Package: fastQR (via r-universe)

November 18, 2026

Type Package

Title Fast QR Decomposition and Update

Version 1.2.0

Date 2026-11-18

Author Mauro Bernardi [aut, cre], Claudio Busatto [aut], Manuela Cattelan [aut]

Maintainer Mauro Bernardi <mauro.bernardi@unipd.it>

Description Methods for computing QR factorizations and updating or downdating QR and triangular factors when rows or columns are added or removed. Includes compact, thin and complete factorizations, least-squares and ridge regression, and cross-validation. Selected dense operations use the BLAS and LAPACK libraries configured for R. Background references include Golub and Van Loan (2013, ISBN:9781421407944) and Bjorck (2015, <doi:10.1007/978-3-319-05089-8>).

License GPL (>= 2)

Imports Rcpp (>= 1.0.10), RcppEigen, Rdpack

LinkingTo Rcpp, RcppArmadillo, RcppEigen

Encoding UTF-8

RdMacros Rdpack

SystemRequirements GNU make

NeedsCompilation yes

Config/roxygen2/version 8.1.0

Config/pak/sysreqs make

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

Date/Publication 2026-11-18 13:10:12 UTC

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

RemoteRef HEAD

RemoteSha 574447b787e91346e14fcac268eb3e4c25a38424

Contents

qr . . . 2
qr_coef . . . 4
qr_fast . . . 5
qr_fitted . . . 8
qr_lapack . . . 10
qr_lm . . . 11
qr_lse_coef . . . 12
qr_lse_fitted . . . 13
qr_lse_Qty . . . 14
qr_lse_Qy . . . 15
qr_lse_resid . . . 16
qr_pivot2perm . . . 17
qr_Q . . . 18
qr_Q_full . . . 20
qr_Q_reduced2full . . . 21
qr_Qty . . . 22
qr_Qy . . . 23
qr_R . . . 24
qr_resid . . . 26
qr_thin . . . 27
qr_X . . . 28
qrchol . . . 30
qrdowndate . . . 30
qrls . . . 34
qrmls . . . 35
qrmridge . . . 37
qrmridge_cv . . . 38
qrridge . . . 41
qrridge_cv . . . 43
qrsolve . . . 45
qrupdate . . . 46
rchol . . . 49
rdowndate . . . 51
rupdate . . . 54

Index **58**

`qr` *The QR factorization of a matrix*

Description

qr provides the QR factorization of the matrix $X \in \mathbb{R}^{n \times p}$ with $n > p$. The QR factorization of the matrix X returns the matrices $Q \in \mathbb{R}^{n \times n}$ and $R \in \mathbb{R}^{n \times p}$ such that $X = QR$. See Golub and Van Loan (2013) for further details on the method.

Usage

```
qr(X, complete = NULL, type = NULL, nb = NULL)
```

Arguments

X	a $n \times p$ matrix.
type	either "givens" or "householder".
nb	integer terminal column-width threshold for recursive QR. NULL or zero selects the non-recursive algorithm. For reduced output use nb >= 2; complete output also supports nb = 1 (single-column terminal panels).
complete	logical expression of length 1. Indicates whether an arbitrary orthogonal completion of the Q matrix is to be made, or whether the R matrix is to be completed by binding zero-value rows beneath the square upper triangle.

Details

When nb is positive and complete is TRUE, recursive Householder panel transformations are applied to the trailing design and accumulated in the full Q matrix. The returned matrices have dimensions n by n and n by p. No orthogonal completion of a separately computed thin basis is used. The recursive reduced-output algorithm is unchanged. Recursive complete QR supports tall and square inputs; for a wide input the existing non-recursive fallback is used with a warning.

For tall inputs with complete = FALSE, the non-recursive Householder and Givens methods form only the n by p Q factor. Their stored transformations are applied in reverse order to the first p columns of the identity, without constructing a complete n by n Q. Rounding in Q can differ from the complete-output path; the R factor uses the same transformations.

Value

A named list containing

Q the Q matrix.

R the R matrix.

References

Golub GH, Van Loan CF (2013). *Matrix computations*, Johns Hopkins Studies in the Mathematical Sciences, Fourth edition. Johns Hopkins University Press, Baltimore, MD. ISBN 978-1-4214-0794-4; 1-4214-0794-9; 978-1-4214-0859-0.

Björck Å (2015). *Numerical methods in matrix computations*, volume 59 of *Texts in Applied Mathematics*. Springer, Cham. ISBN 978-3-319-05088-1; 978-3-319-05098-8. doi:10.1007/9783319-050898.

Björck Å (2024). *Numerical Methods for Least Squares Problems: Second Edition*. Society for Industrial and Applied Mathematics, Philadelphia, PA. doi:10.1137/1.9781611977950. https://doi.org/10.1137/1.97816119779

Bernardi M, Busatto C, Cattelan M (2024). "Fast QR updating methods for statistical applications." 2412.05905, https://arxiv.org/abs/2412.05905.

Examples

```
## generate sample data
set.seed(1234)
n <- 10
p <- 6
X <- matrix(rnorm(n * p, 1), n, p)

## QR factorization via Givens rotation
output <- qr(X, type = "givens", complete = TRUE)
Q      <- output$Q
R      <- output$R

## check
round(Q %*% R - X, 5)
max(abs(Q %*% R - X))

## QR factorization via Householder rotation
output <- qr(X, type = "householder", complete = TRUE)
Q      <- output$Q
R      <- output$R

## check
round(Q %*% R - X, 5)
max(abs(Q %*% R - X))
```

`qr_coef` *Compute least-squares coefficients from a QR decomposition*

Description

Computes the coefficient vector $\widehat{\beta}$ solving the least-squares problem $\min_\beta \|y - X\beta\|_2$, using a QR decomposition stored in compact (Householder) form.

Usage

```
qr_coef(qr, tau, y, pivot = NULL, rank = NULL)
```

Arguments

`qr`	numeric matrix containing the QR decomposition of X in compact form (as returned by `qr_fast()`).
`tau`	numeric vector of Householder coefficients.
`y`	numeric response vector of length n.
`pivot`	optional integer vector of length p containing the 1-based column permutation used during the QR factorization. If supplied, the returned coefficients are re-ordered to match the original column order.
`rank`	optional integer specifying the numerical rank of X. If supplied, only the leading `rank` components are used in the triangular solve.

Details

The coefficients are obtained by first computing $Q^\top y$ and then solving the resulting upper-triangular system involving the matrix R. The orthogonal matrix Q is never formed explicitly.

Value

a numeric vector of regression coefficients.

Note

Wide designs (more columns than rows) are not supported by this model-fitting interface and raise an R error. This is a restriction of the interface, not of QR factorization itself.

Examples

```
set.seed(1)
n <- 10; p <- 4
X <- matrix(rnorm(n * p), n, p)
y <- rnorm(n)

qr_res <- fastQR::qr_fast(X)
coef1  <- fastQR::qr_coef(qr = qr_res$qr, tau = qr_res$qraux, y = y)

## reference computation
coef2 <- base::qr.coef(base::qr(X), y)

max(abs(coef1 - coef2))
```

---

qr_fast *Fast full QR decomposition*

Description

qr_fast provides the fast QR factorization of the matrix $X \in \mathbb{R}^{n\times p}$ with $n > p$. The full QR factorization of the matrix X returns the matrices $Q \in \mathbb{R}^{n\times p}$ and the upper triangular matrix $R \in \mathbb{R}^{p\times p}$ such that $X = QR$. See Golub and Van Loan (2013) for further details on the method.

Usage

```
qr_fast(X, tol = NULL, pivot = NULL)
```

Arguments

X	a $n \times p$ matrix with $n > p$.
tol	the tolerance for detecting linear dependencies in the columns of X.
pivot	a logical value indicating whether to pivot the columns of X. Defaults to FALSE, meaning no pivoting is performed.

Details

The QR decomposition plays an important role in many statistical techniques. In particular it can be used to solve the equation $Ax = b$ for given matrix $A \in \mathbb{R}^{n \times p}$ and vectors $x \in \mathbb{R}^p$ and $b \in \mathbb{R}^n$. It is useful for computing regression coefficients and in applying the Newton-Raphson algorithm.

Value

A named list containing

qr a matrix with the same dimensions as X. The upper triangle contains the R of the decomposition and the lower triangle contains information on the Q of the decomposition (stored in compact form).

qraux a vector of length ncol(x) which contains additional information on Q.

rank the rank of X as computed by the decomposition.

pivot information on the pivoting strategy used during the decomposition.

pivoted a boolean variable returning one if the pivoting has been performed and zero otherwise.

Note

Wide designs are supported in compact form, with min(n,p) Householder coefficients. At least one row is required. This factorization does not imply that the regression routines support underdetermined models.

