes occur.

`"v4"` and `"v5"` are defined by their derivations, which state that no clamp, sampler-argument floor, or $\exp(-\kappa_\lambda/\lambda_j^2)$ tilt is part of the target. A fit that names them and enables such a device is refused rather than silently recorded; see `bbqr_alasso`. `"v3"` leaves every switch reachable, because reproducing the historical build needs them.

`a`, `b`

Shape and rate of the $\mathrm{Gamma}(a, b)$ prior on the ALD inverse-scale σ. Ignored when the inverse-scale is anchored (`anchor = "sigma1"`, the default) and, **in the binary model**, for `penalty = "none"`, where $\sigma \equiv 1$ is the identification convention. The continuous unpenalised kernel samples σ and does use `a` and `b`. Defaults are `a = b = 2` under `"v4"` and `"v5"`, the mean-one reference that also satisfies the $a > 1$ condition a flat intercept needs against a free σ; under `"v3"` they follow the source papers, `a = b = 1` for the adaptive lasso and `a = b = 0.1` for the lasso.

`alpha_delta`, `beta_delta`

Shape and rate of the Gamma prior on the penalty hyperparameter δ.

For `penalty = "lasso"` the default is `(2, 2)` under `"v5"`, which requires `beta_delta > 0`: the flat prior is not integrable at the large-δ end, so it has no place in a hierarchy whose every component is proper. Under `"v3"` the default is `(1, 0)`, the improper flat prior $p(\delta) \propto 1$ of Benoit, Al-Hamzawi and Yu (2013): at `alpha_delta = 1` and `beta_delta = 0` the prior contribution $(\alpha - 1)\log\delta - \beta\delta$ vanishes and the Metropolis step targets that paper's Eq. (13) exactly. There is a single shrinkage parameter in that model.

For `penalty = "alasso"` the default is `(2, 2)` under every `model`, and `beta_delta` must be strictly positive. With k adaptive penalties the marginal posterior of δ tends to a positive constant, so a flat prior leaves it non-integrable and the joint posterior does not exist; any positive rate restores propriety, with no condition on k. δ is the shape of the inverse-Gamma prior on λ_j^2 and therefore controls how dispersed the per-coefficient shrinkage is: both $\delta \to 0$ (where the inverse-Gamma family degenerates) and $\delta \to \infty$ (where the λ_j^2 coincide and the penalty stops being adaptive) are degenerate, so the prior is chosen to vanish at both ends, which requires `alpha_delta > 1`. At `(2, 2)` the prior has mode $1/2$ and mean 1.

`alpha_omega`, `beta_omega`

Shape and rate of the Gamma prior on the global shrinkage scale ω, whose full conditional is $\mathrm{Gamma}(\alpha_\omega + k\delta, \beta_\omega + \sum_j \lambda_j^{-2})$. Both zero is $p(\omega) \propto 1/\omega$ exactly, which is `model = "v3"`; `"v4"` and `"v5"` require both strictly positive. Defaults are `(0, 0)` for `"v3"` and `(2, 2)` otherwise.

`alpha_tau`, `beta_tau`

Shape and rate of the Gamma prior on the lasso's global shrinkage hyperparameter τ_h; `penalty = "lasso"` only, where τ_h plays the role ω plays in the adaptive lasso. Its full conditional is $\mathrm{Gamma}(\alpha_\tau + \delta, \beta_\tau + \lambda^2)$. Both zero is $p(\tau_h) \propto 1/\tau_h$, the hyperprior of Benoit, Al-Hamzawi and Yu (2013) and the `"v3"` default; it leaves the joint posterior improper along $\tau_h \to 0$ by the same mechanism as ω. `"v5"` requires both strictly positive and defaults to `(2, 2)`.

`b0_mean`, `b0_var` Mean and variance of the $N(m_{00}, V_{00})$ prior on the intercept. `b0_var = Inf` is the flat intercept of `"v3"` and `"v4"`, carried into the kernel as a prior precision of exactly zero rather than as a large finite variance. `NA`, the `"v5"` default, means the τ-calibrated reference, which cannot be resolved here because τ is not known until the fit.

That reference is the moment match to the exactly calibrated prior. At the $\sigma = 1$ anchor with $x'\beta = 0$ the model reduces to $\Pr(y = 1) = G_\tau(\beta_0)$ with G_τ the distribution function of $-E$ for $E \sim \mathrm{ALD}(0, 1, \tau)$, so by the probability integral transform $G_\tau(\beta_0)$ is exactly uniform when $\beta_0 \sim \mathrm{ALD}(0, 1, 1-\tau)$. That law is not conjugate; its first two moments are $m_{00} = -\theta = -(1-2\tau)/(\tau(1-\tau))$ and $V_{00} = (1 - 2\tau + 2\tau^2)/(\tau^2(1-\tau)^2)$, giving $(0, 8)$ at the median.

Note that m_{00} is not zero away from the median. Centring the intercept prior at the origin puts the prior-predictive mean of $\Pr(y = 1)$ at 0.73 rather than 0.5 at $\tau = 0.1$, so it is a specification error and not a neutral default.

