Package: kir (via r-universe)

November 19, 2026

Title Knowledge Integration Regression for Mean and Quantile Models

Version 0.3.0

Description Combines observed data with prior knowledge, supplied as coefficients or encoded from lists of important variables, for sparse mean and quantile regression. A data-driven knowledge weight controls the contribution of prior predictions. Knowledge Integration Quantile Regression (KIQR) uses a Huber-smoothed quantile loss with smoothly clipped absolute deviation (SCAD) or least absolute shrinkage and selection operator (LASSO) penalties, as described in Wang, Lian, Yu, Zu and Zhang (2026) <doi:10.48550/arXiv.2406.12212>. Mean regression implements the prior LASSO of Jiang, He and Zhang (2016) <doi:10.1080/01621459.2015.1008363> and a prior SCAD counterpart using 'ncvreg'. Provides fixed-parameter fitting and tuning by cross-validation or a high-dimensional Bayesian information criterion; zero-weight quantile fits use 'rqPen'.

URL https://arxiv.org/abs/2406.12212v3

License MIT + file LICENSE

Encoding UTF-8

Imports ncvreg, quantreg, rqPen

Suggests glmnet, testthat (>= 3.1.7)

Config/testthat/edition 3

NeedsCompilation no

Config/roxygen2/version 8.1.1

Author Jiantong Wang [aut, cre], Tianhai Zu [aut, ctb], Yan Yu [aut, ctb]

Maintainer Jiantong Wang <wangj11@xavier.edu>

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

Date/Publication 2026-11-18 20:50:07 UTC

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

RemoteRef HEAD

RemoteSha 895bae5c70da5c63a7331c4b84b7be0c6380ea6d

Contents

kir-package . . . 2
coef.kiqr_fit . . . 3
kiqr . . . 4
kiqr_cv . . . 6
kiqr_fit . . . 10
kiqr_hbic . . . 13
kir . . . 15
kir-methods . . . 18
kir_cv . . . 19
kir_fit . . . 22
kir_hbic . . . 24
predict.kiqr_fit . . . 26
print.kiqr_fit . . . 26
summary.kiqr_fit . . . 27

Index **29**

`kir-package` *kir: Knowledge Integration Regression for Mean and Quantile Models*

Description

Fits sparse mean and quantile regression models that combine observed responses with prior knowledge supplied as prior coefficients. A knowledge weight `zeta` in `[0, 1]`, tuned by cross-validation or a high-dimensional Bayesian information criterion (HBIC), controls how much the prior predictions contribute. At `zeta = 0`, the fit is ordinary penalized regression on the observed data. At `zeta = 1`, the fit uses only the prior predictions. Quantile models (`kiqr()`, Knowledge Integration Quantile Regression) use a Huber-smoothed check loss at positive knowledge weights; mean models (`kir()`, Knowledge Integration Regression) use least squares. Both offer smoothly clipped absolute deviation (SCAD, default) and LASSO penalties.

kiqr functions

For quantile regression, `kiqr()` supports fitting at fixed tuning values and selecting tuning parameters by cross-validation or a high-dimensional Bayesian information criterion (HBIC). The direct interfaces `kiqr_fit()`, `kiqr_cv()`, and `kiqr_hbic()` are also available. Fitted objects support `stats::coef()`, `stats::predict()`, `base::summary()`, and `base::print()`. See the individual function help pages for input requirements, tuning options, and implementation details.

kir functions

`kir()`, `kir_fit()`, `kir_cv()`, and `kir_hbic()` provide the mean-regression counterpart (Knowledge Integration Regression): the prior LASSO of Jiang, He and Zhang (2016) and a prior SCAD estimator, fitted by penalized least squares on a pseudo-response with 'ncvreg'.

Author(s)

Maintainer: Jiantong Wang <wangj11@xavier.edu>

Authors:

- Jiantong Wang <wangj11@xavier.edu>
- Tianhai Zu [contributor]
- Yan Yu [contributor]

References

Wang, J., Lian, H., Yu, Y., Zu, T., and Zhang, H. (2026). Identifying Genetic Variants for Obesity: A Knowledge Integration Quantile Regression (KIQR) Approach for Ultra-High-Dimensional Data. arXiv:2406.12212v3. doi:10.48550/arXiv.2406.12212.

Jiang, Y., He, Y., and Zhang, H. (2016). Variable Selection With Prior Information for Generalized Linear Models via the Prior LASSO Method. Journal of the American Statistical Association, 111(513), 355–376. doi:10.1080/01621459.2015.1008363.

Breheny, P., and Huang, J. (2011). Coordinate Descent Algorithms for Nonconvex Penalized Regression, with Applications to Biological Feature Selection. The Annals of Applied Statistics, 5(1), 232–253. doi:10.1214/10AOAS388.

Sherwood, B., Li, S., and Maidman, A. (2025). rqPen: An R Package for Penalized Quantile Regression. The R Journal, 17(2), 146–175. doi:10.32614/RJ2025017.

Yi, C., and Huang, J. (2017). Semismooth Newton Coordinate Descent Algorithm for Elastic-Net Penalized Huber Loss Regression and Quantile Regression. Journal of Computational and Graphical Statistics, 26(3), 547–557. doi:10.1080/10618600.2016.1256816.

See Also

`kiqr()`, `kiqr_fit()`, `kiqr_cv()`, `kiqr_hbic()`, `kir()`, `kir_fit()`, `kir_cv()`, `kir_hbic()`

`coef.kiqr_fit` *Extract KIQR Model Coefficients*

Description

Return the fitted intercept and slopes. For CV objects, choose between the minimum-loss and one-standard-error fits; the default is `"min"`.

Usage

```
## S3 method for class 'kiqr_fit'
coef(object, ...)

## S3 method for class 'kiqr_cv'
coef(object, rule = c("min", "1se"), ...)
```

Arguments

`object`	A fitted KIQR object.
`...`	Additional arguments, currently unused.
`rule`	For a cross-validated model, return the minimum-loss (`"min"`) or one-standard-error (`"1se"`) coefficients.

Value

A named numeric coefficient vector.

`kiqr` *Fit or Tune a Knowledge Integration Quantile Regression Model*

Description

`kiqr()` fits Knowledge Integration Quantile Regression models using cross-validation (CV), a high-dimensional Bayesian information criterion (HBIC), or a fixed pair of tuning values. It delegates estimation to `kiqr_cv()`, `kiqr_hbic()`, or `kiqr_fit()`, respectively. At `zeta = 0`, the model reduces to standard penalized quantile regression using only the observed responses.

Usage

```
kiqr(
  x,
  y,
  beta_prior,
  tau,
  lambda = NULL,
  zeta = NULL,
  method = c("cv", "hbic", "fit"),
  ...
)
```

Arguments

`x`	A nonempty numeric predictor matrix or data frame with finite, non-missing values and no intercept column. Column names, if supplied, must be unique. Predictors are not automatically standardized.
`y`	A finite numeric response vector with one value per row of `x`.
`beta_prior`	A numeric prior coefficient vector containing the intercept followed by one finite coefficient per column of `x`, in exactly the same feature order, coding, and units. It must be for the same quantile level as the fit. Coefficient names do not trigger automatic reordering.
`tau`	A single quantile level strictly between 0 and 1.

`lambda` Optional finite, nonnegative penalty tuning value or candidate vector. In a tuning mode, omitted or `NULL` values generate an automatic path.

`zeta` Optional knowledge weight or candidate vector in `[0, 1]`. In a tuning mode, omitted or `NULL` values use `seq(0, 1, by = 0.1)`.