References

Golub GH, Van Loan CF (2013). *Matrix computations*, Johns Hopkins Studies in the Mathematical Sciences, Fourth edition. Johns Hopkins University Press, Baltimore, MD. ISBN 978-1-4214-0794-4; 1-4214-0794-9; 978-1-4214-0859-0.

Björck Å (2015). *Numerical methods in matrix computations*, volume 59 of *Texts in Applied Mathematics*. Springer, Cham. ISBN 978-3-319-05088-1; 978-3-319-05098-8. doi:10.1007/9783319-050898.

Björck Å (2024). *Numerical Methods for Least Squares Problems: Second Edition*. Society for Industrial and Applied Mathematics, Philadelphia, PA. doi:10.1137/1.9781611977950. https://doi.org/10.1137/1.97816119779

Bernardi M, Busatto C, Cattelan M (2024). "Fast QR updating methods for statistical applications." 2412.05905, https://arxiv.org/abs/2412.05905.

Examples

```
## generate sample data
set.seed(1234)
n <- 12
p <- 5
X <- matrix(rnorm(n * p), n, p)

## get the full QR decomposition with pivot
qr_res <- fastQR::qr_fast(X = X,
                          tol = sqrt(.Machine$double.eps),
                          pivot = TRUE)
```

```
## reconstruct the reduced Q and R matrices
## reduced Q matrix
Q1 <- qr_Q(qr = qr_res$qr, tau = qr_res$qraux,
           rank = qr_res$rank, complete = FALSE)
Q1

## check the Q matrix (orthogonality)
max(abs(crossprod(Q1)-diag(1, p)))

## complete Q matrix
Q2 <- qr_Q(qr = qr_res$qr, tau = qr_res$qraux,
           rank = NULL, complete = TRUE)
Q2

## check the Q matrix (orthogonality)
max(abs(crossprod(Q2)-diag(1, n)))

## reduced R matrix
R1 <- qr_R(qr = qr_res$qr,
           rank = NULL,
           complete = FALSE)

## check that X^TX = R^TR
## get the permutation matrix
P <- qr_pivot2perm(pivot = qr_res$pivot)
max(abs(crossprod(R1 %*% P) - crossprod(X)))
max(abs(crossprod(R1) - crossprod(X %*% t(P))))

## complete R matrix
R2 <- qr_R(qr = qr_res$qr,
           rank = NULL,
           complete = TRUE)

## check that X^TX = R^TR
## get the permutation matrix
P <- qr_pivot2perm(pivot = qr_res$pivot)
max(abs(crossprod(R2 %*% P) - crossprod(X)))
max(abs(crossprod(R2) - crossprod(X %*% t(P))))

## check that X = Q %*% R
max(abs(Q2 %*% R2 %*% P - X))
max(abs(Q1 %*% R1 %*% P - X))

## create data: n > p
set.seed(1234)
n <- 120
p <- 75
X <- matrix(rnorm(n * p), n, p)

## get the full QR decomposition with pivot
qr_res <- fastQR::qr_fast(X = X, pivot = FALSE)
```

```
## reconstruct the reduced Q and R matrices
## reduced Q matrix
Q1 <- qr_Q(qr = qr_res$qr, tau = qr_res$qraux,
           rank = p,
           complete = FALSE)

## check the Q matrix (orthogonality)
max(abs(crossprod(Q1)-diag(1, p)))

## complete Q matrix
Q2 <- qr_Q(qr = qr_res$qr, tau = qr_res$qraux,
           rank = NULL, complete = TRUE)

## check the Q matrix (orthogonality)
max(abs(crossprod(Q2)-diag(1, n)))

## reduced R matrix
R1 <- qr_R(qr = qr_res$qr,
           rank = NULL,
           complete = FALSE)


## check that X^TX = R^TR
max(abs(crossprod(R1) - crossprod(X)))

## complete R matrix
R2 <- qr_R(qr = qr_res$qr,
           rank = NULL,
           complete = TRUE)

## check that X^TX = R^TR
max(abs(crossprod(R2) - crossprod(X)))

## check that X^TX = R^TR
max(abs(crossprod(R2) - crossprod(X)))
max(abs(crossprod(R2) - crossprod(X)))
max(abs(crossprod(R1) - crossprod(X)))

# check that X = Q %*% R
max(abs(Q2 %*% R2 - X))
max(abs(Q1 %*% R1 - X))
```

---

qr_fitted *Compute fitted values from a QR decomposition*

Description

Computes the fitted values $\widehat{y} = X\widehat{\beta}$ for a linear least-squares problem using a QR decomposition stored in compact (Householder) form.

Usage

```
qr_fitted(qr, tau, y)
```

Arguments

qr	Numeric matrix containing the QR decomposition of X in compact form (as returned by `qr_fast()`).
tau	numeric vector of Householder coefficients.
y	numeric response vector of length n.

Details

The fitted values are computed as

$$\widehat{y} = QQ^\top y$$

without explicitly forming the orthogonal matrix Q. The computation relies on the Householder reflectors stored in `qr` and `tau`.

Value

a numeric vector of fitted values $\hat{y}$.

Note

Wide designs (more columns than rows) are not supported by this model-fitting interface and raise an R error. This is a restriction of the interface, not of QR factorization itself.

Examples

```
set.seed(1)
n <- 10; p <- 4
X <- matrix(rnorm(n * p), n, p)
y <- rnorm(n)

qr_res <- fastQR::qr_fast(X)
yhat1  <- fastQR::qr_fitted(qr = qr_res$qr, tau = qr_res$qraux, y = y)

## reference computation
yhat2 <- base::qr.fitted(base::qr(X), y)

max(abs(yhat1 - yhat2))
```

qr_lapack *Unpivoted QR decomposition using R's LAPACK libraries*

Description

Calls DGEQRF explicitly, followed by DORGQR only when an explicit orthogonal factor is requested. No numerical rank is estimated.

Usage

```
qr_lapack(X, output = c("thin", "full", "R", "compact"))
```

Arguments

X	A nonempty finite real numeric matrix; tall, square and wide matrices are supported.
output	Representation to return: "thin", "full", "R", or "compact".

Details

Factor signs are not standardized. Rank-deficient inputs retain the same output dimensions. The input is not modified. The backend uses LAPACK and BLAS selected by R, without requiring the LAPACKE C interface.

Value

For "thin", a list with Q of size n by min(n,p) and R of size min(n,p) by p. For "full", Q is n by n and R is n by p. For "R", the triangular or trapezoidal matrix R alone is returned. For "compact", a list containing the DGEQRF packed matrix qr and Householder coefficients tau is returned. This compact list is not a base R qr object or a fastQR::qr_fast object.

Examples

```
set.seed(1)
X <- matrix(rnorm(40), 10, 4)
f <- qr_lapack(X)
max(abs(f$Q %*% f$R - X))
qr_lapack(X, output = "R")
```

qr_lm *Ordinary least squares for the linear regression model*

Description

qr_lm, or LS for linear regression models, solves the following optimization problem

$$\min_\beta \frac{1}{2}\|y - X\beta\|_2^2,$$

for $y \in \mathbb{R}^n$ and $X \in \mathbb{R}^{n\times p}$ witn $n > p$, to obtain a coefficient vector $\widehat{\beta} \in \mathbb{R}^p$. The design matrix $X \in \mathbb{R}^{n\times p}$ contains the observations for each regressor.

Usage

```
qr_lm(y, X, X_test = NULL)
```

Arguments

y	a vector of length-n response vector.
X	an $(n \times p)$ full column rank matrix of predictors.
X_test	an $(q \times p)$ full column rank matrix. Test set. By default it set to NULL.

Value

A named list containing

coeff a length-p vector containing the solution for the parameters β.

coeff.se a length-p vector containing the standard errors for the estimated regression parameters β.

fitted a length-n vector of fitted values, $\widehat{y} = X\widehat{\beta}$.

residuals a length-n vector of residuals, $\varepsilon = y - \widehat{y}$.

residuals_norm2 the squared L2-norm of the residuals, $\|\varepsilon\|_2^2$.

y_norm2 the squared L2-norm of the response variable, $\|y\|_2^2$.

R the $R \in \mathbb{R}^{p\times p}$ upper triangular matrix of the QR decomposition.

L the inverse of the $R \in \mathbb{R}^{p\times p}$ upper triangular matrix of the QR decomposition $L = R^{-1}$.

XTX the Gram matrix $X^\top X \in \mathbb{R}^{p\times p}$ of the least squares problem.

XTX_INV the inverse of the Gram matrix $X^\top X \in \mathbb{R}^{p\times p}$ of the least squares problem $(X^\top X)^{-1}$.

XTy A vector equal to $X^\top y$, the cross-product of the design matrix X with the response vector y.

sigma2_hat An estimate of the error variance σ^2, computed as the residual sum of squares divided by the residual degrees of freedom $\widehat{\sigma}^2 = \frac{\|y - X\hat{\beta}\|_2^2}{df}$

df The residual degrees of freedom, given by $n - p$, where n is the number of observations and p is the number of estimated parameters.

