

Package: mixqr (via r-universe)

June 26, 2026

Type Package

Title Extensible Finite Mixtures of Quantile and Expectile Regressions

Version 0.2.0

Description An extensible expectation-maximization (EM) framework for finite mixtures of quantile regressions (clusterwise / mixture-of-experts quantile regression). A single EM substrate with an engine/extension contract carries a family of capabilities: the core free-weight mixture of Wu and Yao (2016) <[doi:10.1016/j.csda.2014.04.014](https://doi.org/10.1016/j.csda.2014.04.014)> -- a fast asymmetric-Laplace path and the nonparametric kernel-density EM with components constrained to have their tau-quantile equal to zero (Hall and Presnell 1999 device); expectile and M-quantile component-loss families (Newey and Powell 1987; Breckling and Chambers 1988); component-specific penalized variable selection (SCAD / adaptive-LASSO, the quantile analogue of Khalili and Chen 2007); and joint multi-quantile estimation with a shared latent classification and non-crossing component curves. Provides classification-aware standard errors (sparsity and stochastic-EM multiple imputation), multi-start estimation, component-count selection, and prediction. The companion package 'mixqrgate' adds location-varying gating.

License MIT + file LICENSE

Encoding UTF-8

LazyData true

Depends R (>= 4.1)

Imports quantreg, stats, graphics, utils

Suggests ggplot2, rqPen, testthat (>= 3.0.0), knitr, rmarkdown

RoxygenNote 7.3.3

Roxygen list(markdown = TRUE)

Config/testthat/edition 3

Config/Needs/website pkgdown

VignetteBuilder knitr

URL <https://github.com/kvenkita/mixqr>,
<https://kvenkita.github.io/mixqr/>

BugReports <https://github.com/kvenkita/mixqr/issues>

Repository <https://kvenkita.r-universe.dev>

Date/Publication 2026-06-22 11:37:44 UTC

RemoteUrl <https://github.com/kvenkita/mixqr>

RemoteRef HEAD

RemoteSha 6c8036a7b9851014d4eb9d2ccf7434fde5a043be

Contents

coef.mixqr	3
confint.mixqr	3
engine	4
fitted.mixqr	5
get_mixqr_engine	5
list_mixqr_engines	6
logLik.mixqr	6
mixqr	7
mixqr_control	9
mixqr_nc	10
mixqr_pen	11
mixqr_select	12
mixqr_vcontrol	13
plot.mixqr	14
plot.mixqr_multitau	15
predict.mixqr	15
predict.mixqr_multitau	16
register_mixqr_engine	17
selectedVars	17
sim_mixqr_cross	18
sim_mixqr2	18
sim_mixqr3	19
summary.mixqr	19
vcov.mixqr	20
Index	21

coef.mixqr	<i>Component coefficients of a mixqr fit</i>
------------	--

Description

Component coefficients of a mixqr fit

Usage

```
## S3 method for class 'mixqr'
coef(object, include_pi = FALSE, ...)
```

Arguments

object	A "mixqr" object.
include_pi	If TRUE, return a list with both beta and the mixing probabilities pi.
...	Unused.

Value

A $(p+1) \times m$ coefficient matrix (one column per component), or a list if `include_pi = TRUE`.

confint.mixqr	<i>Confidence intervals for a mixqr fit</i>
---------------	---

Description

Wald intervals from the fitted standard errors. Requires `variance = "sparsity"` or `"stochEM"`; `"stochEM"` is recommended (sparsity SEs are classification-conditional and understate uncertainty).

Usage

```
## S3 method for class 'mixqr'
confint(object, parm, level = 0.95, ...)
```

Arguments

object	A "mixqr" object.
parm	Optional subset of parameter names.
level	Confidence level. Default 0.95.
...	Unused.