`method` Selection method: `"cv"`, `"hbic"`, or `"fit"`. If omitted, use `"fit"` when both `lambda` and `zeta` have length one; otherwise use `"cv"`. An explicitly supplied method always takes precedence.

`...` Named additional arguments supported by the selected method.

Common options

- Initialization: `beta_init`.
- Penalty: `penalty` (`"SCAD"` or `"LASSO"`); factors: `penalty_factor`.
- Smoothing: `huber_gamma`.
- Iteration limit: `max_iter`.
- Compatibility: `half_life` (ignored).
- rqPen settings: `rqpen_control`.

Additional CV options Fold configuration: `folds` and `foldid`. Fold-specific priors: `beta_prior_fold_fun`. Parallel execution: `cores`. Automatic grids: `nlambda`, `lambda_min_ratio`, and `lambda_scale`.

Additional HBIC options `cores`, `nlambda`, `lambda_min_ratio`, and `lambda_scale`. The latter three control automatic HBIC lambda paths and are ignored when a lambda grid is supplied.

Additional fixed-fit options `scad_a`, `tol`, and `lambda_path`.

Unsupported, unnamed, or duplicated additional arguments produce an error. `beta_init` initializes a fixed fit or HBIC candidates; in CV it is used only for the final full-data refits. Zero-weight fits do not use it.

Details

With `method = "fit"`, both tuning arguments must be supplied as single values. With `method = "cv"` or `"hbic"`, a single value fixes that parameter while a vector supplies its candidate grid. Each omitted grid is generated independently. In particular, supplying just one single-valued parameter still performs tuning over the other one.

CV delegates to `kiqr_cv()`. HBIC uses the same automatic lambda-path construction and default zeta grid as CV, then delegates to `kiqr_hbic()`. The direct `kiqr_hbic()` interface requires both grids. HBIC requires at least three predictors (excluding the intercept), at least two observations, and finite, strictly positive total check loss for every scored candidate. Fixed fits delegate to `kiqr_fit()`. Prior coefficients retain the same intercept, ordering, coding, units, and quantile requirements as those interfaces. Formula input is not supported. See `kiqr_cv()` for fold-specific priors and the 1-SE rule, and `kiqr_fit()` for solver and endpoint behavior.

Value

A `kiqr_cv` object for CV, or a `kiqr_fit` object for HBIC or a fixed fit. The `coef()`, `predict()`, `summary()`, and `print()` methods apply. CV retains both the `"min"` and `"1se"` fits. The recorded call is the user's call to `kiqr()`. HBIC's `tuning` component records the candidate grids, whether they were supplied or generated automatically, and automatic path metadata.

Solver warnings

Automatic lambda construction and default initialization call rqPen and quantreg routines, which may warn that a solution is nonunique. For example, an intercept-only median regression with an even sample size has an interval of optimal intercepts. Initialization warnings can repeat across HBIC candidates. A nonunique-solution warning alone does not establish whether the final PGD fit converged. Check the returned `convergence` component and, for CV, `param_grid$eligible` and `fold_diagnostics`. For `zeta = 0` fits, solver convergence is reported as unavailable. Warnings are not suppressed.

See Also

`kiqr_cv()`, `kiqr_hbic()`, `kiqr_fit()`, `kir()`

Examples

```
set.seed(42)
n <- 41
x <- cbind(age = rnorm(n), marker = rnorm(n))
beta_prior <- c(0.5, 0.7, 0)
y <- drop(cbind(1, x) %*% c(1, 1, -0.5) + rnorm(n, sd = 0.2))
foldid <- rep(1:3, length.out = n)

# Both single values select a fixed fit automatically.
fit <- kiqr(x, y, beta_prior, tau = 0.5, lambda = 0.01, zeta = 0.25,
            beta_init = beta_prior, huber_gamma = 0.1)
coef(fit)
predict(fit, newdata = x[1:2, , drop = FALSE])

# Default CV with an explicit candidate grid and reproducible folds.
cv <- kiqr(x, y, beta_prior, tau = 0.5,
           lambda = c(0.01, 0.02), zeta = c(0, 0.5),
           foldid = foldid, huber_gamma = 0.1, max_iter = 3000)
coef(cv, rule = "1se")

# Automatic lambda path with a small candidate grid, using the same data.
cv_auto <- kiqr(x, y, beta_prior, tau = 0.5, nlambda = 3,
                zeta = c(0, 0.5, 1), foldid = foldid, cores = 1,
                huber_gamma = 0.1)
summary(cv_auto, rule = "1se")
```

---

kiqr_cv *Tune a KIQR Model Using Cross-Validation*

Description

`kiqr_cv()` evaluates a grid of `lambda` and `zeta` values using k-fold cross-validation. Both the minimum-loss fit and the one-standard-error fit are retained in the returned object. Every eligible candidate must complete the same set of folds with finite validation losses. Positive-weight fits must satisfy the PGD convergence criterion; zero-weight fits must return finite coefficients and predictions.

Usage

```
kiqr_cv(
  x,
  y,
  beta_prior,
  tau,
  lambda = NULL,
  zeta = NULL,
  folds = 5,
  foldid = NULL,
  beta_init = NULL,
  penalty_factor = NULL,
  cores = 1,
  huber_gamma = 0.01,
  half_life = 100,
  max_iter = 10000,
  beta_prior_fold_fun = NULL,
  rqpen_control = list(),
  nlambda = 30,
  lambda_min_ratio = NULL,
  penalty = c("SCAD", "LASSO"),
  lambda_scale = c("observed", "combined")
)
```

Arguments

`x`	A nonempty numeric predictor matrix or data frame with finite, non-missing values and no intercept column. Column names, if supplied, must be unique. Predictors are not automatically standardized.
`y`	A finite numeric response vector with one value per row of `x`.
`beta_prior`	A numeric prior coefficient vector containing the intercept followed by one finite coefficient per column of `x`, in exactly the same feature order, coding, and units. It must be for the same quantile level as the fit. Coefficient names do not trigger automatic reordering.
`tau`	A single quantile level strictly between 0 and 1.
`lambda`	Optional numeric vector of nonnegative penalty tuning values. If omitted or `NULL`, construct a shared logarithmic path automatically.
`zeta`	Optional knowledge weights in `[0, 1]`. If omitted or `NULL`, use `seq(0, 1, by = 0.1)`. See `kiqr_fit()` for their meaning.
`folds`	A positive whole number of cross-validation folds, between 2 and the number of observations. Ignored when `foldid` is supplied.
`foldid`	Optional positive whole-number vector assigning each observation to a fold. IDs must be consecutive starting at 1. When supplied, the number of unique IDs determines the number of folds and `folds` is ignored. The assignments are returned with the fitted object.

`beta_init` Optional initial coefficient vector for the two final full-data refits only. CV fold fits always initialize from their training data and fold-specific prior; they do not use this vector.

`penalty_factor` Optional nonnegative penalty multiplier for each predictor, in column order; `NULL` uses all ones. A zero value leaves that predictor unpenalized. The intercept is never penalized.

`cores` A positive whole number of worker processes used for the grid search. Use `cores = 1` on Windows, where fork-based parallelism is unavailable.

`huber_gamma` A positive Huber smoothing parameter.

`half_life` Deprecated compatibility argument. It is ignored and has no effect because the solver selects its step size by backtracking.

`max_iter` A positive whole-number limit on PGD iterations per fit (default 10000).