R2 R^2, coefficient of determination, measure of goodness-of-fit of the model.

predicted predicted values for the test set, $X_{\text{test}}\widehat{\beta}$. It is only available if X_test is not NULL.

Note

Wide designs (nrow(X) < ncol(X)) are not supported by this model-fitting interface and raise an R error.

Examples

```
## generate sample data
## create data: n > p
set.seed(1234)
n    <- 12
n0   <- 3
p    <- 7
X    <- matrix(rnorm(n * p), n, p)
b    <- rep(1, p)
sig2 <- 0.25
y    <- X %*% b + sqrt(sig2) * rnorm(n)
summary(lm(y~X))

## test
X_test <- matrix(rnorm(n0 * p), n0, p)

## lm
qr_lm(y = y, X = X, X_test = X_test)
qr_lm(y = y, X = X)
```

`qr_lse_coef` *Compute least-squares coefficients using QR decomposition*

Description

Computes the coefficient vector $\hat{\beta}$ solving the least-squares problem $\min_\beta \|y - X\beta\|_2$, using a QR decomposition computed internally.

Usage

```
qr_lse_coef(X, y)
```

Arguments

`X`	numeric matrix of dimension $n \times p$.
`y`	numeric response vector of length n.

Details

The QR decomposition of X is computed internally. The coefficients are obtained by first computing $Q^\top y$ and then solving the resulting upper-triangular system involving the matrix R. The orthogonal matrix Q is never formed explicitly.

This function is intended as a convenience wrapper for least-squares estimation when the explicit QR factors are not required.

Value

a numeric vector of regression coefficients.

Note

Wide designs (more columns than rows) are not supported by this model-fitting interface and raise an R error. This is a restriction of the interface, not of QR factorization itself.

Examples

```
set.seed(1)
n <- 10; p <- 4
X <- matrix(rnorm(n * p), n, p)
y <- rnorm(n)

coef1 <- fastQR::qr_lse_coef(X, y)

## reference computation
coef2 <- base::qr.coef(base::qr(X), y)

max(abs(coef1 - coef2))
```

---

qr_lse_fitted *Compute fitted values using QR decomposition*

Description

Computes the fitted values $\hat{y} = X\hat{\beta}$ for a linear least-squares problem using a QR decomposition computed internally.

Usage

```
qr_lse_fitted(X, y)
```

Arguments

X	numeric matrix of dimension $n \times p$.
y	numeric response vector of length n.

Details

The QR decomposition of X is computed internally. The fitted values are obtained as

$$\widehat{y} = QQ^{\top} y$$

without explicitly forming the orthogonal matrix Q.

This function is intended as a convenience wrapper for least-squares computations when the explicit QR factors are not required.

Value

a numeric vector of fitted values $\hat{y}$.

Note

Wide designs (more columns than rows) are not supported by this model-fitting interface and raise an R error. This is a restriction of the interface, not of QR factorization itself.

Examples

```
set.seed(1)
n <- 10; p <- 4
X <- matrix(rnorm(n * p), n, p)
y <- rnorm(n)

yhat1 <- fastQR::qr_lse_fitted(X, y)

## reference computation
yhat2 <- base::qr.fitted(base::qr(X), y)

max(abs(yhat1 - yhat2))
```

---

qr_lse_Qty *Compute Q'y for a least-squares problem*

Description

Computes the product $Q^\top y$, where Q is the orthogonal matrix from the QR decomposition of the design matrix X.

Usage

```
qr_lse_Qty(X, y)
```

Arguments

X	numeric matrix of dimension $n \times p$.
y	numeric response vector of length n.

Details

The QR decomposition of X is computed internally, and the orthogonal matrix Q is never formed explicitly. The product $Q^\top y$ is evaluated efficiently using Householder reflectors.

This function is intended as a convenience wrapper for least-squares computations when the explicit QR factors are not required.

Value

a numeric vector equal to $Q^\top y$.

Note

The complete orthogonal transformation is applied directly, without forming Q. Tall, square and wide designs are supported; both input and output vectors have length nrow(X).

Examples

```
set.seed(1)
n <- 10; p <- 4
X <- matrix(rnorm(n * p), n, p)
y <- rnorm(n)

res1 <- fastQR::qr_lse_Qty(X, y)

## reference computation
res2 <- crossprod(base::qr.Q(base::qr(X), complete = TRUE), y)

max(abs(res1 - drop(res2)))
```

---

qr_lse_Qy *Compute Qy for a least-squares problem*

Description

Computes the product Qy, where Q is the orthogonal matrix from the QR decomposition of the design matrix X.

Usage

```
qr_lse_Qy(X, y)
```

Arguments

X	numeric matrix of dimension $n \times p$.
y	numeric response vector of length n.

Details

The QR decomposition of X is computed internally, and the orthogonal matrix Q is never formed explicitly. The product Qy is evaluated efficiently using Householder reflectors.

This function is intended as a convenience wrapper for least-squares computations when the explicit QR factors are not required.

Value

a numeric vector equal to Qy.

Note

The complete orthogonal transformation is applied directly, without forming Q. Tall, square and wide designs are supported; both input and output vectors have length nrow(X).

Examples

```
set.seed(1)
n <- 10; p <- 4
X <- matrix(rnorm(n * p), n, p)
y <- rnorm(n)

res1 <- fastQR::qr_lse_Qy(X, y)

## reference computation
res2 <- base::qr.Q(base::qr(X), complete = TRUE) %*% y

max(abs(res1 - drop(res2)))
```

---

qr_lse_resid *Compute residuals using QR decomposition*

Description

Computes the residual vector $r = y - \hat{y}$ for a linear least-squares problem using a QR decomposition computed internally.

Usage

```
qr_lse_resid(X, y)
```

Arguments

X	numeric matrix of dimension $n \times p$.
y	numeric response vector of length n.

Details

The QR decomposition of X is computed internally. The residuals are obtained as

$$r = y - QQ^\top y$$

without explicitly forming the orthogonal matrix Q.

This function is intended as a convenience wrapper for least-squares computations when the explicit QR factors are not required.

Value

a numeric vector of residuals.

Note

Wide designs (more columns than rows) are not supported by this model-fitting interface and raise an R error. This is a restriction of the interface, not of QR factorization itself.

Examples

```
set.seed(1)
n <- 10; p <- 4
X <- matrix(rnorm(n * p), n, p)
y <- rnorm(n)

r1 <- fastQR::qr_lse_resid(X, y)

## reference computation
r2 <- base::qr.resid(base::qr(X), y)

max(abs(r1 - r2))
```

---

qr_pivot2perm *Reconstruct the permutation matrix from the pivot vector.*

Description

returns the permutation matrix for the QR decomposition.

Usage

```
qr_pivot2perm(pivot)
```

Arguments

pivot	a vector of dimension n of pivot elements from the QR factorization.

Value

the perumutation matrix P of dimension $n \times n$.

References

Golub GH, Van Loan CF (2013). *Matrix computations*, Johns Hopkins Studies in the Mathematical Sciences, Fourth edition. Johns Hopkins University Press, Baltimore, MD. ISBN 978-1-4214-0794-4; 1-4214-0794-9; 978-1-4214-0859-0.

Björck Å (2015). *Numerical methods in matrix computations*, volume 59 of *Texts in Applied Mathematics*. Springer, Cham. ISBN 978-3-319-05088-1; 978-3-319-05098-8. doi:10.1007/9783319-050898.

Björck Å (2024). *Numerical Methods for Least Squares Problems: Second Edition*. Society for Industrial and Applied Mathematics, Philadelphia, PA. doi:10.1137/1.9781611977950. https://doi.org/10.1137/1.97816119779

Bernardi M, Busatto C, Cattelan M (2024). "Fast QR updating methods for statistical applications." 2412.05905, https://arxiv.org/abs/2412.05905.

Examples

```
## generate sample data
set.seed(1234)
n <- 12
p <- 5
X <- matrix(rnorm(n * p), n, p)

## get the full QR decomposition with pivot
qr_res <- fastQR::qr_fast(X = X,
                          tol = sqrt(.Machine$double.eps),
                          pivot = TRUE)

## get the pivot matrix
P <- qr_pivot2perm(qr_res$pivot)
```

`qr_Q` *Reconstruct the Q, matrix from a QR object.*

Description

returns the Q matrix of the full QR decomposition. If $r = \text{rank}(X) < p$, then only the reduced $Q \in \mathbb{R}^{n \times r}$ matrix is returned.

