Package 'raiseR'

September 15, 2026

Type Package

Title Raise Regression and Robust Methods for Multicollinearity

Version 0.1.0

Description Implements Raise Regression as an inference-preserving alternative to Ridge Regression for combating multicollinearity in linear models, including the classical single-variable Raise Regression, the Simultaneous Raise Regression (SRR) based on QR decomposition and the Sequential Variance Inflation Factor (SVIF) of Jacob and Varadharajan (2022) <doi:10.1007/s11135-022-01557-9>, and the original raise parameter selection strategy of Jacob and Varadharajan (2023) <doi:10.13189/ms.2023.110106>. Also implements Robust Raise Regression for data contaminated by outliers, with exact finite-sample inference (sandwich standard errors, Wald tests, Satterthwaite-corrected degrees of freedom) obtained by down-weighting observations using Stahel-Donoho projection outlyingness and Tukey's biweight function. Provides ordinary and robust Ridge Regression (Hoerl and Kennard, 1970, <doi:10.1080/00401706.1970.10488634>), ordinary and robust Liu Regression (Liu, 1993, <doi:10.1080/03610929308831027>), with the robust variants of both based on the MM-estimates of Yohai (1987, <doi:10.1214/aos/1176350366>) and, for Liu Regression specifically, the biasing-parameter derivation of Filzmoser and Kurnaz (2018) <doi:10.1080/03610918.2016.1271889>. Also provides the classical Variance Inflation Factor (VIF) and Condition Number (Belsley, 1991) computed from the correlation matrix of the predictors, and the Robust Variance Inflation Factor (RVIF) and robust Condition Number of Jacob and Varadharajan (2024, Sankhya B, <doi:10.1007/s13571-024-00342-y>), which use the same projection outlyingness and biweight down-weighting scheme to obtain a weighted correlation matrix that resists the influence of outliers. A flexible scaleDat() function supports classical (mean and standard deviation), robust weighted (Stahel-Donoho and Tukey biweight), median and Median Absolute Deviation Normalized (MADN, the median absolute deviation scaled by 1.4826 to estimate the standard deviation under normality), and min-max scaling. Diagnostic and goodness-of-fit plots, and the standard influence-diagnostic suite (Cook's distance, DFBETAS and COVRATIO regression diagnostics) and heteroskedasticity tests (via the 'lmtest' and 'car' packages) analogous to those for objects of class 'lm', are provided for the exact, unbiased Raise Regression fit.

License GPL (>= 3)

Encoding UTF-8

RoxygenNote 7.3.1

Depends R (>= 3.5.0)

Imports stats, graphics, grDevices, mrfDepth, MASS, withr

Suggests testthat (>= 3.0.0), lmtest, car

URL https://github.com/jinsejacob/raiseR

BugReports https://github.com/jinsejacob/raiseR/issues

Config/testthat/edition 3

Author Jinse Jacob [aut, cre] (ORCID: <https://orcid.org/0000-0002-0873-4864>)

Maintainer Jinse Jacob <jinsejacob@hotmail.com>

NeedsCompilation no

Repository CRAN

Date/Publication 2026-09-15 10:50:07 UTC

Contents

cn *Condition Number and Condition Indices*

Description

Computes the overall Condition Number (Belsley, 1991) together with the full vector of Condition Indices, one per principal axis of the (correlation-matrix) eigenstructure of the predictors – either in the ordinary sense (`type = "O"`) or a robust, outlier-resistant sense (`type = "R"`).

Usage

```
cn(formula, data, type = c("O", "R"), cc = 4.685, seed = NULL)
```

Arguments

`formula`	a two-sided formula.
`data`	a data frame.
`type`	`"O"` (ordinary, default) or `"R"` (robust/weighted).
`cc`	tuning constant for Tukey's biweight (default 4.685). Only used when `type = "R"`.
`seed`	optional integer for reproducible direction sampling. Only used when `type = "R"`.