`beta_prior_fold_fun`
Optional function with arguments `x`, `y`, and `tau` that re-estimates prior coefficients within each training fold. It must return the intercept followed by one finite coefficient per predictor in the same feature order as `x`. It is called once per fold, and its result is shared by all positive-weight candidates evaluated on that fold. It is not called for a grid containing only zero weights. It does not replace the supplied `beta_prior` used for the final full-data refits.

`rqpen_control` Named list of rqPen endpoint options: `alg` (default `"huber"`; also `"br"`), `scalex` (default `FALSE`), `max.iter` (5000), `converge.eps` (1e-4), and `coef.cutoff` (1e-8). The chosen `penalty`, `lambda.discard = FALSE`, and supplied penalty factors are fixed by the kir package. These options control `zeta = 0` fits and the rqPen baseline used to construct automatic lambda paths in tuning modes. They do not control positive-weight PGD fits or their automatic initializers. PGD `max_iter`, `tol`, `half_life`, and `huber_gamma` do not affect a zero-weight fit on a supplied path. Match the path and options when comparing to standalone rqPen.

`nlambda` Number of automatic lambda values, at least 2; default 30. Ignored when `lambda` is supplied. Degenerate automatic paths contain only zero when lambda is irrelevant or all gradient thresholds are zero.

`lambda_min_ratio`
Optional ratio of the smallest automatic lambda to the observed-data baseline upper bound. Defaults to 0.01 for `n >= p` and 0.05 for `n < p`; must lie strictly between 0 and 1. If the observed-data baseline is zero, use the mixed-loss upper bound instead. Ignored when `lambda` is supplied.

`penalty` The penalty, either `"SCAD"` (default) or `"LASSO"`. Names are case-sensitive.

`lambda_scale` How the upper end of an automatic lambda path is chosen. `"observed"` (default) uses the observed data only, so the path does not depend on the prior or on the zeta grid, and the zeta = 0 candidates match ordinary penalized quantile regression with the same automatic path. `"combined"` also covers thresholds that depend on the prior and the zeta grid. Ignored when `lambda` is supplied.

Details

The minimum rule selects the eligible candidate with the smallest mean validation check loss. The 1-SE rule fixes `zeta` at the minimum-rule value and selects the largest eligible `lambda` at that `zeta`

whose mean loss is within one standard error of the minimum. It does not maximize `zeta`. Fold losses are mean check losses, and their standard error is the sample standard deviation across folds divided by the square root of the number of folds; each fold has equal weight. If the prior is estimated using the observed responses, use `beta_prior_fold_fun` to estimate it within each training fold. Supply the corresponding full-data prior as `beta_prior` for the final refits. Observed-data initialization for CV fold fits is computed once per training fold and shared across positive-weight tuning candidates. Zero-weight candidates instead reuse one rqPen path per training fold and one full-data path for final refits. A prior callback failure does not invalidate the zero-weight candidates; a grid containing only zero does not invoke it. rqPen endpoint eligibility requires finite coefficients and validation losses; it does not certify PGD stationarity.

The automatic lambda path uses the observed-data upper bound from rqPen's public LASSO interface as its baseline, respecting `penalty_factor` and `rqpen_control`. If rqPen cannot construct a baseline, an observed-data Huber-gradient threshold is used and the reason is saved in `tuning`. With `lambda_scale = "combined"`, the upper bound also covers mixed-loss gradient thresholds at all requested weights strictly between zero and one, with a 5 percent margin. Each threshold is calculated after fitting the intercept and unpenalized predictors while fixing penalized slopes to zero, then dividing the penalized gradient magnitudes by `(1 - zeta) * penalty_factor`. Both penalties have derivative `lambda` at zero. These are local stationarity scales for SCAD; for the convex LASSO objective they give the zero-slope optimality threshold. The lower bound remains on the observed-data scale even when prior information enlarges the upper bound. Path construction uses the supplied full-data inputs once; fold fits and learned fold priors continue to use only their training rows. No automatic predictor standardization is added. To keep the candidate grids independent of held-out responses, supply prespecified `lambda` and `zeta` values instead of automatic grids. The minimum-rule selection warns if it reaches an automatic path boundary. Lambda does not affect the prior-only (`zeta = 1`) estimator; its fold evaluation is shared across lambda values, as is its final refit when both selection rules choose that endpoint.

Value

An object of class `kiqr_cv`. Use `coef()`, `predict()`, or `summary()` with `rule = "min"` or `rule = "1se"`. The object includes both coefficient vectors and tuning pairs, the candidate grid, fold-loss matrix, fold assignments, candidate eligibility information, detailed per-fold diagnostics, final-refit convergence diagnostics, and `tuning` metadata containing the actual grids, whether they were supplied or generated automatically, automatic bounds, gradient thresholds, and boundary selections.

Examples

```
set.seed(42)
n <- 41
x <- cbind(age = rnorm(n), marker = rnorm(n))
y <- drop(cbind(1, x) %*% c(1, 1, -0.5) + rnorm(n, sd = 0.2))
# A supplied prior independent of y, including the intercept.
beta_prior <- c(0.5, 0.7, 0)
foldid <- rep(1:3, length.out = n)

# Explicit candidate grids keep this three-fold example small.
cv <- kiqr_cv(
  x, y, beta_prior, tau = 0.5,
  lambda = c(0.01, 0.02), zeta = c(0, 0.5), foldid = foldid,
  huber_gamma = 0.1, max_iter = 3000
```

```
)
coef(cv, rule = "min")
coef(cv, rule = "1se")
summary(cv, rule = "1se")
predict(cv, newdata = x[1:2, , drop = FALSE], rule = "1se")
```

`kiqr_fit` *Fit a Knowledge Integration Quantile Regression Model*

Description

`kiqr_fit()` is the user-facing interface for fitting one KIQR model at a specified pair of `lambda` and `zeta` values. It computes the prior predictions internally and uses a Huber-smoothed quantile loss with a backtracking proximal-gradient (PGD) solver for positive knowledge weights. At `zeta = 0`, KIQR reduces to standard penalized quantile regression using only the observed responses.

Usage

```
kiqr_fit(
  x,
  y,
  beta_prior,
  tau,
  lambda,
  zeta,
  beta_init = NULL,
  penalty_factor = NULL,
  huber_gamma = 0.01,
  scad_a = 3.7,
  max_iter = 10000,
  tol = 1e-04,
  half_life = 100,
  lambda_path = NULL,
  rqpen_control = list(),
  penalty = c("SCAD", "LASSO")
)
```

Arguments

`x`	A nonempty numeric predictor matrix or data frame with finite, non-missing values and no intercept column. Column names, if supplied, must be unique. Predictors are not automatically standardized.
`y`	A finite numeric response vector with one value per row of `x`.
`beta_prior`	A numeric prior coefficient vector containing the intercept followed by one finite coefficient per column of `x`, in exactly the same feature order, coding, and units. It must be for the same quantile level as the fit. Coefficient names do not trigger automatic reordering.