Usage

```
qr_Q(qr, tau, rank = NULL, complete = NULL)
```

Arguments

`qr`	object representing a QR decomposition. This will typically have come from a previous call to qr.
`tau`	a vector of length $ncol(X)$ which contains additional information on Q. It corresponds to qraux from a previous call to qr.

rank	the rank of x as computed by the decomposition.
complete	logical flag (length 1). Indicates whether to compute the full $Q \in \boldsymbol{R}^{n \times n}$ or the thin $Q \in \boldsymbol{R}^{n \times p}$. If $r = \mathrm{rank}(X) < p$, then only the reduced $Q \in \mathbb{R}^{n \times r}$ matrix is returned.

Value

returns part or all of Q, the order-n orthogonal (unitary) transformation represented by qr. If the supplied rank is $r < p$, Q has r columns, including zero when $r = 0$, regardless of complete. Otherwise, it has n columns when complete is TRUE and p columns when complete is FALSE.

Note

For wide compact factors, missing rank with complete = TRUE uses min(n,p). A supplied rank must not exceed min(n,p). The existing rank-dependent output rule still applies.

References

Golub GH, Van Loan CF (2013). *Matrix computations*, Johns Hopkins Studies in the Mathematical Sciences, Fourth edition. Johns Hopkins University Press, Baltimore, MD. ISBN 978-1-4214-0794-4; 1-4214-0794-9; 978-1-4214-0859-0.

Björck Å (2015). *Numerical methods in matrix computations*, volume 59 of *Texts in Applied Mathematics*. Springer, Cham. ISBN 978-3-319-05088-1; 978-3-319-05098-8. doi:10.1007/9783319-050898.

Björck Å (2024). *Numerical Methods for Least Squares Problems: Second Edition*. Society for Industrial and Applied Mathematics, Philadelphia, PA. doi:10.1137/1.9781611977950. https://doi.org/10.1137/1.97816119779

Bernardi M, Busatto C, Cattelan M (2024). "Fast QR updating methods for statistical applications." 2412.05905, https://arxiv.org/abs/2412.05905.

Examples

```
## generate sample data
set.seed(1234)
n <- 12
p <- 5
X <- matrix(rnorm(n * p), n, p)

## get the full QR decomposition with pivot
qr_res <- fastQR::qr_fast(X = X,
                          tol = sqrt(.Machine$double.eps),
                          pivot = TRUE)

## get the full Q matrix
Q1 <- qr_Q(qr_res$qr, qr_res$qraux, complete = TRUE)

## check the Q matrix (orthogonality)
max(abs(crossprod(Q1)-diag(1, n)))

## get the reduced Q matrix
Q2 <- qr_Q(qr_res$qr, qr_res$qraux, qr_res$rank, complete = FALSE)
```

```
## check the Q matrix (orthogonality)
max(abs(crossprod(Q2)-diag(1, p)))
```

`qr_Q_full` *Reconstruct the full Q matrix from the qr object.*

Description

returns the full $Q \in \mathbb{R}^{n \times n}$ matrix.

Usage

```
qr_Q_full(qr, tau)
```

Arguments

`qr`	object representing a QR decomposition. This will typically have come from a previous call to qr.
`tau`	a vector of length $ncol(X)$ which contains additional information on Q. It corresponds to qraux from a previous call to qr.

Value

returns the matrix $Q \in \mathbb{R}^{n \times n}$.

Note

Both tall and wide compact factors are supported. At least min(nrow(qr),ncol(qr)) Householder coefficients must be supplied.

References

Golub GH, Van Loan CF (2013). *Matrix computations*, Johns Hopkins Studies in the Mathematical Sciences, Fourth edition. Johns Hopkins University Press, Baltimore, MD. ISBN 978-1-4214-0794-4; 1-4214-0794-9; 978-1-4214-0859-0.

Björck Å (2015). *Numerical methods in matrix computations*, volume 59 of *Texts in Applied Mathematics*. Springer, Cham. ISBN 978-3-319-05088-1; 978-3-319-05098-8. doi:10.1007/9783319-050898.

Björck Å (2024). *Numerical Methods for Least Squares Problems: Second Edition*. Society for Industrial and Applied Mathematics, Philadelphia, PA. doi:10.1137/1.9781611977950. https://doi.org/10.1137/1.97816119779

Bernardi M, Busatto C, Cattelan M (2024). "Fast QR updating methods for statistical applications." 2412.05905, https://arxiv.org/abs/2412.05905.

Examples

```
## create data: n > p
set.seed(1234)
n <- 12
p <- 7
X <- matrix(rnorm(n * p), n, p)

## get the full QR decomposition with pivot
qr_res <- fastQR::qr_fast(X = X,
                          tol = sqrt(.Machine$double.eps),
                          pivot = TRUE)

## complete the reduced Q matrix
Q <- fastQR::qr_Q_full(qr  = qr_res$qr,
                       tau = qr_res$qraux)

## check the Q matrix (orthogonality)
max(abs(crossprod(Q)-diag(1, n)))
```

`qr_Q_reduced2full` *Reconstruct the full Q matrix from the reduced Q matrix.*

Description

returns the full $Q \in \mathbb{R}^{n \times n}$ matrix.

Usage

```
qr_Q_reduced2full(Q)
```

Arguments

`Q`	a $n \times p$ reduced Q matrix from the QR decomposition (with $n > p$).

Details

The supplied columns must be finite and orthonormal. The maximum absolute entry of Q'Q - I must not exceed 100 * .Machine$double.eps * nrow(Q). The leading columns of the returned matrix preserve the supplied values exactly. For a matrix with zero columns the result is the identity matrix. The remaining columns are obtained from a Householder orthogonal completion and need not match the previous completion algorithm.

Value

a $n \times n$ orthogonal matrix Q.

References

Golub GH, Van Loan CF (2013). *Matrix computations*, Johns Hopkins Studies in the Mathematical Sciences, Fourth edition. Johns Hopkins University Press, Baltimore, MD. ISBN 978-1-4214-0794-4; 1-4214-0794-9; 978-1-4214-0859-0.

Björck Å (2015). *Numerical methods in matrix computations*, volume 59 of *Texts in Applied Mathematics*. Springer, Cham. ISBN 978-3-319-05088-1; 978-3-319-05098-8. doi:10.1007/9783319-050898.

Björck Å (2024). *Numerical Methods for Least Squares Problems: Second Edition*. Society for Industrial and Applied Mathematics, Philadelphia, PA. doi:10.1137/1.9781611977950. https://doi.org/10.1137/1.97816119779

Bernardi M, Busatto C, Cattelan M (2024). "Fast QR updating methods for statistical applications." 2412.05905, https://arxiv.org/abs/2412.05905.

Examples

```
## create data: n > p
set.seed(1234)
n <- 12
p <- 7
X <- matrix(rnorm(n * p), n, p)

## get the full QR decomposition with pivot
qr_res <- fastQR::qr_fast(X = X,
                          tol = sqrt(.Machine$double.eps),
                          pivot = TRUE)

## reconstruct the reduced Q matrix
Q1 <- qr_Q(qr = qr_res$qr, tau = qr_res$qraux,
           rank = qr_res$rank, complete = FALSE)

## complete the reduced Q matrix
Q2 <- fastQR::qr_Q_reduced2full(Q = Q1)
R  <- fastQR::qr_R(qr = qr_res$qr, rank = NULL, complete = TRUE)

X1 <- qr_X(Q = Q2, R = R, pivot = qr_res$pivot)
max(abs(X - X1))
```

`qr_Qty` *Multiply Q by a vector using a QR decomposition*

Description

Computes $Q^\top y$, where Q is the orthogonal matrix from the QR decomposition stored in compact (Householder) form.

Usage

```
qr_Qty(qr, tau, y)
```

Arguments

qr	numeric matrix containing the QR decomposition in compact form (as returned by qr_fast()).
tau	numeric vector of Householder coefficients.
y	numeric vector of length n.

Details

The orthogonal matrix Q is not formed explicitly. The product $Q^\top y$ is computed efficiently using the Householder reflectors stored in qr and tau.

Value

a numeric vector equal to $Q^\top y$.

Note

Tall, square and wide compact factors are supported. The response vector must have length nrow(qr).

Examples

```
set.seed(1)
n <- 10; p <- 4
X <- matrix(rnorm(n * p), n, p)
y <- rnorm(n)

qr_res <- fastQR::qr_fast(X)
res1   <- fastQR::qr_Qty(qr = qr_res$qr, tau = qr_res$qraux, y = y)

## reference computation
Q    <- base::qr.Q(base::qr(X), complete = TRUE)
res2 <- crossprod(Q, y)

max(abs(res1 - drop(res2)))
```

---

qr_Qy *Multiply Q by a vector using a QR decomposition*

Description

Computes Qy, where Q is the orthogonal matrix from the QR decomposition stored in compact (Householder) form.

Usage

```
qr_Qy(qr, tau, y)
```

Arguments

qr	numeric matrix containing the QR decomposition in compact form (as returned by qr_fast()).
tau	numeric vector of Householder coefficients.
y	numeric vector of length n.