Value

A matrix of lower/upper bounds.

engine

Engine ethanol-combustion data (Brinkman 1981)

Description

Measurements from a single-cylinder automobile test engine burning ethanol, exhibiting two homogeneous regimes when the equivalence ratio is regressed on nitrous-oxide concentration – the engine example of Wu & Yao (2016, Fig. 5). Derived from the public-domain ethanol data redistributed in the `lattice` package (Brinkman 1981).

Usage

`engine`

Format

A data frame with 88 rows and 3 columns:

equivalence equivalence ratio: richness of the air/ethanol mix (response).

nox concentration of nitrous oxide (NO and NO₂) in the exhaust, normalized by engine work (predictor).

compression compression ratio of the engine.

Source

Brinkman, N. D. (1981). Ethanol fuel – a single-cylinder engine study of efficiency and exhaust emissions. *SAE Transactions* 90, 1410–1427. Redistributed via the `lattice` package ethanol dataset.

References

Wu, Q. and Yao, W. (2016). Mixtures of quantile regressions. *CSDA* 93, 162–176.

Examples

```
fit <- mixqr(equivalence ~ nox, data = engine, tau = 0.5, m = 2)
summary(fit)
```

fitted.mixqr	<i>Fitted values, residuals and number of observations</i>
--------------	--

Description

fitted() returns the classified conditional tau-quantile for each observation; residuals() the corresponding residuals (or, with type = "component", the full n x m residual matrix); nobs() the sample size.

Usage

```
## S3 method for class 'mixqr'
fitted(object, ...)

## S3 method for class 'mixqr'
residuals(object, type = c("hard", "component"), ...)

## S3 method for class 'mixqr'
nobs(object, ...)
```

Arguments

object	A "mixqr" object.
...	Unused.
type	For residuals: "hard" (classified) or "component".

Value

A numeric vector or matrix.

get_mixqr_engine	<i>Retrieve a registered mixqr engine constructor</i>
------------------	---

Description

Retrieve a registered mixqr engine constructor

Usage

```
get_mixqr_engine(name)
```

Arguments

name	Character engine name.
------	------------------------

Value

The constructor function.

list_mixqr_engines	<i>List registered mixqr engines</i>
--------------------	--------------------------------------

Description

List registered mixqr engines

Usage

```
list_mixqr_engines()
```

Value

Character vector of engine names.

logLik.mixqr	<i>Log-likelihood, AIC and BIC of a mixqr fit</i>
--------------	---

Description

Available for the parametric ALD engine (a genuine working likelihood); NA for the semiparametric kdEM engine, which has no global likelihood.

Usage

```
## S3 method for class 'mixqr'
logLik(object, ...)
```

```
## S3 method for class 'mixqr'
AIC(object, ..., k = 2)
```

```
## S3 method for class 'mixqr'
BIC(object, ...)
```

Arguments

object	A "mixqr" object.
...	Unused.
k	The penalty per parameter for AIC (default 2).

Value

A "logLik" object (logLik) or a numeric criterion (AIC, BIC).

Description

Estimates a finite mixture of tau-quantile regressions (clusterwise quantile regression) at a single quantile level. Two engines are available: a fast parametric asymmetric-Laplace mixture ("ald", genuine likelihood and AIC/BIC) and the kernel-density EM of Wu & Yao (2016) ("kdEM", non-parametric component error densities constrained to have their tau-th quantile = 0).

Usage

```

mixqr(
  formula,
  data,
  tau = 0.5,
  m = 2L,
  family = c("quantile", "expectile", "mquantile"),
  engine = "ald",
  error_density = c("unequal", "equal"),
  init = c("ald", "kmeans", "random", "manual"),
  nstart = 20L,
  control = mixqr_control(),
  weights = NULL,
  manual_init = NULL,
  variance = c("none", "sparsity", "stochEM"),
  vcontrol = mixqr_vcontrol(),
  ...
)

```

Arguments

formula	A model formula $y \sim x_1 + x_2$ (intercept implied).
data	A data frame.
tau	Quantile level in (0, 1). Default 0.5.
m	Number of mixture components (≥ 1). Default 2.
family	Component-loss family: "quantile" (default; check loss, the Wu & Yao model), "expectile" (asymmetric least squares, Newey & Powell 1987 – a smooth, crossing-free location device), or "mquantile" (asymmetric Huber, Breckling & Chambers 1988 – a robust expectile/quantile blend). A non-quantile family selects the matching engine.
engine	"ald" (default) or "kdEM", or the name of a custom engine registered via register_mixqr_engine() (e.g. "expectile", "mquantile").
error_density	For "kdEM": "unequal" (per-component densities, eq. 2.5) or "equal" (pooled density, eq. 2.8).