Details

For `type = "R"`, weights are computed from Tukey's biweight function applied to each observation's Stahel-Donoho projection outlyingness – the same scheme used throughout this package's robust methods (`rvif()`, `robRaise()`, `robRidge()`, `robLiu()`) – with one important difference: outlyingness here is computed jointly over the predictor *and* response space (i.e. from `cbind(X, y)`, not `X` alone), following Jacob and Varadharajan (2024). This is because a point that is only an outlier in the direction of y (a pure vertical outlier) can distort the fitted model just as much as an outlier in the predictor space, and the resulting weights should downweight both kinds – the weighted correlation matrix used for the Condition Number itself is then formed from the predictors alone, using those joint-space weights.

Value

an object of class `"cn"`, a list with `$cn` (the Condition Number, a single number – the largest Condition Index) and `$ci` (the full named vector of Condition Indices). Both are printed by default; access either individually via `$cn` / `$ci` if only one is needed.

Examples

```
cn(mpg ~ disp + hp + wt + drat, data = mtcars)

## type = "R" requires the mrfDepth package
cn(mpg ~ disp + hp + wt + drat, data = mtcars, type = "R")
```

`liuReg` *Ordinary Liu Regression*

Description

Fits Liu Regression (Liu, 1993) on a mean-centred design:

$$\hat{\beta}_{Liu} = (X'X + I)^{-1}(X'X + dI)\hat{\beta}_{OLS}$$

where $0 < d < 1$ is the biasing parameter.

Usage

```
liuReg(formula, data, d = c("dopt", "dmm"), clip = TRUE)
```

Arguments

`formula`	a two-sided formula.
`data`	a data frame.
`d`	either `"dopt"` (default), `"dmm"`, or a fixed numeric value in principle taken in $(0, 1)$ (values outside this range are allowed, with a warning when `clip = FALSE`).
`clip`	if `TRUE` (default), an automatically-computed `d` ("dopt" or "dmm") outside $[0, 1]$ is snapped to the nearer boundary. Ignored when `d` is supplied numerically. Set to `FALSE` to inspect the raw, unclipped formula output instead.

Details

Three ways of choosing `d` are available, computed from the canonical (eigenvector) form of the OLS fit:

`"dopt"` **(default)** The exact MSE-minimizing d, derived from the canonical-form bias-variance tradeoff directly. Numerically better-behaved than `"dmm"` when predictors are close to exactly collinear (see below).

`"dmm"` The original Liu (1993) plug-in formula, $d = 1 - \hat{\sigma}^2 \sum 1/[\lambda_i(\lambda_i + 1)] \big/ \sum \hat{\alpha}_i^2/(\lambda_i + 1)^2$, where λ_i are the eigenvalues of $X'X$ and $\hat{\alpha}_i$ are the OLS coefficients rotated into the eigenvector basis. This formula divides by eigenvalues directly, so on near-exactly singular designs it can return an extreme, meaningless value; a warning is issued in that regime and `"dopt"` is suggested instead.

a numeric value Used directly as `d`, bypassing both formulas.

Both `"dmm"` and `"dopt"` are plug-in estimates built from $\hat{\sigma}^2$ and $\hat{\beta}_{OLS}$; in finite samples, especially under multicollinearity or a low signal-to-noise ratio, they can fall outside the $(0, 1)$ range the underlying theory assumes. `clip = TRUE` (the default) snaps an out-of-range estimate to the nearest boundary (0 or 1) rather than using it as computed – turn this off only if you want to inspect the raw, unclipped formula output, since an unclipped out-of-range `d` can make predictions considerably worse.

Value

an object of class "liuReg".

Examples

```
fit <- liuReg(mpg ~ disp + hp + wt + drat, data = mtcars)
summary(fit)
fit_mm <- liuReg(mpg ~ disp + hp + wt + drat, data = mtcars, d = "dmm")
fit_manual <- liuReg(mpg ~ disp + hp + wt + drat, data = mtcars, d = 0.5)
```

---

plot.liuReg *Diagnostic plots for a liuReg object*

Description

Diagnostic plots for a liuReg object

Usage

```
## S3 method for class 'liuReg'
plot(x, which = 1:2, ...)
```

Arguments

x	a liuReg object.
which	subset of 1:2.
...	unused.

Value

Invisibly returns the fitted liuReg object ‘x‘. Called for its side effect of drawing diagnostic plots.