tau	A single quantile level strictly between 0 and 1.
lambda	A single nonnegative penalty tuning parameter.
zeta	A single knowledge weight in [0, 1]. At zero, KIQR reduces to standard penalized quantile regression using only the observed responses; at one, it uses only the prior predictions and no penalty. For intermediate values, the observed-response loss and chosen penalty have weight 1 - zeta, and the prior-prediction loss has weight zeta.
beta_init	Optional initial coefficient vector containing the intercept followed by one value per predictor in the same order as x. If omitted for a fit with zeta = 1, initialization uses beta_prior. For 0 < zeta < 1, the default blends an observed-data initializer with beta_prior. Ignored when zeta = 0.
penalty_factor	Optional nonnegative penalty multiplier for each predictor, in column order; NULL uses all ones. A zero value leaves that predictor unpenalized. The intercept is never penalized.
huber_gamma	A positive Huber smoothing parameter.
scad_a	The SCAD shape parameter; must be greater than 2 when penalty = "SCAD". Ignored when penalty = "LASSO".
max_iter	A positive whole-number limit on PGD iterations per fit (default 10000).
tol	A positive tolerance for the scaled proximal-gradient stationarity measure.
half_life	Deprecated compatibility argument. It is ignored and has no effect because the solver selects its step size by backtracking.
lambda_path	Optional complete rqPen lambda path for a fixed zeta = 0 fit. It must contain lambda; unique values are fitted in decreasing order. With the default Huber algorithm, a path containing only one unique value is extended to c(2 * lambda, lambda) for positive lambda or c(1, 0) for zero, whether supplied or omitted. A path with multiple unique values is used without adding values. Ignored by positive-weight fits. CV and HBIC automatically use their complete lambda grid for endpoint paths.
rqpen_control	Named list of rqPen endpoint options: alg (default "huber"; also "br"), scalex (default FALSE), max.iter (5000), converge.eps (1e-4), and coef.cutoff (1e-8). The chosen penalty, lambda.discard = FALSE, and supplied penalty factors are fixed by the kir package. These options control zeta = 0 fits and the rqPen baseline used to construct automatic lambda paths in tuning modes. They do not control positive-weight PGD fits or their automatic initializers. PGD max_iter, tol, half_life, and huber_gamma do not affect a zero-weight fit on a supplied path. Match the path and options when comparing to standalone rqPen.
penalty	The penalty, either "SCAD" (default) or "LASSO". Names are case-sensitive.

Details

At zeta = 0, the prior-prediction loss has zero weight, and KIQR reduces to standard penalized quantile regression on the observed responses. With all slope penalty factors equal to one, this uses the chosen penalty (SCAD by default); zero factors leave the corresponding predictors unpenalized. The intercept is always unpenalized.

For `0 < zeta < 1`, `kiqr_fit()` minimizes

$$(1-\zeta)\frac{1}{n}\sum_{i=1}^{n} h_{\tau,\gamma}(Y_i - \beta_0 - X_i\beta) + \zeta\frac{1}{n}\sum_{i=1}^{n} h_{\tau,\gamma}(\widehat{Y}_i^p - \beta_0 - X_i\beta) + (1-\zeta)\sum_{j=1}^{p} w_j p_\lambda(|\beta_j|).$$

Here $\widehat{Y}_i^p = \beta_0^p + X_i\beta^p$ are the prior predictions, p_λ is the SCAD or LASSO penalty, and $w_j = \text{penalty_factor}[j]$, with unit factors when `penalty_factor = NULL`. The intercept is not penalized. The Huber-smoothed check loss is

$$h_{\tau,\gamma}(u) = \frac{H_\gamma(u) + (2\tau-1)u}{2}$$

with `gamma = huber_gamma` and

$$H_\gamma(u) = \begin{cases} u^2/(2\gamma), & |u| \le \gamma, \\ |u| - \gamma/2, & |u| > \gamma. \end{cases}$$

For `0 < zeta < 1`, each penalty factor multiplies the penalty itself, $w_j p_\lambda(|\beta_j|)$. At `zeta = 0`, rqPen instead applies the factor inside the penalty, $p_{w_j\lambda}(|\beta_j|)$. These factor conventions agree for LASSO and for factors equal to zero or one. At `zeta = 1`, the loss uses only the prior predictions and there is no penalty.

If `beta_init = NULL` and `zeta = 1`, the fit starts from `beta_prior` without fitting an observed-data initializer. For `0 < zeta < 1`, observed-data initialization uses unpenalized quantile regression for fewer than three predictors; with at least three predictors it uses an `rqPen` path with the chosen penalty selected by HBIC. The starting vector is `zeta * beta_prior + (1 - zeta) * beta_obs`, where `beta_obs` is that observed-data initializer. A supplied `beta_init` overrides this default for positive-weight fits. The underlying quantile-regression fit can warn when its solution is nonunique; the warning is not suppressed. This can occur, for example, in intercept-only median regression with an even sample size. It does not by itself imply failure of the final PGD fit. Check `convergence` for the final fit's diagnostics. See `kiqr()` for warnings during automatic tuning.

Value

An object of class `kiqr_fit` containing named coefficients, the supplied tuning values, quantile, predictor names, call, and solver convergence diagnostics.

Zero-weight implementation

The penalized quantile regression fit at `zeta = 0` is computed using rqPen with the chosen penalty on the supplied lambda path and penalty factors. Initial coefficients and the prior predictions are ignored; the PGD solver is not used. At least one slope penalty factor must be positive. Coefficients match standalone rqPen fits with identical data, path, shape parameter, controls, and dependency versions. Diagnostics retain `engine = "rqPen"`, `successful = TRUE`, and `converged = NA`, because this fitting engine does not report the PGD stationarity criterion. Warnings are propagated.

Positive-weight fits use a Huber-smoothed loss, so the implementation does not guarantee numerical continuity as positive `zeta` approaches zero.

The rqPen software is described in Sherwood, Li and Maidman (2025). With default controls, automatic lambda construction uses rqPen's Huber-approximated LASSO routines, associated with Yi and Huang (2017). See kir-package for the full references.

Examples

```
set.seed(42)
n <- 41
x <- cbind(age = rnorm(n), marker = rnorm(n))
y <- drop(cbind(1, x) %*% c(1, 1, -0.5) + rnorm(n, sd = 0.2))
beta_prior <- c(0.5, 0.7, 0)
fit <- kiqr_fit(
  x, y, beta_prior, tau = 0.5, lambda = 0.01, zeta = 0.25,
  beta_init = beta_prior, huber_gamma = 0.1
)
coef(fit)
```

---

kiqr_hbic *Tune a KIQR Model Using HBIC*

Description

`kiqr_hbic()` evaluates supplied grids of `lambda` and `zeta` using a high-dimensional Bayesian information criterion (HBIC) and returns the selected model. HBIC requires at least three predictors (excluding the intercept), at least two observations, and finite, strictly positive total check loss for every scored candidate; use `kiqr_cv()` otherwise.