<code>init</code>	Initialisation strategy: "ald" (default; ALD pre-fit seeds the kdEM engine), "kmeans", "random", or "manual".
<code>nstart</code>	Number of multi-start initialisations (the mixture likelihood is multimodal). Default 20.
<code>control</code>	A <code>mixqr_control()</code> list.
<code>weights</code>	Optional prior observation weights.
<code>manual_init</code>	Optional $n \times m$ responsibility matrix for <code>init = "manual"</code> .
<code>variance</code>	Standard-error method: "none" (default), "sparsity" (eq. 3.3) or "stochEM" (Algorithm 3.1 multiple imputation).
<code>vcontrol</code>	A <code>mixqr_vcontrol()</code> list for <code>variance = "stochEM"</code> .
<code>...</code>	Reserved for engine extensions; currently unused.

Value

An object of class "mixqr".

Bias under asymmetric errors (Wu & Yao sec.6)

The semiparametric (kdEM) estimator solves estimating equations whose score $I(a \leq \theta) - \tau$ is not orthogonal to the nuisance tangent space of the unknown error densities, so it can be BIASED when component error densities are asymmetric and the clusters have imbalanced overlap (Wu & Yao 2016, sec.6, Fig.6). Watch `fit$diagnostics$overlap` (responsibility entropy; larger = more overlap = more bias risk) and cross-check against the parametric ald engine. Well-separated clusters and (near-)symmetric errors are the safe regime.

Standard errors

Sparsity SEs (`variance = "sparsity"`, eq.3.3) are CONDITIONAL on the fitted classification and understate uncertainty (they are flagged as such by `summary()`). Use `variance = "stochEM"` (the stochastic-EM multiple-imputation estimator, Algorithm 3.1) for inference; it propagates classification and mixing uncertainty.

References

Wu, Q. and Yao, W. (2016). Mixtures of quantile regressions. *CSDA* 93, 162–176.

Examples

```
set.seed(1)
d <- sim_mixqr2(n = 300)
fit <- mixqr(y ~ x, data = d, tau = 0.5, m = 2, engine = "ald", nstart = 10)
fit
coef(fit)
```

mixqr_control	<i>Control parameters for mixqr()</i>
---------------	---------------------------------------

Description

Control parameters for `mixqr()`

Usage

```

mixqr_control(
  tol = 1e-06,
  maxit = 500L,
  reltol = TRUE,
  bandwidth = NULL,
  kde_grid = 512L,
  label_order = "slope",
  distinct_tol = 0.001,
  trace = FALSE,
  seed = NULL
)

```

Arguments

<code>tol</code>	Convergence tolerance on the summed absolute parameter change (Wu & Yao 2016, p. 164). Default 1e-6.
<code>maxit</code>	Maximum EM iterations. Default 500.
<code>reltol</code>	Logical; if TRUE use the relative criterion $\sum d\theta / (\sum \theta + \text{eps}) < \text{tol}$ (recommended because π and β differ in scale). Default TRUE.
<code>bandwidth</code>	Optional numeric KDE bandwidth override (kdEM engine). If NULL, Silverman's rule $1.06 * \text{sd} * n^{(-1/5)}$ is used per component.
<code>kde_grid</code>	Number of grid points for stored KDE evaluation / plotting. Default 512.
<code>label_order</code>	Label-switching constraint: "slope" (ascending first slope coordinate, default – aligned with the distinct-slope identifiability of Wu & Yao Thm 2.1), "intercept", or "slope:k" for the k-th slope.
<code>distinct_tol</code>	RELATIVE threshold below which component slope vectors are flagged as near-coincident (near-violation of Wu & Yao Thm 2.1), measured as $\ b_a - b_b\ / (\ b_a\ + \ b_b\)$. Default 1e-3.
<code>trace</code>	Logical; print iteration progress. Default FALSE.
<code>seed</code>	Optional integer RNG seed for reproducible multi-start.