---

plot.raiseReg *Diagnostic plots for a raiseReg object*

Description

Produces the same family of goodness-of-fit / diagnostic plots as plot.lm: residuals vs fitted, a normal Q-Q plot of the residuals, a scale-location plot, and a before/after bar chart of VIF values.

Usage

```
## S3 method for class 'raiseReg'
plot(x, which = 1:4, ...)
```

Arguments

x	a raiseReg object.
which	subset of 1:4 selecting which plots to draw.
...	passed on to the underlying plotting functions.

Value

Invisibly returns the fitted raiseReg object ‘x‘. Called for its side effect of drawing diagnostic plots.

---

plot.ridgeReg *Diagnostic plots for a ridgeReg object*

Description

Diagnostic plots for a ridgeReg object

Usage

```
## S3 method for class 'ridgeReg'
plot(x, which = 1:2, ...)
```

Arguments

x	a ridgeReg object.
which	subset of 1:2.
...	unused.

Value

Invisibly returns the fitted ridgeReg object ‘x‘. Called for its side effect of drawing diagnostic plots.

plot.robLiu *Diagnostic plots for a robLiu object*

Description

Diagnostic plots for a robLiu object

Usage

```
## S3 method for class 'robLiu'
plot(x, which = 1:2, ...)
```

Arguments

x	a robLiu object.
which	subset of 1:2.
...	unused.

Value

Invisibly returns the fitted robLiu object 'x'. Called for its side effect of drawing diagnostic plots.

plot.robRaise *Diagnostic plots for an robRaise object*

Description

Residuals-vs-fitted, a normal Q-Q plot of the residuals with points sized by their downweighting, a weight-vs-outlyingness plot, and a before/after weighted-VIF bar chart.

Usage

```
## S3 method for class 'robRaise'
plot(x, which = 1:4, ...)
```

Arguments

x	an robRaise object.
which	subset of 1:4 selecting which plots to draw.
...	passed on to the underlying plotting functions.

Value

Invisibly returns the fitted robRaise object 'x'. Called for its side effect of drawing diagnostic plots.

plot.robRidge *Diagnostic plots for an robRidge object*

Description

Diagnostic plots for an robRidge object

Usage

```
## S3 method for class 'robRidge'
plot(x, which = 1:2, ...)
```

Arguments

x	an robRidge object.
which	subset of 1:2.
...	unused.

Value

Invisibly returns the fitted robRidge object 'x'. Called for its side effect of drawing diagnostic plots.

raiseReg *Raise Regression*

Description

Fits Raise Regression, an inference-preserving alternative to Ridge Regression for combating multicollinearity. Two strategies are available:

"simultaneous" Simultaneous Raise Regression (SRR) of Jacob and Varadharajan (2022) <doi:10.1007/s11135-022-01557-9>. A QR decomposition of the (mean-centred) design matrix is used to compute the Sequential Variance Inflation Factor (SVIF) for every predictor, and every predictor at or above threshold is raised in a single step.

"sequential" The original single-variable raise strategy of Jacob and Varadharajan (2023) <doi:10.13189/ms.2023.110106> the predictor with the largest ordinary VIF is raised, VIFs are recomputed, and the process repeats until every VIF is below threshold.

In both cases, by construction, regressing y on the *raised* design $\tilde{X} = Q\tilde{R}$ exactly reproduces the ordinary least squares fitted values $QQ'y$: raising only reallocates the fitted signal among collinear predictors, it never changes the fitted subspace. The residual standard error, R^2, adjusted R^2, F-statistic and the standard errors underlying the coefficient table are therefore all computed from this raised-design residual and are numerically identical to their ordinary least squares counterparts – only the individual coefficient estimates change, becoming far more stable, so the usual t- and

F-testing machinery remains exactly valid. `fitted()`, `residuals()` and `predict()`, in contrast, apply the raise coefficients to the *original, unraised* predictors, as is required to score new data; these values are close to, but not numerically identical to, the training fit implied by the reported R^2 and residual standard error.