Usage

```
kiqr_hbic(
  x,
  y,
  beta_prior,
  tau,
  lambda,
  zeta,
  beta_init = NULL,
  penalty_factor = NULL,
  cores = 1,
  huber_gamma = 0.01,
  half_life = 100,
  max_iter = 10000,
  rqpen_control = list(),
  penalty = c("SCAD", "LASSO")
)
```

Arguments

x	A nonempty numeric predictor matrix or data frame with finite, non-missing values and no intercept column. Column names, if supplied, must be unique. Predictors are not automatically standardized.
y	A finite numeric response vector with one value per row of x.
beta_prior	A numeric prior coefficient vector containing the intercept followed by one finite coefficient per column of x, in exactly the same feature order, coding, and units. It must be for the same quantile level as the fit. Coefficient names do not trigger automatic reordering.
tau	A single quantile level strictly between 0 and 1.
lambda	One or more nonnegative penalty tuning values.
zeta	One or more knowledge weights in [0, 1]; see kiqr_fit() for their interpretation.
beta_init	Optional initial coefficient vector containing the intercept followed by one value per predictor in the same order as x. If omitted for a fit with zeta = 1, initialization uses beta_prior. For 0 < zeta < 1, the default blends an observed-data initializer with beta_prior. Ignored when zeta = 0.
penalty_factor	Optional nonnegative penalty multiplier for each predictor, in column order; NULL uses all ones. A zero value leaves that predictor unpenalized. The intercept is never penalized.
cores	A positive whole number of worker processes used for the grid search. Use cores = 1 on Windows, where fork-based parallelism is unavailable.
huber_gamma	A positive Huber smoothing parameter.
half_life	Deprecated compatibility argument. It is ignored and has no effect because the solver selects its step size by backtracking.
max_iter	A positive whole-number limit on PGD iterations per fit (default 10000).
rqpen_control	Named list of rqPen endpoint options: alg (default "huber"; also "br"), scalex (default FALSE), max.iter (5000), converge.eps (1e-4), and coef.cutoff (1e-8). The chosen penalty, lambda.discard = FALSE, and supplied penalty factors are fixed by the kir package. These options control zeta = 0 fits and the rqPen baseline used to construct automatic lambda paths in tuning modes. They do not control positive-weight PGD fits or their automatic initializers. PGD max_iter, tol, half_life, and huber_gamma do not affect a zero-weight fit on a supplied path. Match the path and options when comparing to standalone rqPen.
penalty	The penalty, either "SCAD" (default) or "LASSO". Names are case-sensitive.

Details

Both lambda and zeta must be supplied. Use kiqr() with method = "hbic" for automatic grids. The selected candidate minimizes log(sum(u * (tau - (u < 0)))) + s * log(log(p)) * log(n) / n, where u = y - fitted, n is the number of observations, p is the number of predictors, and s counts nonzero slopes. The intercept is excluded from p and s. Positive-weight candidates must satisfy the PGD convergence criterion; zero-weight candidates require successful fits with finite coefficients.

Nonconverged PGD candidates are excluded with a warning. A nonpositive total check loss in any scored candidate causes an error. The returned `tuning` component holds the candidate grids, rather than the full table of candidate scores. Default initialization can emit repeated nonunique-solution warnings; see `kiqr()` for their interpretation.

Value

An object of class `kiqr_fit` containing the HBIC-selected model, candidate tuning grids, and selected-fit convergence diagnostics.

Examples

```
set.seed(42)
n <- 41
# HBIC requires at least three predictors.
x <- cbind(age = rnorm(n), marker = rnorm(n), noise = seq_len(n) / n)
y <- drop(cbind(1, x) %*% c(1, 1, -0.5, 0) + rnorm(n, sd = 0.2))
# Prior coefficients include the intercept, then slopes in column order.
beta_prior <- c(0.5, 0.7, 0, 0)

fit <- kiqr_hbic(
  x, y, beta_prior, tau = 0.5,
  lambda = c(0.01, 0.02), zeta = c(0, 0.5),
  beta_init = beta_prior, huber_gamma = 0.1, max_iter = 3000
)
coef(fit)
summary(fit)
predict(fit, newdata = x[1:2, , drop = FALSE])
```

---

kir *Fit or Tune a Knowledge Integration Regression Model*

Description

`kir()` is the mean-regression counterpart of `kiqr()`. It fits penalized least-squares models that combine observed responses with predictions from supplied prior coefficients, using cross-validation (CV), a high-dimensional Bayesian information criterion (HBIC), or a fixed pair of tuning values. It delegates estimation to `kir_cv()`, `kir_hbic()`, or `kir_fit()`, respectively. With `penalty = "LASSO"` this is the prior LASSO (pLASSO) of Jiang, He and Zhang (2016); the default `penalty = "SCAD"` gives the corresponding prior SCAD (pSCAD) estimator.

Usage

```
kir(
  x,
  y,
  beta_prior,
  lambda = NULL,
  zeta = NULL,
```

```
  method = c("cv", "hbic", "fit"),
  ...
)
```

Arguments

x	A nonempty numeric predictor matrix or data frame with finite, non-missing values and no intercept column. Column names, if supplied, must be unique.
y	A finite numeric response vector with one value per row of x.
beta_prior	A numeric prior coefficient vector containing the intercept followed by one finite coefficient per column of x, in exactly the same feature order, coding, and units. Coefficient names do not trigger automatic reordering.
lambda	Optional finite, nonnegative penalty tuning value or candidate vector. In a tuning mode, omitted or NULL values generate an automatic path.
zeta	Optional knowledge weight or candidate vector in [0, 1]. In a tuning mode, omitted or NULL values use seq(0, 1, by = 0.1).
method	Selection method: "cv", "hbic", or "fit". If omitted, use "fit" when both lambda and zeta have length one; otherwise use "cv". An explicitly supplied method always takes precedence.
...	Named additional arguments supported by the selected method: penalty, penalty_factor, scad_a, max_iter, tol, nlambda, and lambda_min_ratio for every method; cores for CV and HBIC; folds, foldid, and beta_prior_fold_fun for CV; lambda_path for a fixed fit. Unsupported, unnamed, or duplicated additional arguments produce an error.

Details

Let the squared-error loss be

$$L(\beta; X, Y) = \frac{1}{2n}\|Y - \beta_0 - X\beta\|^2$$

and let $\widehat{Y}^p = \beta_0^p + X\beta^p$ be the predictions from the prior coefficients beta_prior. For 0 <= zeta < 1 and unit penalty factors, kir() minimizes

$$L_{\lambda,\zeta}(\beta; X, Y, \widehat{Y}^p) = (1-\zeta)L(\beta; X, Y) + \zeta L(\beta; X, \widehat{Y}^p) + \sum_{j=1}^{p} p_\lambda(|\beta_j|)$$

where p_λ is the SCAD or LASSO penalty. The criterion is computed on the standardized predictors used by ncvreg. With penalty factors $w_j = \text{penalty_factor}[j]$, the penalty term is

$$\sum_{j=1}^{p} p_{w_j\lambda}(|\beta_j|)$$

so each factor multiplies lambda inside the penalty, and a zero factor leaves that predictor unpenalized. For SCAD this differs from multiplying the entire penalty by w_j; for LASSO the two are equal.

With unit penalty factors, the LASSO penalty, and `zeta = eta / (1 + eta)`, this is the pLASSO criterion of Jiang, He and Zhang (2016) divided by `1 + eta`, with `lambda = lambda_pLASSO / (1 + eta)`:

$$\frac{L(\beta; X, Y) + \eta L(\beta; X, \widehat{Y}^p) + \lambda_{\mathrm{pLASSO}} \sum_{j=1}^{p} |\beta_j|}{1 + \eta}$$

At `zeta = 0`, the fit is ordinary penalized least squares on the observed responses. At `zeta = 1`, `kir()` returns `beta_prior` without a penalty, and `lambda` is not used.

The weighted loss `(1 - zeta) L(beta; X, Y) + zeta L(beta; X, Yhat^p)` equals `L(beta; X, Ytilde)` plus a constant independent of the fitted coefficients, where `Ytilde = (1 - zeta) Y + zeta Yhat^p`. Thus the fit is computed as penalized least squares on this pseudo-response.

Unlike `kiqr()`, whose objective multiplies the penalty by `1 - zeta`, `kir()` applies the standard penalty to the pseudo-response problem. For LASSO the two conventions differ only by a rescaling of `lambda` for each `zeta`; for SCAD, `(1 - zeta)` times a SCAD penalty is not itself a SCAD penalty, so the conventions define different estimators.