Value

A list of class "mixqr_control".

mixqr_nc	<i>Fit a joint multi-tau mixture of quantile regressions (non-crossing, shared labels)</i>
----------	--

Description

Estimates a finite mixture of quantile regressions at a *vector* of quantile levels τ jointly, with one latent classification shared across all levels (a coupled E-step that pools the per-level component likelihoods) and guaranteed non-crossing of the within-component quantile curves. This closes the two problems Wu & Yao (2016, sec.5) leave open: within-component crossing and cross-tau classification ambiguity.

Usage

```

mixqr_nc(
  formula,
  data,
  m = 2L,
  tau = c(0.25, 0.5, 0.75),
  noncrossing = c("rearrange", "none"),
  class_coupling = c("pool", "median"),
  nstart = 10L,
  control = mixqr_control(),
  weights = NULL
)

```

Arguments

`formula`, `data`, `m`, `nstart`, `control`, `weights`
 As in `mixqr()`.

`tau` Increasing vector of quantile levels (length ≥ 2).

`noncrossing` "rearrange" (default; monotone-rearrange the fitted per-tau quantiles within each component, Chernozhukov et al. 2010) or "none" (legacy per-tau fits, which may cross – for diagnosis/comparison).

`class_coupling` How the shared E-step pools over τ : "pool" (default; geometric mean, weights $1/L$) or "median" (classify on the median level only).

Value

An object of class `c("mixqr_multitau", "mixqr")` with a coefficient array `coefficients` ($[p+1, m, L]$), the shared posterior/classification, `mix_prop`, the τ grid, and a crossing diagnostic (raw crossings, 0 after rearrangement).

References

Wu, Q. and Yao, W. (2016). Mixtures of quantile regressions. *CSDA* 93, 162–176. Chernozhukov, V., Fernandez-Val, I. and Galichon, A. (2010). Quantile and probability curves without crossing. *Econometrica* 78, 1093–1125.

Examples

```
d <- sim_mixqr_cross(n = 160, seed = 1)
fit <- mixqr_nc(y ~ x, data = d, m = 2, tau = c(0.1, 0.25, 0.5, 0.75, 0.9))
fit$crossing      # raw crossings -> 0 after rearrangement
predict(fit)[1, , ] # non-crossing quantiles for observation 1
```

mixqr_pen

Fit a penalized (variable-selecting) mixture of quantile regressions

Description

Component-specific penalized variable selection for the `mixqr()` model: each latent component gets its own sparse slope vector, a covariate active in one quantile-regression cluster and dropped in another. The weighted check-loss M-step gains a SCAD / adaptive-LASSO / LASSO / MCP penalty (intercept free); the penalty strength λ is selected by a mixture BIC over a path, and components whose mixing weight falls below π_{\min} are pruned.

Usage

```
mixqr_pen(
  formula,
  data,
  tau = 0.5,
  m = 2L,
  penalty = c("SCAD", "aLASSO", "LASSO", "MCP"),
  lambda = NULL,
  nlambda = 40L,
  a = 3.7,
  alasso_gamma = 1,
  penalty_factor = NULL,
  pi_min = 0.02,
  nstart = 10L,
  control = mixqr_control(),
  weights = NULL
)
```

Arguments

formula, data, tau, m, nstart, control, weights

As in `mixqr()`.

penalty One of "SCAD" (default), "aLASSO", "LASSO", "MCP". "aLASSO" is implemented as a LASSO with data-driven penalty factors $1 / |b_{\text{pilot}}|^{\gamma}$ from a responsibility-weighted unpenalised pilot.

lambda Optional fixed penalty (skips selection). If NULL, a path of nlambda values is built and tuned by BIC.

nlambda	Path length when lambda is NULL. Default 40.
a	SCAD/MCP concavity. Default 3.7.
alasso_gamma	Adaptive-LASSO weight exponent (default 1).
penalty_factor	Optional length-p per-slope multipliers (0 never penalises a covariate). NULL penalises all slopes equally.
pi_min	Components with mixing weight below this are pruned. Default 0.02.