Usage

```
raiseReg(
  formula,
  data,
  method = c("sequential", "simultaneous"),
  threshold = 10,
  margin = 0.04,
  lambda_max = 1000,
  lambda_step = 0.1,
  lambda = NULL,
  raise_method = c("closed_form", "grid"),
  ...
)
```

Arguments

`formula`	a two-sided formula, as in `lm`.
`data`	a data frame.
`method`	`"sequential"` (default) or `"simultaneous"`.
`threshold`	the VIF/SVIF threshold above which a predictor is raised (default 10, the conventional cutoff).
`margin`	safety margin subtracted from `threshold` when solving for the raise parameter for the sequential method, so that the *post-raise* VIF is strictly, not just marginally, below the threshold (default 0.04). The simultaneous method solves its raise parameters in closed form directly against `threshold`, so `margin` does not apply to it.
`lambda_max`	raise parameters above this are treated as "drop the predictor" rather than "raise it this much" (default 1000, matching the original implementation's grid bound).
`lambda_step`	grid resolution in the raise-parameter search, only used when `raise_method = "grid"` (default 0.1); ignored by the default closed-form solver.
`lambda`	optional named numeric vector of user-supplied raise parameters for some or all predictors (the actual multiplier applied, i.e. what is reported in `$k`). Named entries are used directly instead of being computed automatically, and are never dropped regardless of `lambda_max`.
`raise_method`	`"closed_form"` (default, exact) or `"grid"` (a literal replica of the original grid search, for reproducibility with earlier published results); only affects `method = "simultaneous"`.
`...`	additional arguments, ignored. Present so that generic callers such as `stats::update()` (used internally by `car::ncvTest()`) can pass extra arguments without error.

Details

If a predictor's automatically-computed raise parameter would exceed `lambda_max`, it is dropped from the model entirely (with a message) rather than raised to that extreme a value, and everything is recomputed on the remaining predictors; a raise parameter needing to be that large signals a predictor carrying essentially no information the other predictors don't already supply. Dropped predictors get an `NA` coefficient, exactly as `lm()` does for aliased terms; see `$dropped`.

Value

an object of class `"raiseReg"`. In addition to the usual `coefficients`/`fitted.values`/`residuals`, it carries `$vif` and `$svif` (before/after matrices; SVIF is what actually drives the raising decision, ordinary VIF depends on *all* other predictors and can move even for a predictor that was not itself raised), `$cn` (before/after Condition Number), `$k` (the raise parameters actually applied), and `$dropped` (predictors excluded for exceeding `lambda_max`).

Examples

```
fit <- raiseReg(mpg ~ disp + hp + wt + drat, data = mtcars)
summary(fit)
fit$vif; fit$svif; fit$cn
fit_seq <- raiseReg(mpg ~ disp + hp + wt + drat, data = mtcars, method = "sequential")
summary(fit_seq)
```

`raiseReg-diagnostics` *Regression Diagnostics for a raiseReg Object*

Description

`raiseReg` objects support the standard influence-diagnostic suite (`hatvalues()`, `cooks.distance()`, `dfbetas()`, `covRatio()`) as well as `lmtest::bptest()` and `car::ncvTest()`. This is possible without approximation because regressing y on the raised design reproduces the ordinary least squares projection of y *exactly* (see `?raiseReg`): the hat matrix, residuals and $\hat{\sigma}^2$ underlying these diagnostics are therefore identical to those of an ordinary `lm()` fit on the surviving predictors, and every value returned here is numerically identical to fitting that `lm()` directly and calling the same generic on it.