Value

A `kir_cv` object for CV, or a `kir_fit` object for HBIC or a fixed fit. The `coef()`, `predict()`, `summary()`, and `print()` methods apply. The recorded call is the user's call to `kir()`.

Solver and standardization

All fits with `zeta < 1` use `ncvreg::ncvreg()` (Breheny and Huang, 2011) on the pseudo-response with a decreasing lambda path and warm starts. ncvreg always standardizes the columns of `x` internally (centered and scaled to unit mean square) and penalizes the standardized coefficients; coefficients are returned on the original scale. Consequently `lambda` is on the standardized scale, unlike in `kiqr()`, which does not standardize. Columns that are constant (in the data or in a CV training fold) receive a zero slope. The intercept is never penalized.

References

Jiang, Y., He, Y., and Zhang, H. (2016). Variable Selection With Prior Information for Generalized Linear Models via the Prior LASSO Method. Journal of the American Statistical Association, 111(513), 355–376. doi:10.1080/01621459.2015.1008363.

Breheny, P., and Huang, J. (2011). Coordinate Descent Algorithms for Nonconvex Penalized Regression, with Applications to Biological Feature Selection. The Annals of Applied Statistics, 5(1), 232–253. doi:10.1214/10AOAS388.

See Also

`kir_cv()`, `kir_hbic()`, `kir_fit()`, `kiqr()`

Examples

```
set.seed(42)
n <- 60
x <- matrix(rnorm(n * 6), n, 6, dimnames = list(NULL, paste0("v", 1:6)))
y <- drop(1 + x[, 1] - 0.5 * x[, 2] + rnorm(n, sd = 0.5))
```

```
# Prior coefficients: intercept first, then one per column of x.
beta_prior <- c(0.8, 0.9, -0.4, 0, 0, 0, 0)

# pSCAD with cross-validated lambda and zeta (default).
cv <- kir(x, y, beta_prior, foldid = rep(1:5, length.out = n))
cv
coef(cv, rule = "1se")

# pLASSO at fixed tuning values.
fit <- kir(x, y, beta_prior, lambda = 0.05, zeta = 0.5, penalty = "LASSO")
summary(fit)
predict(fit, newdata = x[1:2, ])
```

---

kir-methods *Methods for Knowledge Integration Regression Fits*

Description

Extract coefficients, predict the conditional mean, summarize, and print fits from `kir()`, `kir_fit()`, `kir_cv()`, and `kir_hbic()`. For CV objects, choose between the minimum-loss (`"min"`, default) and one-standard-error (`"1se"`) fits. At `zeta = 1` the penalty is not used, so lambda is shown as "not used".

Usage

```
## S3 method for class 'kir_fit'
coef(object, ...)

## S3 method for class 'kir_cv'
coef(object, rule = c("min", "1se"), ...)

## S3 method for class 'kir_fit'
predict(object, newdata, ...)

## S3 method for class 'kir_cv'
predict(object, newdata, rule = c("min", "1se"), ...)

## S3 method for class 'kir_fit'
summary(object, threshold = 0, ...)

## S3 method for class 'kir_cv'
summary(object, rule = c("min", "1se"), threshold = 0, ...)

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

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

```
## S3 method for class 'summary_kir'
print(
  x,
  max_print = getOption("kir.max_print", 30),
  show = c("auto", "all", "nonzero"),
  digits = NULL,
  ...
)
```

Arguments

`object`, `x`	A fitted or summarized Knowledge Integration Regression object.
`...`	Additional arguments, currently unused.
`rule`	For a cross-validated model, the minimum-loss (`"min"`) or one-standard-error (`"1se"`) fit.
`newdata`	A numeric matrix or data frame with one column per predictor and no intercept column, or a numeric vector representing one observation. Named columns are reordered to the training order.
`threshold`	A finite nonnegative threshold used only to count predictor coefficients with `abs(coefficient) > threshold`.
`max_print`, `show`, `digits`	Coefficient display options; see `print.summary_kiqr()`.

Value

`coef()` returns a named coefficient vector, `predict()` a numeric vector of predicted means, `summary()` an object of class `summary_kir`, and `print()` its argument, invisibly.

`kir_cv` *Tune a Knowledge Integration Regression Model Using Cross-Validation*

Description

`kir_cv()` evaluates a grid of `lambda` and `zeta` values by k-fold cross-validation of the mean squared prediction error for the observed responses. Both the minimum-loss fit and the one-standard-error fit are retained.

Usage

```
kir_cv(
  x,
  y,
  beta_prior,
  lambda = NULL,
```

```
  zeta = NULL,
  folds = 5,
  foldid = NULL,
  penalty_factor = NULL,
  cores = 1,
  beta_prior_fold_fun = NULL,
  penalty = c("SCAD", "LASSO"),
  scad_a = 3.7,
  nlambda = 100,
  lambda_min_ratio = NULL,
  max_iter = 10000,
  tol = 1e-04
)
```

Arguments

x	A nonempty numeric predictor matrix or data frame with finite, non-missing values and no intercept column. Column names, if supplied, must be unique.
y	A finite numeric response vector with one value per row of x.
beta_prior	A numeric prior coefficient vector containing the intercept followed by one finite coefficient per column of x, in exactly the same feature order, coding, and units. Coefficient names do not trigger automatic reordering.
lambda	Optional numeric vector of nonnegative penalty values. If omitted or NULL, an automatic path is built from the observed data.
zeta	Optional knowledge weights in [0, 1]. If omitted or NULL, use seq(0, 1, by = 0.1).
folds	A positive whole number of cross-validation folds, between 2 and the number of observations. Ignored when foldid is supplied.
foldid	Optional positive whole-number vector assigning each observation to a fold. IDs must be consecutive starting at 1. When supplied, the number of unique IDs determines the number of folds.
penalty_factor	Optional nonnegative factor for each predictor, in column order; each factor multiplies lambda for that predictor inside the penalty (see kir()). NULL uses all ones. A zero value leaves that predictor unpenalized. At least one value must be positive. The intercept is never penalized.
cores	A positive whole number of worker processes. Use cores = 1 on Windows, where fork-based parallelism is unavailable. Results do not depend on cores.
beta_prior_fold_fun	Optional function with arguments x and y that re-estimates the prior coefficients within each training fold. It must return the intercept followed by one finite coefficient per predictor. It is called once per fold, and its result is shared by all positive-weight candidates on that fold. It is not called for a grid containing only zero weights, and it does not replace beta_prior for the final full-data refits.
penalty	The penalty, either "SCAD" (default, giving the pSCAD estimator) or "LASSO" (giving the pLASSO estimator). Names are case-sensitive.

`scad_a` The SCAD shape parameter; must be greater than 2 when `penalty = "SCAD"`. Ignored when `penalty = "LASSO"`.

`nlambda` Number of values in the automatic lambda path, at least 2; default 100. Ignored when a lambda path or grid is supplied.

`lambda_min_ratio` Optional ratio of the smallest to the largest automatic lambda, strictly between 0 and 1. Defaults to 0.001 when the number of observations exceeds the number of nonconstant predictors and 0.05 otherwise, as in ncvreg.

`max_iter` A positive whole-number limit on the total number of coordinate-descent iterations over the whole lambda path (ncvreg's `max.iter`; default 10000).

`tol` A positive convergence tolerance (ncvreg's `eps`).

Details

Every candidate is evaluated on the same folds. For each fold and each `zeta < 1`, one ncvreg path over the whole lambda grid is fitted to the training pseudo-response built from the training-fold prior; the validation loss is the mean squared error against the observed validation responses (not the pseudo-response). At `zeta = 1` the validation loss uses the (fold-specific) prior coefficients and is shared across lambda values.

The minimum rule selects the eligible candidate with the smallest mean validation loss. The 1-SE rule fixes `zeta` at the minimum-rule value and selects the largest eligible `lambda` at that `zeta` whose mean loss is within one standard error of the minimum, as in `kiqr_cv()`. The standard error is the sample standard deviation of the fold losses divided by the square root of the number of folds. Candidates that do not complete every fold (for example because ncvreg reached `max_iter`) are excluded with a warning.

The automatic lambda path is built from the observed data only, as in ncvreg's default path for `(x, y)`. By default it contains 100 values, logarithmically spaced from `lambda_max` down to `lambda_min_ratio * lambda_max`. On the standardized predictor scale, `lambda_max` is the smallest lambda giving zero penalized slopes at `zeta = 0`. The default `lambda_min_ratio` is 0.001 when the number of observations exceeds the number of nonconstant predictors and 0.05 otherwise. `nlambda` and `lambda_min_ratio` control the path. A degenerate path contains only zero when all penalized gradients are zero. The same path is used for every `zeta` and every fold, so the `zeta = 0` candidates equal penalized least squares without prior information on the same folds. The default zeta grid is `seq(0, 1, by = 0.1)`. If the prior is estimated using the observed responses, use `beta_prior_fold_fun` so that validation responses do not enter the training-fold prior.

Value

An object of class `kir_cv`. Use `coef()`, `predict()`, or `summary()` with `rule = "min"` or `rule = "1se"`. The object includes both coefficient vectors and tuning pairs, the candidate grid with `cv_mean` and `cv_se`, the fold-loss matrix, fold assignments, per-fold diagnostics, and `tuning` metadata.

See Also

`kir()`

Examples