Value

A "mixqr" object with an extra \$selection slot (active covariates per component, chosen lambda, the BIC path, and df).

References

Khalili, A. and Chen, J. (2007). Variable selection in finite mixture of regression models. *JASA* 102, 1025–1038. Sherwood, B., Li, S. and Maidman, A. (2025). rqPen. *R Journal*.

Examples

```
set.seed(1)
n <- 250; p <- 8; x <- matrix(rnorm(n * p), n)
colnames(x) <- paste0("x", 1:p)
z <- rbinom(n, 1, 0.5)
y <- ifelse(z == 0, 1 + 2 * x[, 1], -1 + 2 * x[, 2]) + rnorm(n)
d <- data.frame(y = y, x)
fit <- mixqr_pen(y ~ ., data = d, tau = 0.5, m = 2, penalty = "SCAD")
selectedVars(fit)
```

mixqr_select

Select the number of mixture components

Description

Fits mixqr over a range of component counts and scores them by an information criterion or cross-validated check loss.

Usage

```
mixqr_select(
  formula,
  data,
  tau = 0.5,
  m = 1:4,
  criterion = c("BIC", "AIC", "cv"),
  engine = "ald",
  error_density = c("unequal", "equal"),
```

```

    folds = 5L,
    weights = NULL,
    nstart = 20L,
    control = mixqr_control()
  )

```

Arguments

formula, data, tau, weights	Passed to <code>mixqr()</code> .
m	Integer vector of component counts to try. Default 1:4.
criterion	"BIC" (default), "AIC", or "cv".
engine	Engine to use ("ald" or "kdEM"). For "AIC"/"BIC" the score is always computed with the ALD working likelihood.
error_density	For the kdEM engine.
folds	Number of CV folds when criterion = "cv". Default 5.
nstart	Multi-starts per fit.
control	A <code>mixqr_control()</code> list.

Details

For criterion `%in% c("AIC", "BIC")` the score uses the parametric ALD working-likelihood (the semiparametric kdEM engine has no global likelihood; Wu & Yao 2016, p. 164). Note that AIC/BIC for the *number of mixture components* is heuristic: testing m vs $m+1$ places a component on the parameter-space boundary ($\pi_{i,j} = 0$), so the usual penalty asymptotics do not hold (McLachlan & Peel 2000, ch. 6; Chen, Chen & Kalbfleisch 2001). The "cv" criterion (cross-validated held-out check loss) avoids the likelihood entirely and works for either engine.

Value

A list with `table` (data frame of scores by m), `best` (chosen m), `criterion`, and `fit` (the chosen model refit on all data).

<code>mixqr_vcontrol</code>	<i>Control parameters for stochastic-EM variance estimation (Algorithm 3.1)</i>
-----------------------------	---

Description

Control parameters for stochastic-EM variance estimation (Algorithm 3.1)

Usage

```
mixqr_vcontrol(B = 500L, burnin = 50L, thin = 1L, seed = NULL)
```

Arguments

B	Number of imputation draws. Default 500.
burnin	Burn-in draws discarded before accumulation. Default 50.
thin	Thinning interval. Default 1.
seed	Optional integer RNG seed.

Value

A list of class "mixqr_vcontrol".

plot.mixqr	<i>Plot a mixqr fit</i>
------------	-------------------------

Description

Draws the data coloured by hard classification with the component quantile-regression lines, and a panel of the estimated component error densities (mirroring Wu & Yao Figs. 1, 3–5).