Usage

```
## S3 method for class 'raiseReg'
hatvalues(model, ...)

## S3 method for class 'raiseReg'
cooks.distance(model, ...)

## S3 method for class 'raiseReg'
dfbetas(model, ...)
```

```
covRatio(model, ...)

## Default S3 method:
covRatio(model, ...)

## S3 method for class 'raiseReg'
covRatio(model, ...)

## S3 method for class 'raiseReg'
model.matrix(object, ...)

## S3 method for class 'raiseReg'
weights(object, ...)

## S3 method for class 'raiseReg'
df.residual(object, ...)

## S3 method for class 'raiseReg'
ncvTest(model, var.formula, ...)

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

Arguments

`model`, `object`	a `raiseReg` object.
`...`	unused (present for generic consistency).
`var.formula`	optional one-sided formula specifying the variables against which non-constant variance is tested (`ncvTest` only); defaults to the fitted values.

Details

These methods are only provided for `raiseReg` (the exact, unbiased case). `ridgeReg`, `liuReg`, and their robust counterparts are deliberately biased estimators without an idempotent hat matrix, so classical influence measures don't have an agreed-on, unambiguous definition for them, and are not provided here; the robust methods (`robRaise()`, `robRidge()`, `robLiu()`) already downweight influential/outlying observations directly via their fitting procedure.

Value

`hatvalues.raiseReg` returns a named numeric vector of hat (leverage) values, one per observation.

`cooks.distance.raiseReg` returns a named numeric vector of Cook's distances, one per observation.

`dfbetas.raiseReg` returns a numeric matrix of DFBETAS, with one row per observation and one column per model coefficient.

`covRatio` and `covRatio.raiseReg` return a named numeric vector of covariance ratios, one per observation. `covRatio` is the S3 generic; `covRatio.default` dispatches to `stats::covratio()`.

`ncvTest.raiseReg` returns an object of class `"chisqTest"` (from the **car** package): a list containing the score-test statistic `ChiSquare`, its degrees of freedom `Df`, and the p-value `p` for the test of non-constant error variance.

`ridgeReg` *Ordinary Ridge Regression*

Description

Fits Ridge Regression (Hoerl and Kennard, 1970) on a mean-centred design, with the biasing parameter chosen as $k = p\hat{\sigma}^2 / \hat{\beta}'\hat{\beta}$, where $\hat{\beta}$ and $\hat{\sigma}^2$ come from the ordinary least squares fit.

Usage

```
ridgeReg(formula, data, k = NULL)
```

Arguments

`formula`	a two-sided formula.
`data`	a data frame.
`k`	optional fixed biasing parameter; if `NULL` (default) it is estimated from the data as described above.

Value

an object of class `"ridgeReg"`.

Examples

```
fit <- ridgeReg(mpg ~ disp + hp + wt + drat, data = mtcars)
summary(fit)
```

`robLiu` *Robust Liu Regression*

Description

Fits a robust version of Liu Regression using one of two strategies, mirroring `robRidge()`.

Usage

```
robLiu(
  formula,
  data,
  type = c("MM", "SDO"),
  d = NULL,
  clip = TRUE,
  cc = 4.685,
  maxit = 200,
  acc = 1e-04,
  seed = NULL
)
```

Arguments

formula	a two-sided formula.
data	a data frame.
type	"MM" (default) or "SDO".
d	for type = "SDO": either "dopt" (default, the more numerically robust of the two – see ?liuReg), "dmm", or a fixed numeric value. For type = "MM": either NULL (default, the closed-form MSE-minimizing value described above – the MM-specific analogue of "dopt", so the "dmm"/"dopt" strings are not applicable and raise an error) or a fixed numeric value.
clip	if TRUE (default), an automatically-computed d is snapped to $[0, 1]$ if it falls outside that range.
cc	tuning constant for Tukey's biweight (default 4.685). Used by both types.
maxit, acc	passed to MASS::rlm(); only used when type = "MM".
seed	optional integer for reproducible direction sampling; only used when type = "SDO".

Details

type = "MM" **(default)** Following Filzmoser and Kurnaz's MM-Liu estimator: MASS::rlm(..., method = "MM") supplies both the plug-in coefficient vector $\hat{\beta}_{MM}$ and observation weights W (from the MM-estimator's own IRLS iterations), and

$$\hat{\beta}_{MM\text{-}Liu} = (X'WX + I)^{-1}(X'WX + dI)\hat{\beta}_{MM}.$$

The biasing parameter d minimizes $\mathrm{MSE}(d) = \mathrm{bias}(d)'\mathrm{bias}(d) + tr(\mathrm{cov}(d))$ where $\mathrm{bias}(d) = (d-1)(X'WX+I)^{-1}\hat{\beta}_{MM}$ and $\mathrm{cov}(d) = G(d)\,\hat{V}\,G(d)'$ with $G(d) = (X'WX+I)^{-1}(X'WX+dI)$ and $\hat{V}$ the (asymptotic, sandwich-type) covariance of $\hat{\beta}_{MM}$ reported by MASS::rlm(). Because $G(d)$ is affine in d, this objective is an exact quadratic in d and is minimized here in closed form (not by numerical search).

type = "SDO" Observations are weighted once, up front, by Tukey's biweight function applied to their Stahel-Donoho projection outlyingness in the joint (response, predictors) space – exactly as in robRidge(type = "SDO") – and ordinary Liu regression (liuReg()'s dopt/dmm formulas) is then applied to the resulting weighted design and response.

Value

an object of class "robLiu".

References

Filzmoser, P. and Kurnaz, F.S. A Robust Liu Regression Estimator. Yohai, V.J. (1987). High breakdown-point and high efficiency robust estimates for regression. *Annals of Statistics*.

Examples