```
set.seed(42)
n <- 60
x <- matrix(rnorm(n * 6), n, 6)
y <- drop(1 + x[, 1] - 0.5 * x[, 2] + rnorm(n, sd = 0.5))
beta_prior <- c(0.8, 0.9, -0.4, 0, 0, 0, 0)
cv <- kir_cv(x, y, beta_prior, zeta = c(0, 0.5, 1),
             foldid = rep(1:5, length.out = n))
summary(cv, rule = "1se")
```

---

kir_fit *Fit a Knowledge Integration Regression Model*

Description

`kir_fit()` fits one Knowledge Integration Regression (pSCAD or pLASSO) model at a specified pair of `lambda` and `zeta` values. See `kir()` for the model.

Usage

```
kir_fit(
  x,
  y,
  beta_prior,
  lambda,
  zeta,
  penalty_factor = NULL,
  penalty = c("SCAD", "LASSO"),
  scad_a = 3.7,
  lambda_path = NULL,
  nlambda = 100,
  lambda_min_ratio = NULL,
  max_iter = 10000,
  tol = 1e-04
)
```

Arguments

`x`	A nonempty numeric predictor matrix or data frame with finite, non-missing values and no intercept column. Column names, if supplied, must be unique.
`y`	A finite numeric response vector with one value per row of `x`.
`beta_prior`	A numeric prior coefficient vector containing the intercept followed by one finite coefficient per column of `x`, in exactly the same feature order, coding, and units. Coefficient names do not trigger automatic reordering.
`lambda`	A single nonnegative penalty tuning parameter, on the scale of standardized predictors (see `kir()`).

`zeta` A single knowledge weight in `[0, 1]`. At zero, the fit uses only the observed responses; at one, it returns `beta_prior`.

`penalty_factor` Optional nonnegative factor for each predictor, in column order; each factor multiplies `lambda` for that predictor inside the penalty (see `kir()`). `NULL` uses all ones. A zero value leaves that predictor unpenalized. At least one value must be positive. The intercept is never penalized.

`penalty` The penalty, either `"SCAD"` (default, giving the pSCAD estimator) or `"LASSO"` (giving the pLASSO estimator). Names are case-sensitive.

`scad_a` The SCAD shape parameter; must be greater than 2 when `penalty = "SCAD"`. Ignored when `penalty = "LASSO"`.

`lambda_path` Optional lambda path along which the fit is computed with warm starts. Unique values are fitted in decreasing order and the path must contain `lambda`. If `NULL`, the automatic observed-data path (see `kir_cv()`) with `lambda` added is used, so a fit at a lambda selected by `kir_cv()` with an automatic path reproduces its refit. A path with a single value `lambda` is extended to `c(2 * lambda, lambda)` (or `c(1, 0)` for zero). With SCAD the solution can depend on the path.

`nlambda` Number of values in the automatic lambda path, at least 2; default 100. Ignored when a lambda path or grid is supplied.

`lambda_min_ratio` Optional ratio of the smallest to the largest automatic lambda, strictly between 0 and 1. Defaults to 0.001 when the number of observations exceeds the number of nonconstant predictors and 0.05 otherwise, as in ncvreg.

`max_iter` A positive whole-number limit on the total number of coordinate-descent iterations over the whole lambda path (ncvreg's `max.iter`; default 10000).

`tol` A positive convergence tolerance (ncvreg's `eps`).

Details

The fit is computed by ncvreg on the pseudo-response `(1 - zeta) * y + zeta * cbind(1, x) %*% beta_prior`; see `kir()` for the criterion and the standardization performed by ncvreg. An error is raised if ncvreg reaches `max_iter` before reaching `lambda`.

Value

An object of class `kir_fit` containing named coefficients, the supplied tuning values, predictor names, call, and solver diagnostics.

See Also

`kir()`

Examples

```
set.seed(42)
n <- 60
x <- matrix(rnorm(n * 6), n, 6)
y <- drop(1 + x[, 1] - 0.5 * x[, 2] + rnorm(n, sd = 0.5))
```

```
beta_prior <- c(0.8, 0.9, -0.4, 0, 0, 0, 0)
fit <- kir_fit(x, y, beta_prior, lambda = 0.05, zeta = 0.5)
coef(fit)
```

---

kir_hbic *Tune a Knowledge Integration Regression Model Using HBIC*

Description

kir_hbic() evaluates grids of lambda and zeta values with a high-dimensional Bayesian information criterion (HBIC) and returns the selected model. The criterion has the same form as in kiqr_hbic(), with the residual sum of squares in place of the total check loss.

Usage