Usage

```
## S3 method for class 'mixqr'
plot(x, which = c("both", "fit", "density"), ...)
```

Arguments

x	A "mixqr" object.
which	"both" (default), "fit", or "density".
...	Passed to <code>graphics::plot()</code> .

Value

Invisibly x.

plot.mixqr_multitau *Plot the non-crossing component quantile curves of a multi-tau fit*

Description

Overlays, for each component, the (rearranged, non-crossing) fitted quantile curves at every tau against the first covariate.

Usage

```
## S3 method for class 'mixqr_multitau'
plot(x, ...)
```

Arguments

x A `mixqr_nc()` fit.
 ... Passed to `graphics::matplot()`.

Value

x, invisibly.

predict.mixqr *Predict from a mixqr fit*

Description

Predict from a mixqr fit

Usage

```
## S3 method for class 'mixqr'
predict(
  object,
  newdata = NULL,
  type = c("quantile", "quantile_byclass", "posterior", "class", "density"),
  ...
)
```

Arguments

object	A "mixqr" object.
newdata	Optional data frame. If omitted, the training data are used.
type	One of "quantile" (per-component conditional tau-quantiles, an $n \times m$ matrix), "quantile_byclass" (the conditional tau-quantile of each point's most likely component, a vector), "posterior" (responsibilities p_{ij}), "class" (hard labels), or "density" (the component error-density objects).
...	Unused.

Value

A matrix, vector, or list depending on type.

Classifying genuinely new x

"posterior", "class", and "quantile_byclass" require the RESPONSE in newdata, because component membership is inferred from the residual densities. This core therefore cannot assign a class to a covariate vector x alone; covariate-based gating (membership as a function of x) is provided by the location-varying-gating extension (QMM sub-project 03).

predict.mixqr_multitau

Predict non-crossing component quantiles from a multi-tau fit

Description

Predict non-crossing component quantiles from a multi-tau fit

Usage

```
## S3 method for class 'mixqr_multitau'
predict(object, newdata = NULL, ...)
```

Arguments

object	A <code>mixqr_nc()</code> fit.
newdata	Optional data frame; defaults to the training design.
...	Unused.

Value

An $[n, m, L]$ array of fitted quantiles. With `noncrossing = "rearrange"` each component's L quantiles are sorted to be nondecreasing in tau (Chernozhukov et al. 2010), guaranteeing no crossing.

register_mixqr_engine *Register a mixqr EM engine*

Description

An engine is a constructor function(`tau`, `m`, `control`) returning a list of closures with the FROZEN signatures documented in `mixqr_engine_contract()`. Sub-projects 03–06 of the QMM suite ship engines this way without forking the driver.

Usage

```
register_mixqr_engine(name, constructor)
```

Arguments

<code>name</code>	Character engine name (e.g. "ald", "kdEM").
<code>constructor</code>	Function (<code>tau</code> , <code>m</code> , <code>control</code>) -> <code>mixqr_engine</code> (a list of closures: <code>estep</code> , <code>mstep_pi</code> , <code>mstep_beta</code> , <code>update_density</code> , <code>loglik</code> , <code>npar</code> , plus name).

Value

Invisibly TRUE.

`selectedVars` *Active (selected) covariates per component of a penalized mixqr fit*

Description

Active (selected) covariates per component of a penalized mixqr fit

Usage

```
selectedVars(object)
```

Arguments

<code>object</code>	A <code>mixqr_pen()</code> fit.
---------------------	---------------------------------

Value

A named list (one entry per component) of selected covariate names.

sim_mixqr_cross	<i>Simulate a two-component mixture whose per-tau fits cross (for O4 validation)</i>
-----------------	--

Description

One component is well-behaved; the other has a near-flat slope and strong heteroscedasticity (error scale growing across the design), so its independently fitted per-tau quantile lines cross in finite samples – exactly the within-component crossing Wu & Yao (2016, sec.5) flag as common "when the sample size is small." Use to demonstrate that noncrossing = "none" crosses while "rearrange" does not. (A valid location-scale model has non-crossing true quantiles; crossing here is the finite-sample estimation artefact the method repairs.)