```
fit_mm <- robLiu(mpg ~ disp + hp + wt + drat, data = mtcars)
summary(fit_mm)

fit_sdo <- robLiu(mpg ~ disp + hp + wt + drat, data = mtcars,
                  type = "SDO", seed = 1)
summary(fit_sdo)
```

---

robRaise *Robust Raise Regression (RRM)*

Description

Fits Robust Raise Regression: observations are downweighted by Tukey's biweight function applied to their Stahel-Donoho projection outlyingness in the joint (response, predictors) space, the weighted design is mean-centred using the corresponding weighted means, and every predictor whose weighted Sequential Variance Inflation Factor (SVIF) is at or above `threshold` is raised via a weighted QR decomposition – combining the Raise Regression's inference-preserving handling of collinearity with resistance to outliers in X, Y and XY space.

Usage

```
robRaise(
  formula,
  data,
  method = "simultaneous",
  cc = 4.685,
  seed = NULL,
  threshold = 10,
  margin = 0.04,
  lambda_max = 1000,
  lambda_step = 0.1,
  lambda = NULL,
  raise_method = c("closed_form", "grid")
)
```

Arguments

formula	a two-sided formula.
data	a data frame.
method	only "simultaneous" is implemented for the robust raise regression (there is no robust analogue of the sequential one-variable-at-a-time strategy in this package); the argument exists so that passing anything else gives an informative error rather than R's generic "unused argument" message.
cc	Tukey biweight tuning constant (default 4.685). Only used when type = "R".
seed	integer for reproducible direction sampling. Only used when type = "R".
threshold	the SVIF threshold above which a predictor is raised (default 10).
margin	unused for the robust method (its raise parameters are solved in closed form directly against threshold); retained only for argument-signature symmetry with raiseReg().
lambda_max	raise parameters above this are treated as "drop the predictor" rather than "raise it this much" (default 1000).
lambda_step	grid resolution in the raise-parameter search, only used when raise_method = "grid" (default 0.1); ignored by the default closed-form solver.
lambda	optional named numeric vector of user-supplied raise parameters for some or all predictors (the actual multiplier applied, i.e. what is reported in $k). Named entries are used directly instead of being computed automatically, and are never dropped regardless of lambda_max.
raise_method	"closed_form" (default, exact) or "grid" (a literal replica of the original grid search).

Details

Exact finite-sample inference is used throughout, following the sandwich covariance $\Sigma_{RRM} = \sigma^2 \tilde{R}_w^{-1}(Q'_w W Q_w)\tilde{R}_w^{-T}$, the effective residual degrees of freedom $\nu_2 = tr[(I-P_w)W]$, the exactly-unbiased variance estimate $\hat{\sigma}_w^2 = SSE_w/\nu_2$, and a Satterthwaite-Welch degrees of freedom correction ν^* for the resulting Wald t-tests, as derived from first principles for this estimator.

If a predictor's automatically-computed raise parameter would exceed lambda_max, it is dropped from the model entirely (with a message) rather than raised to that extreme a value, and everything – including the weighted QR decomposition and every inference quantity above – is recomputed on the remaining predictors. This is not just a convenience: down-weighting rows can and does induce exact (not just near) collinearity in low-cardinality predictors once enough rows are zeroed out, which no finite raise parameter can fix. Dropped predictors get an NA coefficient, exactly as lm() does for aliased terms; see $dropped.

Value

an object of class "robRaise". In addition to the usual coefficients/fitted.values/residuals, it carries $vif and $svif (before/after matrices, on the weighted design), $cn (before/after Condition Number of the weighted design), $k (the raise parameters actually applied), and $dropped (predictors excluded for exceeding lambda_max).

Examples