```
kir_hbic(
  x,
  y,
  beta_prior,
  lambda = NULL,
  zeta = NULL,
  penalty_factor = NULL,
  cores = 1,
  penalty = c("SCAD", "LASSO"),
  scad_a = 3.7,
  nlambda = 100,
  lambda_min_ratio = NULL,
  max_iter = 10000,
  tol = 1e-04
)
```

Arguments

x	A nonempty numeric predictor matrix or data frame with finite, non-missing values and no intercept column. Column names, if supplied, must be unique.
y	A finite numeric response vector with one value per row of x.
beta_prior	A numeric prior coefficient vector containing the intercept followed by one finite coefficient per column of x, in exactly the same feature order, coding, and units. Coefficient names do not trigger automatic reordering.
lambda	Optional numeric vector of nonnegative penalty values. If omitted or NULL, an automatic path is built from the observed data.
zeta	Optional knowledge weights in [0, 1]. If omitted or NULL, use seq(0, 1, by = 0.1).
penalty_factor	Optional nonnegative factor for each predictor, in column order; each factor multiplies lambda for that predictor inside the penalty (see kir()). NULL uses all ones. A zero value leaves that predictor unpenalized. At least one value must be positive. The intercept is never penalized.

`cores` A positive whole number of worker processes. Use `cores = 1` on Windows, where fork-based parallelism is unavailable. Results do not depend on `cores`.

`penalty` The penalty, either `"SCAD"` (default, giving the pSCAD estimator) or `"LASSO"` (giving the pLASSO estimator). Names are case-sensitive.

`scad_a` The SCAD shape parameter; must be greater than 2 when `penalty = "SCAD"`. Ignored when `penalty = "LASSO"`.

`nlambda` Number of values in the automatic lambda path, at least 2; default 100. Ignored when a lambda path or grid is supplied.

`lambda_min_ratio` Optional ratio of the smallest to the largest automatic lambda, strictly between 0 and 1. Defaults to 0.001 when the number of observations exceeds the number of nonconstant predictors and 0.05 otherwise, as in ncvreg.

`max_iter` A positive whole-number limit on the total number of coordinate-descent iterations over the whole lambda path (ncvreg's `max.iter`; default 10000).

`tol` A positive convergence tolerance (ncvreg's `eps`).

Details

The selected candidate minimizes `log(RSS) + s * log(log(p)) * log(n) / n`, where `RSS` is the residual sum of squares for the observed responses (whatever the value of `zeta`), `n` the number of observations, `p` the number of predictors, and `s` the number of nonzero slopes (the intercept is excluded from `p` and `s`). As in `kiqr_hbic()`, HBIC requires at least three predictors, because `log(log(p))` is not positive for `p < 3`, and a positive `RSS` for every scored candidate. Each `zeta < 1` uses one full-data ncvreg path over the lambda grid; omitted grids are built as in `kir_cv()`.

Value

An object of class `kir_fit` containing the HBIC-selected model, the candidate grid with scores in `tuning$scores`, and solver diagnostics.

See Also

`kir()`

Examples

```
set.seed(42)
n <- 60
x <- matrix(rnorm(n * 6), n, 6)
y <- drop(1 + x[, 1] - 0.5 * x[, 2] + rnorm(n, sd = 0.5))
beta_prior <- c(0.8, 0.9, -0.4, 0, 0, 0, 0)
fit <- kir_hbic(x, y, beta_prior, zeta = c(0, 0.5))
fit
```

predict.kiqr_fit *Predict from a KIQR Model*

Description

Compute linear predictions of the fitted conditional quantile at new predictor values. For CV objects, prediction defaults to the minimum-loss fit.

Usage

```
## S3 method for class 'kiqr_fit'
predict(object, newdata, ...)

## S3 method for class 'kiqr_cv'
predict(object, newdata, rule = c("min", "1se"), ...)
```

Arguments

object	A fitted KIQR object.
newdata	A numeric matrix or data frame with one column per predictor and no intercept column, or a numeric vector representing one observation.
...	Additional arguments, currently unused.
rule	For a cross-validated model, use the minimum-loss ("min") or one-standard-error ("1se") fit.

Details

If newdata has column names, columns are reordered to the training feature order. An unnamed matrix or a vector must already use that order; names on a vector are ignored. Use the same predictor coding and units as in training. newdata must not contain an intercept column.

Value

A numeric vector of predicted conditional quantiles.

print.kiqr_fit *Print a KIQR Model*

Description

Display tuning values and solver diagnostics. CV output shows both selection rules and the diagnostics of their full-data refits. Zero-weight fits have no reported convergence criterion or iteration count. At zeta = 1 the penalty is not used, so lambda is shown as "not used". Predictor counts exclude the intercept; summaries display the threshold used for counting. The penalty is always shown. Summary printing can omit zero coefficients or abbreviate long coefficient vectors; the complete vector remains in the summary object.

Usage

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

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

## S3 method for class 'summary_kiqr'
print(
  x,
  max_print = getOption("kir.max_print", 30),
  show = c("auto", "all", "nonzero"),
  digits = NULL,
  ...
)
```

Arguments

x	A fitted or summarized KIQR object.
...	Additional arguments, currently unused.
max_print	Maximum number of coefficients to print, including the intercept. A positive whole number or Inf. Defaults to the kir.max_print option, or 30 if that option is unset. When abbreviated, the first ceiling(max_print / 2) and the last remaining coefficients are shown with ... between them.
show	Coefficients to consider: "all" includes zeros; "nonzero" includes the intercept and predictors with absolute coefficients greater than the summary's threshold. The default, "auto", uses "all" if the complete vector fits within max_print, otherwise "nonzero".
digits	Number of significant digits, passed to format() for an abbreviated vector and to print() otherwise. NULL uses R's default.

Value

x, invisibly.

---

summary.kiqr_fit *Summarize a KIQR Model*

Description

Return coefficients, tuning values, predictor counts, and solver diagnostics. For CV objects, the default is the minimum-loss fit. No standard errors, confidence intervals, or hypothesis tests are computed.

Usage

```
## S3 method for class 'kiqr_fit'
summary(object, threshold = 0, ...)

## S3 method for class 'kiqr_cv'
summary(object, rule = c("min", "1se"), threshold = 0, ...)
```

Arguments

object	A fitted KIQR object.
threshold	A finite nonnegative threshold used only to count predictor coefficients satisfying abs(coefficient) > threshold. The intercept is excluded. This does not modify coefficients or predictions.
...	Additional arguments, currently unused.
rule	For CV objects, selects the fit to summarize. Use "min" for the minimum-loss fit or "1se" for the one-standard-error fit.

Value

An object of class summary_kiqr.

Index

base::print(), *2*
base::summary(), *2*

coef.kiqr_cv *(*coef.kiqr_fit*)*, 3
coef.kiqr_fit, 3
coef.kir_cv *(*kir-methods*)*, 18
coef.kir_fit *(*kir-methods*)*, 18

kiqr, 4
kiqr(), *2, 3*, *12*, *14, 15*, *17*
kiqr_cv, 6
kiqr_cv(), *2–6*, *13*, *21*
kiqr_fit, 10
kiqr_fit(), *2–7*, *14*
kiqr_hbic, 13
kiqr_hbic(), *2–6*, *24, 25*
kir, 15
kir(), *2, 3*, *6*, *18*, *20–25*
kir-methods, 18
kir-package, 2, *13*
kir_cv, 19
kir_cv(), *2, 3*, *15*, *17, 18*, *23*, *25*
kir_fit, 22
kir_fit(), *2, 3*, *15*, *17, 18*
kir_hbic, 24
kir_hbic(), *2, 3*, *15*, *17, 18*

ncvreg::ncvreg(), *17*

predict.kiqr_cv *(*predict.kiqr_fit*)*, 26
predict.kiqr_fit, 26
predict.kir_cv *(*kir-methods*)*, 18
predict.kir_fit *(*kir-methods*)*, 18
print.kiqr_cv *(*print.kiqr_fit*)*, 26
print.kiqr_fit, 26
print.kir_cv *(*kir-methods*)*, 18
print.kir_fit *(*kir-methods*)*, 18
print.summary_kiqr *(*print.kiqr_fit*)*, 26
print.summary_kiqr(), *19*
print.summary_kir *(*kir-methods*)*, 18

stats::coef(), *2*
stats::predict(), *2*
summary.kiqr_cv *(*summary.kiqr_fit*)*, 27
summary.kiqr_fit, 27
summary.kir_cv *(*kir-methods*)*, 18
summary.kir_fit *(*kir-methods*)*, 18