Details

The orthogonal matrix Q is not formed explicitly. The product Qy is computed efficiently using the Householder reflectors stored in qr and tau.

Value

a numeric vector equal to Qy.

Note

Tall, square and wide compact factors are supported. The response vector must have length nrow(qr).

Examples

```
set.seed(1)
n <- 10; p <- 4
X <- matrix(rnorm(n * p), n, p)
y <- rnorm(n)

qr_res <- fastQR::qr_fast(X)
res1   <- fastQR::qr_Qy(qr = qr_res$qr, tau = qr_res$qraux, y = y)

## reference computation
Q    <- base::qr.Q(base::qr(X), complete = TRUE)
res2 <- Q %*% y

max(abs(res1 - drop(res2)))
```

---

qr_R *Reconstruct the R, matrix from a QR object.*

Description

returns the R matrix of the full QR decomposition. If $r = \mathrm{rank}(X) < p$, then only the reduced $R \in \mathbb{R}^{r \times p}$ matrix is returned.

Usage

```
qr_R(qr, rank = NULL, pivot = NULL, complete = NULL)
```

Arguments

qr	object representing a QR decomposition. This will typically have come from a previous call to qr.
rank	the rank of x as computed by the decomposition.
pivot	a vector of length p, specifying the permutation of the columns of X applied during the QR decomposition process. The default is NULL if no pivoting has been applied.
complete	logical flag (length 1). Indicates whether the R matrix is to be completed by binding zero-value rows beneath the square upper triangle. If $r = \text{rank}(X) < p$, then only the reduced $R \in \mathbb{R}^{r \times p}$ matrix is returned.

Value

returns part or all of R. If the supplied rank is $r < p$, R has r rows, including zero when $r = 0$, regardless of complete. Otherwise, it has n rows when complete is TRUE and p rows when complete is FALSE.

Note

For wide compact factors, the default rank is min(n,p), yielding an n by p factor. A supplied rank must not exceed min(n,p).

References

Golub GH, Van Loan CF (2013). *Matrix computations*, Johns Hopkins Studies in the Mathematical Sciences, Fourth edition. Johns Hopkins University Press, Baltimore, MD. ISBN 978-1-4214-0794-4; 1-4214-0794-9; 978-1-4214-0859-0.

Björck Å (2015). *Numerical methods in matrix computations*, volume 59 of *Texts in Applied Mathematics*. Springer, Cham. ISBN 978-3-319-05088-1; 978-3-319-05098-8. doi:10.1007/9783319-050898.

Björck Å (2024). *Numerical Methods for Least Squares Problems: Second Edition*. Society for Industrial and Applied Mathematics, Philadelphia, PA. doi:10.1137/1.9781611977950. https://doi.org/10.1137/1.97816119779

Bernardi M, Busatto C, Cattelan M (2024). "Fast QR updating methods for statistical applications." 2412.05905, https://arxiv.org/abs/2412.05905.

Examples

```
## generate sample data
set.seed(1234)
n <- 12
p <- 5
X <- matrix(rnorm(n * p), n, p)

## get the full QR decomposition with pivot
qr_res <- fastQR::qr_fast(X = X,
                          tol = sqrt(.Machine$double.eps),
                          pivot = TRUE)
```

```
## get the full R matrix
R1 <- qr_R(qr_res$qr, complete = TRUE)

## check that X^TX = R^TR
## get the permutation matrix
P <- qr_pivot2perm(pivot = qr_res$pivot)
max(abs(crossprod(R1 %*% P) - crossprod(X)))
max(abs(crossprod(R1) - crossprod(X %*% t(P))))

## get the reduced R matrix
R2 <- qr_R(qr_res$qr, qr_res$rank, complete = FALSE)

## check that X^TX = R^TR
## get the permutation matrix
P <- qr_pivot2perm(pivot = qr_res$pivot)
max(abs(crossprod(R2 %*% P) - crossprod(X)))
max(abs(crossprod(R2) - crossprod(X %*% t(P))))
```

`qr_resid` *Compute residuals from a QR decomposition*

Description

Computes the residual vector $r = y - \widehat{y}$ for a linear least-squares problem using a QR decomposition stored in compact (Householder) form.

Usage

```
qr_resid(qr, tau, y)
```

Arguments

`qr`	numeric matrix containing the QR decomposition of X in compact form (as returned by `qr_fast()`).
`tau`	numeric vector of Householder coefficients.
`y`	numeric response vector of length n.

Details

The residuals are computed as

$$r = y - QQ^\top y$$

without explicitly forming the orthogonal matrix Q. The computation relies on the Householder reflectors stored in `qr` and `tau`.

Value

a numeric vector of residuals of dimension n.

Note

Wide designs (more columns than rows) are not supported by this model-fitting interface and raise an R error. This is a restriction of the interface, not of QR factorization itself.

Examples

```
set.seed(1)
n <- 10; p <- 4
X <- matrix(rnorm(n * p), n, p)
y <- rnorm(n)

qr_res <- fastQR::qr_fast(X)
r1     <- fastQR::qr_resid(qr = qr_res$qr, tau = qr_res$qraux, y = y)

## reference computation
r2 <- base::qr.resid(base::qr(X), y)

max(abs(r1 - r2))
```

---

qr_thin *Fast thin QR decomposition*

Description

qr_thin provides the thin QR factorization of the matrix $X \in \mathbb{R}^{n \times p}$ with $n > p$. The thin QR factorization of the matrix X returns the matrices $Q \in \mathbb{R}^{n \times p}$ and the upper triangular matrix $R \in \mathbb{R}^{p \times p}$ such that $X = QR$. See Golub and Van Loan (2013) for further details on the method.

Usage

```
qr_thin(X)
```

Arguments

X	a $n \times p$ matrix with $n > p$.

Value

A named list containing

Q the Q matrix.

R the R matrix.

Note

This thin-factor interface requires nrow(X) >= ncol(X); wide designs raise an R error. Use qr() or qr_fast() for wide factorizations. At least one row is required.

References

Golub GH, Van Loan CF (2013). *Matrix computations*, Johns Hopkins Studies in the Mathematical Sciences, Fourth edition. Johns Hopkins University Press, Baltimore, MD. ISBN 978-1-4214-0794-4; 1-4214-0794-9; 978-1-4214-0859-0.

Björck Å (2015). *Numerical methods in matrix computations*, volume 59 of *Texts in Applied Mathematics*. Springer, Cham. ISBN 978-3-319-05088-1; 978-3-319-05098-8. doi:10.1007/9783319-050898.

Björck Å (2024). *Numerical Methods for Least Squares Problems: Second Edition*. Society for Industrial and Applied Mathematics, Philadelphia, PA. doi:10.1137/1.9781611977950. https://doi.org/10.1137/1.97816119779

Bernardi M, Busatto C, Cattelan M (2024). "Fast QR updating methods for statistical applications." 2412.05905, https://arxiv.org/abs/2412.05905.

Examples

```
## generate sample data
set.seed(1234)
n <- 12
p <- 5
X <- matrix(rnorm(n * p), n, p)

## get the thin QR factorization
output <- qr_thin(X = X)
Q      <- output$Q
R      <- output$R

## check
max(abs(Q %*% R - X))
```

`qr_X` *Reconstruct the original matrix from which the object was constructed* $X \in \mathbb{R}\hat{}n \times p$ *from the Q and R matrices of the QR decomposition.*

Description

returns the $X \in \mathbb{R}^{n \times p}$ matrix.

Usage

```
qr_X(Q, R, pivot = NULL)
```

Arguments

`Q`	either the reduced $Q \in \mathbb{R}^{n \times p}$ of full $Q \in \mathbb{R}^{n \times n}$, Q matrix obtained from the QR decomposition.
`R`	either the reduced $R \in \mathbb{R}^{p \times p}$ of full $R \in \mathbb{R}^{n \times p}$, R matrix obtained from the QR decomposition.

pivot — a vector of length p, specifying the permutation of the columns of X applied during the QR decomposition process. The default is NULL if no pivoting has been applied.

Value

returns the matrix X.

References

Golub GH, Van Loan CF (2013). *Matrix computations*, Johns Hopkins Studies in the Mathematical Sciences, Fourth edition. Johns Hopkins University Press, Baltimore, MD. ISBN 978-1-4214-0794-4; 1-4214-0794-9; 978-1-4214-0859-0.

Björck Å (2015). *Numerical methods in matrix computations*, volume 59 of *Texts in Applied Mathematics*. Springer, Cham. ISBN 978-3-319-05088-1; 978-3-319-05098-8. doi:10.1007/9783319-050898.

Björck Å (2024). *Numerical Methods for Least Squares Problems: Second Edition*. Society for Industrial and Applied Mathematics, Philadelphia, PA. doi:10.1137/1.9781611977950. https://doi.org/10.1137/1.97816119779

Bernardi M, Busatto C, Cattelan M (2024). "Fast QR updating methods for statistical applications." 2412.05905, https://arxiv.org/abs/2412.05905.