```
fit <- robRaise(mpg ~ disp + hp + wt + drat, data = mtcars, seed = 1)
summary(fit)
fit$vif; fit$svif; fit$cn
```

`robRidge` *Robust Ridge Regression*

Description

Fits a robust version of Ridge Regression using one of two strategies.

Usage

```
robRidge(
  formula,
  data,
  k = NULL,
  type = c("MM", "SDO"),
  cc = 4.685,
  maxit = 200,
  tol = 1e-04,
  seed = NULL
)
```

Arguments

`formula`	a two-sided formula.
`data`	a data frame.
`k`	optional fixed biasing parameter; if `NULL` (default) it is estimated from the data (from the seed M-estimate for `type = "MM"`, or from the weighted OLS fit for `type = "SDO"`), following the same formula as `ridgeReg()`.
`type`	`"MM"` (default) or `"SDO"`.
`cc`	tuning constant for Tukey's biweight (default 4.685). Used by both types (IRLS reweighting for `"MM"`; outlyingness weighting for `"SDO"`).
`maxit`	maximum number of IRLS iterations for `type = "MM"` (default 200).
`tol`	convergence tolerance for `type = "MM"` (default 1e-4).
`seed`	optional integer for reproducible direction sampling; only used when `type = "SDO"`.

Details

`type = "MM"` **(default)** A Tukey biweight M-estimate on the mean-centred design provides a seed coefficient vector and residual scale from which the biasing parameter $k = p\hat{\sigma}_R^2/\hat{\beta}_R'\hat{\beta}_R$ is computed (unless `k` is supplied directly), and the final estimate is obtained by iteratively reweighted ridge regression with observation weights from Tukey's biweight function applied to the MAD-standardized residuals at each step (Yohai, 1987; the same MM philosophy as `robRaise()`).

`type = "SDO"` Observations are weighted once, up front, by Tukey's biweight function applied to their Stahel-Donoho projection outlyingness in the joint (response, predictors) space – exactly as in `robRaise()` – and ordinary ridge regression is then applied to the resulting weighted design and response.

Value

an object of class `"robRidge"`.

Examples

```
fit_mm <- robRidge(mpg ~ disp + hp + wt + drat, data = mtcars)
summary(fit_mm)

fit_sdo <- robRidge(mpg ~ disp + hp + wt + drat, data = mtcars,
                    type = "SDO", seed = 1)
summary(fit_sdo)
```

`scaleDat` *Scale a Dataset (Four Conventions)*

Description

Centers and scales the columns of a numeric matrix or data frame using one of four conventions. Deliberately named `scaleDat()` rather than `scale()` to avoid masking `base::scale()`; it still keeps base R's `center`/`scale` logical-flag interface, extended with a `type` argument for three additional conventions.

Usage

```
scaleDat(
  x,
  center = TRUE,
  scale = TRUE,
  type = c("classical", "weighted", "median", "range"),
  cc = 4.685,
  seed = NULL
)
```

Arguments

x	a numeric matrix or data frame (or an object coercible to a numeric matrix).
center	logical; if TRUE (default), subtract the type-appropriate center statistic; if FALSE, don't center.
scale	logical; if TRUE (default), divide by the type-appropriate scale statistic; if FALSE, don't scale.
type	one of "classical" (default), "weighted", "median", "range".
cc	tuning constant for Tukey's biweight (default 4.685). Only used when type = "weighted".
seed	optional integer for reproducible direction sampling. Only used when type = "weighted".

Details

"classical" **(default)** Subtract the column mean, divide by the column standard deviation – i.e. base::scale()'s own algorithm.

"weighted" Subtract a weighted mean, divide by a weighted standard deviation, with weights from Tukey's biweight function applied to each row's Stahel-Donoho projection outlyingness – the same weighting scheme used throughout this package's robust methods (rvif(), robRaise(), robRidge(), robLiu()).

"median" Subtract the column median, divide by the column MADN – the median absolute deviation scaled by 1.4826 so that it estimates the standard deviation consistently under normality; this is stats::mad()'s default.

"range" Subtract the column minimum and divide by the column range (max - min), giving values in [0, 1].

For every type, the base R center/scale logical flags are still honoured: center = FALSE skips the subtraction step (using 0), and scale = FALSE skips the division step (using 1).

Value

the scaled matrix, with attributes "scaled:center" and "scaled:scale" recording what was subtracted/divided by (mirroring base::scale()'s own attributes), plus, for type = "weighted", a "weights" attribute holding the per-observation weights used.

Examples