Usage

```
sim_mixqr_cross(n = 160, seed = NULL)
```

Arguments

n	Sample size (default 160; crossing is a small-sample phenomenon).
seed	Optional RNG seed.

Value

A data frame with columns y, x, and the true label z.

sim_mixqr2	<i>Simulate the Wu & Yao two-component design (eq. 4.1-4.2)</i>
------------	---

Description

$Y = 10 - 10x + e$ (component 1) or $Y = -10 + 10x + e$ (component 2), with $\pi_i = (0.5, 0.5)$, $x \sim U(0, 1)$, and a bimodal error $e \sim 0.5 N(-1, 1^2) + 0.5 N(2, 2^2)$ (median 0; median regression, $\tau = 0.5$).

Usage

```
sim_mixqr2(n = 300L, pi1 = 0.5)
```

Arguments

n	Sample size.
pi1	Probability of component 1.

Value

A data frame with columns x, y, z (true component label).

sim_mixqr3	<i>Simulate the Wu & Yao three-component design (eq. 4.3-4.4)</i>
------------	---

Description

Three components with $\pi = (1/3, 1/3, 1/3)$, covariates $x_1, x_2 \sim U(0, 1)$, means $-20 x_1 - 20 x_2$, $0, 20 x_1 + 20 x_2$, and shifted-lognormal errors $\exp(N(1, \sigma^2)) - \exp(1)$ with σ^2 in $\{1, 0.5, 0.25\}$ (each median 0).

Usage

```
sim_mixqr3(n = 300L)
```

Arguments

n Sample size.

Value

A data frame with columns x1, x2, y, z.

summary.mixqr	<i>Summarize a mixqr fit</i>
---------------	------------------------------

Description

Prints, per component, the coefficient estimates with standard errors, z-values and p-values (when a variance method was used), the mixing probabilities, and the separability / overlap diagnostics. Sparsity standard errors are flagged as classification-conditional.

Usage

```
## S3 method for class 'mixqr'
summary(object, ...)
```

Arguments

object A "mixqr" object.
... Unused.

Value

An object of class "summary.mixqr".

`vcov.mixqr`*Variance-covariance matrix of a mixqr fit*

Description

The estimated parameter covariance (requires a fitted variance method).

Usage

```
## S3 method for class 'mixqr'  
vcov(object, ...)
```

Arguments

<code>object</code>	A "mixqr" object.
<code>...</code>	Unused.

Value

The variance-covariance matrix.

Index

- * **datasets**
 - engine, 4
- AIC.mixqr (logLik.mixqr), 6
- BIC.mixqr (logLik.mixqr), 6
- coef.mixqr, 3
- confint.mixqr, 3
- engine, 4
- fitted.mixqr, 5
- get_mixqr_engine, 5
- graphics::matplot(), 15
- graphics::plot(), 14
- list_mixqr_engines, 6
- logLik.mixqr, 6
- mixqr, 7
- mixqr(), 9–11, 13
- mixqr_control, 9
- mixqr_control(), 8, 13
- mixqr_engine_contract(), 17
- mixqr_nc, 10
- mixqr_nc(), 15, 16
- mixqr_pen, 11
- mixqr_pen(), 17
- mixqr_select, 12
- mixqr_vcontrol, 13
- mixqr_vcontrol(), 8
- nobs.mixqr (fitted.mixqr), 5
- plot.mixqr, 14
- plot.mixqr_multitau, 15
- predict.mixqr, 15
- predict.mixqr_multitau, 16
- register_mixqr_engine, 17
- register_mixqr_engine(), 7
- residuals.mixqr (fitted.mixqr), 5
- selectedVars, 17
- sim_mixqr2, 18
- sim_mixqr3, 19
- sim_mixqr_cross, 18
- summary.mixqr, 19
- vcov.mixqr, 20