Examples

```
## generate sample data
set.seed(1234)
n <- 12
p <- 5
X <- matrix(rnorm(n * p), n, p)

## get the full QR decomposition with pivot
qr_res <- fastQR::qr_fast(X = X,
                          tol = sqrt(.Machine$double.eps),
                          pivot = TRUE)

## get the full QR decomposition with pivot
qr_res <- fastQR::qr_fast(X = X, pivot = TRUE)

## get the Q and R matrices
Q  <- qr_Q(qr = qr_res$qr, tau = qr_res$qraux, rank = qr_res$rank, complete = TRUE)
R  <- qr_R(qr = qr_res$qr, rank = qr_res$rank, complete = TRUE)
X1 <- qr_X(Q = Q, R = R, pivot = qr_res$pivot)
max(abs(X1 - X))

## get the full QR decomposition without pivot
qr_res <- fastQR::qr_fast(X = X, pivot = FALSE)

## get the Q and R matrices
Q  <- qr_Q(qr = qr_res$qr, tau = qr_res$qraux, rank = p, complete = FALSE)
R  <- qr_R(qr = qr_res$qr, rank = NULL, complete = FALSE)
X1 <- qr_X(Q = Q, R = R, pivot = NULL)
```

```
max(abs(X1 - X))
```

---

qrchol *Cholesky decomposition via QR factorization.*

Description

qrchol, provides the Cholesky decomposition of the symmetric and positive definite matrix $X^\top X \in \mathbb{R}^{p\times p}$, where $X \in \mathbb{R}^{n\times p}$ is the input matrix.

Usage

```
qrchol(X, nb = NULL)
```

Arguments

X	an $(n \times p)$ matrix.
nb	number of blocks for the recursive block QR decomposition, default is NULL.

Value

an upper triangular matrix of dimension $p \times p$ which represents the Cholesky decomposition of $X^\top X$.

Note

Require nrow(X) >= ncol(X). Wide designs are rejected: this interface returns a square triangular factor.

---

qrdowndate *Fast downdating of the QR factorization*

Description

qrdowndate provides the update of the QR factorization after the deletion of $m > 1$ rows or columns to the matrix $X \in \mathbb{R}^{n\times p}$ with $n > p$. The QR factorization of the matrix $X \in \mathbb{R}^{n\times p}$ returns the matrices $Q \in \mathbb{R}^{n\times n}$ and $R \in \mathbb{R}^{n\times p}$ such that $X = QR$. The Q and R matrices are factorized as $Q = \begin{bmatrix} Q_1 & Q_2 \end{bmatrix}$ and $R = \begin{bmatrix} R_1 \\ R_2 \end{bmatrix}$, with $Q_1 \in \mathbb{R}^{n\times p}$, $Q_2 \in \mathbb{R}^{n\times(n-p)}$ such that $Q_1^\top Q_2 = Q_2^\top Q_1 = 0$ and $R_1 \in \mathbb{R}^{p\times p}$ upper triangular matrix and $R_2 \in \mathbb{R}^{(n-p)\times p}$. qrdowndate accepts in input the matrices Q and either the complete matrix R or the reduced one, R_1. See Golub and Van Loan (2013) for further details on the method.

Usage

```
qrdowndate(Q, R, k, m = NULL, type = NULL, fast = NULL, complete = NULL)
```

Arguments

Q	a complete $n \times n$ orthogonal matrix, or a thin $n \times p$ matrix for column deletion with reduced output.
R	the complete $n \times p$ upper trapezoidal factor, or its $p \times p$ upper triangular block for column deletion with reduced output.
k	position where the columns or the rows are removed.
m	number of columns or rows to be removed. Default is $m = 1$.
type	either 'row' of 'column', for deleting rows or columns. Default is 'column'.
fast	fast mode: disable to check whether the provided matrices are valid inputs. Default is FALSE.
complete	logical expression of length 1. Indicates whether an arbitrary orthogonal completion of the Q matrix is to be made, or whether the R matrix is to be completed by binding zero-value rows beneath the square upper triangle.

Details

For column deletion with complete = FALSE, Q and R may independently be complete or thin factors. Only the first p columns of Q and the leading p by p block of R are used. The input matrices are unchanged and the output has dimensions n by (p-m) and (p-m) by (p-m). The reduced path avoids internal n by n factor allocations; double R matrices are read through non-owning views before the required thin blocks are copied. Integer inputs may require coercion to double. Structural dimension and index checks remain active in fast mode; finite-value checks on the used blocks are skipped. For complete output or row deletion, complete input factors are still required. For row deletion, require 1 <= m <= n-p and 1 <= k <= n-m+1. Deleting n-p rows is allowed and produces square factors. These bounds are checked even in fast mode. The reduced output can be supplied directly to another reduced column deletion.

Value

A named list containing

Q the updated Q matrix.

R the updated R matrix.

Note

Wide input factors are not supported and are rejected even in fast mode.

References

Golub GH, Van Loan CF (2013). *Matrix computations*, Johns Hopkins Studies in the Mathematical Sciences, Fourth edition. Johns Hopkins University Press, Baltimore, MD. ISBN 978-1-4214-0794-4; 1-4214-0794-9; 978-1-4214-0859-0.

Björck Å (2015). *Numerical methods in matrix computations*, volume 59 of *Texts in Applied Mathematics*. Springer, Cham. ISBN 978-3-319-05088-1; 978-3-319-05098-8. doi:10.1007/9783319-050898.

Björck Å (2024). *Numerical Methods for Least Squares Problems: Second Edition*. Society for Industrial and Applied Mathematics, Philadelphia, PA. doi:10.1137/1.9781611977950. https://doi.org/10.1137/1.97816119779

Bernardi M, Busatto C, Cattelan M (2024). "Fast QR updating methods for statistical applications." 2412.05905, https://arxiv.org/abs/2412.05905.

Examples

```
## Remove one column
## generate sample data
set.seed(10)
n      <- 10
p      <- 6
X      <- matrix(rnorm(n * p, 1), n, p)

## get the initial QR factorization
output <- fastQR::qr(X, type = "householder",
                     nb = NULL,
                     complete = TRUE)
Q      <- output$Q
R      <- output$R

## select the column to be deleted
## from X and update X
k  <- 2
X1 <- X[, -k]

## downdate the QR decomposition
out <- fastQR::qrdowndate(Q = Q, R = R,
                          k = k, m = 1,
                          type = "column",
                          fast = FALSE,
                          complete = TRUE)

## check
round(out$Q %*% out$R - X1, 5)
max(abs(out$Q %*% out$R - X1))

## Remove m columns
## generate sample data
set.seed(10)
n      <- 10
p      <- 6
X      <- matrix(rnorm(n * p, 1), n, p)

## get the initial QR factorization
output <- fastQR::qr(X, type = "householder",
                     nb = NULL,
                     complete = TRUE)
Q      <- output$Q
```

```
R      <- output$R

## select the column to be deleted from X
## and update X
m  <- 2
k  <- 2
X1 <- X[, -c(k,k+m-1)]

## downdate the QR decomposition
out <- fastQR::qrdowndate(Q = Q, R = R,
                          k = k, m = 2,
                          type = "column",
                          fast = TRUE,
                          complete = FALSE)

## check
round(out$Q %*% out$R - X1, 5)
max(abs(out$Q %*% out$R - X1))

## Remove one row
## generate sample data
set.seed(10)
n      <- 10
p      <- 6
X      <- matrix(rnorm(n * p, 1), n, p)

## get the initial QR factorization
output <- fastQR::qr(X, type = "householder",
                     nb = NULL,
                     complete = TRUE)
Q      <- output$Q
R      <- output$R

## select the row to be deleted from X and update X
k  <- 5
X1 <- X[-k,]

## downdate the QR decomposition
out <- fastQR::qrdowndate(Q = Q, R = R,
                          k = k, m = 1,
                          type = "row",
                          fast = FALSE,
                          complete = TRUE)

## check
round(out$Q %*% out$R - X1, 5)
max(abs(out$Q %*% out$R - X1))

## Remove m rows
## generate sample data
set.seed(10)
n      <- 10
p      <- 6
```

```
X      <- matrix(rnorm(n * p, 1), n, p)

## get the initial QR factorization
output <- fastQR::qr(X, type = "householder",
                     nb = NULL,
                     complete = TRUE)
Q      <- output$Q
R      <- output$R

## select the rows to be deleted from X and update X
k  <- 5
m  <- 2
X1 <- X[-c(k,k+1),]

## downdate the QR decomposition
out <- fastQR::qrdowndate(Q = Q, R = R,
                          k = k, m = m,
                          type = "row",
                          fast = FALSE,
                          complete = TRUE)

## check
round(out$Q %*% out$R - X1, 5)
max(abs(out$Q %*% out$R - X1))
```

---

qrls *Ordinary least squares for the linear regression model*

Description

qrls, or LS for linear regression models, solves the following optimization problem

$$\min_\beta \frac{1}{2}\|y - X\beta\|_2^2,$$

for $y \in \mathbb{R}^n$ and $X \in \mathbb{R}^{n\times p}$, to obtain a coefficient vector $\widehat{\beta} \in \mathbb{R}^p$. The design matrix $X \in \mathbb{R}^{n\times p}$ contains the observations for each regressor.