`delta_init`, `omega_init`, `tau_init`
Starting values for the penalty hyperparameters. `omega_init` applies to the adaptive lasso only and `tau_init` to the lasso only.

`mh_sd0` Initial standard deviation of the random-walk Metropolis proposal for δ, on the log scale.

`target_acc` Target acceptance rate for that Metropolis step. The proposal standard deviation is adapted towards this value during burn-in and then held fixed.

`beta_var` Prior variance of each slope under `penalty = "none"`. Benoit and Van den Poel (2012) use 100, a vague prior.

Details

This object is accepted by `bbqr()` and the three direct binary fitters, and by those only. The continuous fitters (`cbqr()` and friends) take a **plain named list**, not a `bbqr.prior`: their hierarchy has no anchor, a live σ, and the `q` exponent, and its reference defaults differ. Passing the result of this function to `cbqr()` fails with "Unknown prior component(s)"; `cbqr()` documents the components they do accept. The defaults are the `"v5"` hierarchy, in which every prior component is a probability distribution and the posterior is proper unconditionally; `model = "v3"` reproduces the specification of the source paper for each penalty instead. In most cases this function only needs to be called to change one or two values.

Value

A list of class `"bbqr.prior"`.

References

Benoit, D. F. and Van den Poel, D. (2012). Binary quantile regression: a Bayesian approach based on the asymmetric Laplace distribution. *Journal of Applied Econometrics*, 27(7), 1174–1188. doi:10.1002/jae.1216

Benoit, D. F., Al-Hamzawi, R. and Yu, K. (2013). Bayesian lasso binary quantile regression. *Computational Statistics*, 28(6), 2861–2873. doi:10.1007/s0018001304390

See Also

bbqr() and the direct binary fitters, which accept this object; cbqr(), which does not.

Examples

```
# Default adaptive-lasso prior: the fully proper "v5" hierarchy
prior("alasso")

# The published specification, for reproducing earlier results. Its
# posterior does not exist, and the print method says so.
prior("alasso", model = "v3")

# A vaguer prior on the slopes for the unpenalised sampler
prior("none", beta_var = 1000)
```

---

summary.bbqr *Summarise a fitted binary quantile regression*

Description

Posterior means, standard deviations and equal-tailed credible intervals for every coefficient. A coefficient whose interval excludes zero is flagged as selected, which is the variable-selection rule used with these samplers.

Usage

```
## S3 method for class 'bbqr'
summary(object, level = 0.95, ...)
```

Arguments

object	An object of class "bbqr".
level	Credible level for the intervals.
...	Ignored.

Value

An object of class "bbqr.summary". Its coefficients component is a data frame with one row per coefficient and the columns mean, sd, lower, upper, selected and median. The posterior median is returned but not printed. Where the classes are separated through the origin (see the Separation section of bbqr()) the mean and the standard deviation may not exist, while the median and the interval are defined.

Examples

```
set.seed(1)
n <- 150
X <- matrix(rnorm(n * 3), n, 3, dimnames = list(NULL, c("x1", "x2", "x3")))
y <- as.numeric(X %*% c(1, -1, 0) + rnorm(n) > 0)
fit <- bbqr(y ~ x1 + x2 + x3, data = data.frame(y = y, X),
            ndraw = 500, burn = 100)
summary(fit)
```

`summary.cbqr` *Summarise a fitted continuous quantile regression*

Description

Posterior means, standard deviations and equal-tailed credible intervals for every coefficient, plus the ALD inverse-scale. A coefficient whose interval excludes zero is flagged.

Usage

```
## S3 method for class 'cbqr'
summary(object, level = 0.95, ...)
```

Arguments

`object`	An object of class `"cbqr"`.
`level`	Credible level for the intervals.
`...`	Ignored.

Details

Unlike the binary summary, there is no identification caveat: a continuous response identifies the scale, so the coefficients are on the scale of `y` and are directly interpretable.

Value

An object of class `"cbqr.summary"`.

Index

bbqr, 3
bbqr(), *4*, *8, 9*, *11–16*, *18*, *24*, *27*, *31, 32*
bbqr_alasso, 8, *30*
bbqr_alasso(), *4*, *20*
bbqr_lasso, 12
bbqr_lasso(), *22*
bbqr_none, 14
bbqr_none(), *23*

cbqr, 16
cbqr(), *7*, *20*, *22, 23*, *31, 32*
cbqr_alasso, 18
cbqr_alasso(), *11*, *18*
cbqr_lasso, 20
cbqr_lasso(), *14*, *18*
cbqr_none, 22
cbqr_none(), *16*, *18*
coef.bbqr, 24

plot.bbqr, 25
plot.bbqr(), *7*, *25, 26*
plot.cbqr, 25
plot.cbqr(), *18*
predict.bbqr, 26
predict.bbqr(), *7*, *28*
predict.cbqr, 28
predict.cbqr(), *18*
prior, 28
prior(), *3–5*, *7–9*, *11*, *13*, *15*, *17–21*, *23*

summary.bbqr, 32
summary.bbqr(), *7*
summary.cbqr, 33
summary.cbqr(), *18*