```
scaleDat(mtcars)                    # same algorithm as base::scale(mtcars)
scaleDat(mtcars, scale = FALSE)    # same algorithm as base::scale(mtcars, scale = FALSE)
scaleDat(mtcars, type = "median")   # median/MADN
scaleDat(mtcars, type = "range")    # min-max to [0, 1]

scaleDat(mtcars, type = "weighted") # Stahel-Donoho + Tukey biweight
```

vif *Variance Inflation Factor (Ordinary and Robust)*

Description

Computes the Variance Inflation Factor (VIF) as the diagonal elements of the inverse of the correlation matrix of the predictors (Marquaridt, 1970). When `type = "O"` (the default) the ordinary Pearson correlation matrix is used. When `type = "R"` the Robust VIF (RVIF) of Jacob and Varadharajan (2024, Sankhya B, 86(2), 845-871) is returned instead: observations are weighted by Tukey's biweight function applied to their Stahel-Donoho projection outlyingness, and a weighted correlation matrix is formed from those weights.

Usage

```
vif(
  object,
  data = NULL,
  type = c("O", "R"),
  cc = 4.685,
  seed = NULL,
  threshold = 10
)

rvif(object, data = NULL, cc = 4.685, seed = NULL, threshold = 10)
```

Arguments

`object`	a two-sided formula, a fitted model object, or a numeric predictor matrix. See Details.
`data`	a data frame; required when `object` is a formula.
`type`	`"O"` (ordinary, default) or `"R"` (robust).
`cc`	Tukey biweight tuning constant (default 4.685). Only used when `type = "R"`.
`seed`	integer for reproducible direction sampling. Only used when `type = "R"`.
`threshold`	VIF threshold flagged in `print` and `plot` (default 10).

Details

Both types accept the same three forms of `object`:

- a **two-sided formula** plus `data`: e.g. `vif(Sepal.Length ~ ., data = iris)`.
- a **fitted model object** from this package (any of `raiseReg`, `ridgeReg`, `liuReg`, `robRaise`, `robRidge`, `robLiu`) or a plain `lm` object.
- a **numeric matrix** of predictors only (no response column).

Value

when `type = "O"`: a named numeric vector of class `"vif"` with `threshold` and `cn` (Condition Number) attributes. When `type = "R"`: a list of class `"rvif"` with elements `$rvif` (the RVIF values), `$cn` (robust Condition Number), `$weights`, and `$outlyingness`.

References

Marquaridt, D.W. (1970). Generalized inverses, ridge regression, biased linear estimation, and nonlinear estimation. *Technometrics*, 12(3), 591-612.

Jacob, J. and Varadharajan, R. (2024). Robust Variance Inflation Factor: A Promising Approach for Collinearity Diagnostics in the Presence of Outliers. *Sankhya B*, 86(2), 845-871.

Examples

```
## Formula interface (works with dot formulas and factors)
vif(Sepal.Length ~ ., data = iris)

## Fitted model interface
fit <- raiseReg(Sepal.Length ~ ., data = iris)
vif(fit)

## lm interface
vif(lm(Sepal.Length ~ ., data = iris))


## type = "R" (robust VIF) requires the mrfDepth package
vif(Sepal.Length ~ ., data = iris, type = "R", seed = 1)
vif(fit, type = "R", seed = 1)
```

Index