Usage

```
qrls(y, X, X_test = NULL, type = NULL)
```

Arguments

y	a vector of length-n response vector.
X	an $(n \times p)$ full column rank matrix of predictors.
X_test	an $(q \times p)$ full column rank matrix. Test set. By default it set to NULL.
type	either "QR" or "R". Specifies the type of decomposition to use: "QR" for the QR decomposition or "R" for the Cholesky factorization of $A^\top A$. The default is "QR".

Value

A named list containing

coeff a length-p vector containing the solution for the parameters β.

fitted a length-n vector of fitted values, $\widehat{y} = X\widehat{\beta}$.

residuals a length-n vector of residuals, $\varepsilon = y - \widehat{y}$.

residuals_norm2 the L2-norm of the residuals, $\|\varepsilon\|_2^2$.

y_norm2 the L2-norm of the response variable. $\|y\|_2^2$.

XTX_Qmat Q matrix of the QR decomposition of the matrix $X^\top X$.

XTX_Rmat R matrix of the QR decomposition of the matrix $X^\top X$.

QXTy $QX^\top y$, where Q matrix of the QR decomposition of the matrix $X^\top X$.

R2 R^2, coefficient of determination, measure of goodness-of-fit of the model.

predicted predicted values for the test set, $X_{\text{test}}\widehat{\beta}$. It is only available if X_test is not NULL.

Note

Wide designs (more columns than rows) are not supported by this model-fitting interface and raise an R error. This is a restriction of the interface, not of QR factorization itself.

Examples

```
## generate sample data
set.seed(10)
n         <- 30
p         <- 6
X         <- matrix(rnorm(n * p, 1), n, p)
X[,1]     <- 1
eps       <- rnorm(n, sd = 0.5)
beta      <- rep(0, p)
beta[1:3] <- 1
beta[4:5] <- 2
y         <- X %*% beta + eps
X_test    <- matrix(rnorm(5 * p, 1), 5, p)
output    <-  fastQR::qrls(y = y, X = X, X_test = X_test)
output$coeff
```

---

qrmls *Ordinary least squares for the linear multivariate regression model*

Description

qrmls, or LS for linear multivariate regression models, solves the following optimization problem

$$\min_\beta \frac{1}{2}\|Y - XB\|_2^2,$$

for $Y \in \mathbb{R}^{n\times q}$ and $X \in \mathbb{R}^{n\times p}$, to obtain a coefficient matrix $\widehat{B} \in \mathbb{R}^{p\times q}$. The design matrix $X \in \mathbb{R}^{n\times p}$ contains the observations for each regressor.

Usage

```
qrmls(Y, X, X_test = NULL, type = NULL)
```

Arguments

Y	a matrix of dimension ($n \times q$ response variables.
X	an $(n \times p)$ full column rank matrix of predictors.
X_test	an $(q \times p)$ full column rank matrix. Test set. By default it set to NULL.
type	either "QR" or "R". Specifies the type of decomposition to use: "QR" for the QR decomposition or "R" for the Cholesky factorization of $A^\top A$. The default is "QR".

Value

A named list containing

coeff a matrix of dimension $p \times q$ containing the solution for the parameters B.

fitted a matrix of dimension $n \times q$ of fitted values, $\widehat{Y} = X\widehat{B}$.

residuals a matrix of dimension $n \times q$ of residuals, $\varepsilon = Y - \widehat{Y}$.

XTX the matrix $X^\top X$.

Sigma_hat a matrix of dimension $q \times q$ containing the estimated residual variance-covariance matrix.

df degrees of freedom.

R R matrix of the QR decomposition of the matrix $X^\top X$.

XTy $X^\top y$.

R2 R^2, coefficient of determination, measure of goodness-of-fit of the model.

predicted predicted values for the test set, $X_{\text{test}}\widehat{B}$. It is only available if X_test is not NULL.

PMSE

Examples

```
## generate sample data
set.seed(10)
n         <- 30
p         <- 6
q         <- 3
X         <- matrix(rnorm(n * p, 1), n, p)
X[,1]     <- 1
eps       <- matrix(rnorm(n*q), n, q)
B         <- matrix(0, p, q)
B[,1]     <- rep(1, p)
B[,2]     <- rep(2, p)
B[,3]     <- rep(-1, p)
Y         <- X %*% B + eps
X_test    <- matrix(rnorm(5 * p, 1), 5, p)
output    <- fastQR::qrmls(Y = Y, X = X, X_test = X_test, type = "QR")
output$coeff
```

`qrmridge` *multivariate ridge regression*

Description

Fits the normalized ridge criterion

$$\frac{1}{n}\|Y - XB\|_F^2 + \lambda\|B\|_F^2.$$

All columns, including any intercept column supplied by the user, are penalized. No centering, scaling or intercept insertion is performed internally.

Usage

```
qrmridge(Y, X, lambda, X_test = NULL, type = NULL)
```

Arguments

`Y`	A finite response matrix with n rows and q columns.
`X`	A finite n by p design matrix with n >= 1 and p >= 1. Strictly positive penalties permit p > n.
`lambda`	A finite nonnegative penalty (one scalar).
`X_test`	Optional finite prediction matrix with p columns. Prediction rows need not exceed p.
`type`	Either "QR" (the default) or "R". QR factorizes the normalized penalized system; R uses the triangular factor of an augmented design and two triangular solves.

Details

Both backends solve

$$(X^\top X + n\lambda I)\widehat{B} = X^\top Y,$$

with a vector right-hand side for a univariate response. For type="QR", Q times R equals X transpose X divided by n plus lambda I. For type="R", R transpose R equals X transpose X plus n times lambda I; Q is not constructed or returned. The names XTX and XTy/XTY always refer to the unpenalized, unnormalized cross-products. Positive penalties allow rank-deficient designs. A zero penalty requires full column rank. Strictly positive penalties also allow more predictors than observations.

Value

A named list with the following components:

coeff Fitted coefficient matrix, p by q.

fitted Fitted responses on the training data.

residuals Observed minus fitted responses.

XTX The unpenalized cross-product X transpose X.

XTY The unpenalized cross-product X transpose Y.

XTy Compatibility alias for XTY. There are no duplicated field names.

Sigma_hat Residual cross-product divided by n-p, or NA when n<=p. This is a residual scaling diagnostic, not an unbiased ridge covariance estimator.

df Legacy residual dimension: q*(n-p) when n>=p, and NA when p>n. This is not the effective degrees of freedom of the ridge fitted-value map.

R Backend-specific triangular factor; see Details.

Q Present only for type="QR"; orthogonal factor of the normalized penalized system.

QXTY For type="QR", Q transpose times X transpose Y/n.

QTY For type="R", the intermediate right-hand side after solving the transposed triangular system.

R2 One minus the residual sum of squares divided by the centered total sum of squares, summing across responses in the multivariate case. NA when the total is zero.

predicted Predictions at X_test, present only when X_test is supplied.

QXTy Compatibility alias for QXTY, present only for type="QR".

Examples

```
## generate sample data
set.seed(10)
n         <- 30
p         <- 6
q         <- 3
X         <- matrix(rnorm(n * p, 1), n, p)
X[,1]     <- 1
eps       <- matrix(rnorm(n*q), n, q)
B         <- matrix(0, p, q)
B[,1]     <- rep(1, p)
B[,2]     <- rep(2, p)
B[,3]     <- rep(-1, p)
Y         <- X %*% B + eps
X_test    <- matrix(rnorm(5 * p, 1), 5, p)
output    <- fastQR::qrmridge(Y = Y, X = X, lambda = 1, X_test = X_test, type = "QR")
output$coeff
```

---

qrmridge_cv *Cross-validation for multivariate ridge regression*

Description

Fits the normalized ridge criterion

$$\frac{1}{n}\|Y - XB\|_F^2 + \lambda\|B\|_F^2.$$

All columns, including any intercept column supplied by the user, are penalized. No centering, scaling or intercept insertion is performed internally. The penalty is selected by cross-validation and the model is refitted on all observations.

Usage

```
qrmridge_cv(Y, X, lambda, k = NULL, seed = NULL, X_test = NULL, type = NULL)
```

Arguments

`Y`	A finite response matrix with n rows and q columns.
`X`	A finite n by p design matrix with n >= 1 and p >= 1. Strictly positive penalties permit p > n.
`lambda`	A nonempty grid of finite nonnegative penalties.
`k`	Number of folds, between 2 and n. Zero requests leave-one-out CV. The default is min(10,n). When the grid includes zero, every training fold must have at least p rows.
`seed`	Integer seed used to generate reproducible folds; default 1234.
`X_test`	Optional finite prediction matrix with p columns. Prediction rows need not exceed p.
`type`	Either "QR" (the default) or "R". QR factorizes the normalized penalized system; R uses the triangular factor of an augmented design and two triangular solves.

Details

Both backends solve

$$(X^\top X + n\lambda I)\widehat{B} = X^\top Y,$$

with a vector right-hand side for a univariate response. For type="QR", Q times R equals X transpose X divided by n plus lambda I. For type="R", R transpose R equals X transpose X plus n times lambda I; Q is not constructed or returned. The names XTX and XTy/XTY always refer to the unpenalized, unnormalized cross-products. Positive penalties allow rank-deficient designs. A zero penalty requires full column rank. Strictly positive penalties also allow more predictors than observations. Each fold uses its own training size in the normalized criterion. Augmented factors are cached for distinct training sizes and updated by removing held-out rows. A factor is recomputed if a numerical downdate fails its cross-product check. The final fit uses the full sample size and the selected penalty. The seed argument resets R's random-number generator; foldid records the generated split.

Value

A named list with the following components:

coeff Fitted coefficient matrix, p by q.

fitted Fitted responses on the training data.

residuals Observed minus fitted responses.

XTX The unpenalized cross-product X transpose X.

XTY The unpenalized cross-product X transpose Y.

XTy Compatibility alias for XTY. There are no duplicated field names.

Sigma_hat Residual cross-product divided by n-p, or NA when n<=p. This is a residual scaling diagnostic, not an unbiased ridge covariance estimator.

df Legacy residual dimension: q*(n-p) when n>=p, and NA when p>n. This is not the effective degrees of freedom of the ridge fitted-value map.

R Backend-specific triangular factor; see Details.

Q Present only for type="QR"; orthogonal factor of the normalized penalized system.

QXTY For type="QR", Q transpose times X transpose Y/n.

QTY For type="R", the intermediate right-hand side after solving the transposed triangular system.

R2 One minus the residual sum of squares divided by the centered total sum of squares, summing across responses in the multivariate case. NA when the total is zero.

predicted Predictions at X_test, present only when X_test is supplied.

QXTy Compatibility alias for QXTY, present only for type="QR".

lambda The supplied penalty grid.

mse Mean held-out squared errors, averaged equally across folds. Each fold error averages across observations and responses.

mse_min Smallest value of mse.

mse_min_ind Zero-based index of the first minimum, preserving the existing indexing convention.

lambda_min Selected penalty. Ties are resolved by the first supplied grid entry.

foldid Integer fold labels in the original observation order.

Examples

```
## generate sample data
set.seed(10)
n         <- 30
p         <- 6
q         <- 3
X         <- matrix(rnorm(n * p, 1), n, p)
X[,1]     <- 1
eps       <- matrix(rnorm(n*q), n, q)
B         <- matrix(0, p, q)
B[,1]     <- rep(1, p)
B[,2]     <- rep(2, p)
B[,3]     <- rep(-1, p)
```

```
Y         <- X %*% B + eps
X_test    <- matrix(rnorm(5 * p, 1), 5, p)
output    <- fastQR::qrmridge_cv(Y = Y, X = X, lambda = c(1,2),
                                 k = 5, seed = 12, X_test = X_test, type = "QR")
output$coeff
```

---

qrridge *ridge regression*

Description

Fits the normalized ridge criterion

$$\frac{1}{n}\|y - X\beta\|_2^2 + \lambda\|\beta\|_2^2.$$

All columns, including any intercept column supplied by the user, are penalized. No centering, scaling or intercept insertion is performed internally.

Usage

```
qrridge(y, X, lambda, X_test = NULL, type = NULL)
```

Arguments

y	A finite response vector of length n.
X	A finite n by p design matrix with n >= 1 and p >= 1. Strictly positive penalties permit p > n.
lambda	A finite nonnegative penalty (one scalar).
X_test	Optional finite prediction matrix with p columns. Prediction rows need not exceed p.
type	Either "QR" (the default) or "R". QR factorizes the normalized penalized system; R uses the triangular factor of an augmented design and two triangular solves.

Details

Both backends solve

$$(X^\top X + n\lambda I)\widehat{B} = X^\top Y,$$

with a vector right-hand side for a univariate response. For type="QR", Q times R equals X transpose X divided by n plus lambda I. For type="R", R transpose R equals X transpose X plus n times lambda I; Q is not constructed or returned. The names XTX and XTy/XTY always refer to the unpenalized, unnormalized cross-products. Positive penalties allow rank-deficient designs. A zero penalty requires full column rank. Strictly positive penalties also allow more predictors than observations.

Value

A named list with the following components:

coeff Fitted coefficient vector of length p.

fitted Fitted responses on the training data.

residuals Observed minus fitted responses.

XTX The unpenalized cross-product X transpose X.

XTy The unpenalized cross-product X transpose y.

residuals_norm2 Sum of squared residuals.

y_norm2 Centered response sum of squares.

sigma2_hat Sum of squared residuals divided by n-p, or NA when n<=p. This is a residual scaling diagnostic, not an unbiased ridge variance estimator.

df Legacy residual dimension: n-p when n>=p, and NA when p>n. This is not the effective degrees of freedom of the ridge fitted-value map.

R Backend-specific triangular factor; see Details.

Q Present only for type="QR"; orthogonal factor of the normalized penalized system.

QXTy For type="QR", Q transpose times X transpose y/n.

QTy For type="R", the intermediate right-hand side after solving the transposed triangular system.

R2 One minus the residual sum of squares divided by the centered total sum of squares, summing across responses in the multivariate case. NA when the total is zero.

predicted Predictions at X_test, present only when X_test is supplied.

Examples

```
## generate sample data
set.seed(10)
n         <- 30
p         <- 6
X         <- matrix(rnorm(n * p, 1), n, p)
X[,1]     <- 1
eps       <- rnorm(n, sd = 0.5)
beta      <- rep(0, p)
beta[1:3] <- 1
beta[4:5] <- 2
y         <- X %*% beta + eps
X_test    <- matrix(rnorm(5 * p, 1), 5, p)
output    <-  fastQR::qrridge(y = y, X = X,
                              lambda = 0.2,
                              X_test = X_test)
output$coeff
```

qrridge_cv *Cross-validation for ridge regression*

Description

Fits the normalized ridge criterion

$$\frac{1}{n}\|y - X\beta\|_2^2 + \lambda\|\beta\|_2^2.$$

All columns, including any intercept column supplied by the user, are penalized. No centering, scaling or intercept insertion is performed internally. The penalty is selected by cross-validation and the model is refitted on all observations.

Usage

```
qrridge_cv(y, X, lambda, k = NULL, seed = NULL, X_test = NULL, type = NULL)
```

Arguments

y	A finite response vector of length n.
X	A finite n by p design matrix with n >= 1 and p >= 1. Strictly positive penalties permit p > n.
lambda	A nonempty grid of finite nonnegative penalties.
k	Number of folds, between 2 and n. Zero requests leave-one-out CV. The default is min(10,n). When the grid includes zero, every training fold must have at least p rows.
seed	Integer seed used to generate reproducible folds; default 1234.
X_test	Optional finite prediction matrix with p columns. Prediction rows need not exceed p.
type	Either "QR" (the default) or "R". QR factorizes the normalized penalized system; R uses the triangular factor of an augmented design and two triangular solves.

Details

Both backends solve

$$(X^\top X + n\lambda I)\widehat{B} = X^\top Y,$$

with a vector right-hand side for a univariate response. For type="QR", Q times R equals X transpose X divided by n plus lambda I. For type="R", R transpose R equals X transpose X plus n times lambda I; Q is not constructed or returned. The names XTX and XTy/XTY always refer to the unpenalized, unnormalized cross-products. Positive penalties allow rank-deficient designs. A zero penalty requires full column rank. Strictly positive penalties also allow more predictors than observations. Each fold uses its own training size in the normalized criterion. Augmented factors are cached for distinct training sizes and updated by removing held-out rows. A factor is recomputed if a numerical downdate fails its cross-product check. The final fit uses the full sample size and the selected penalty. The seed argument resets R's random-number generator; foldid records the generated split.

Value

A named list with the following components:

coeff Fitted coefficient vector of length p.

fitted Fitted responses on the training data.

residuals Observed minus fitted responses.

XTX The unpenalized cross-product X transpose X.

XTy The unpenalized cross-product X transpose y.

residuals_norm2 Sum of squared residuals.

y_norm2 Centered response sum of squares.

sigma2_hat Sum of squared residuals divided by n-p, or NA when n<=p. This is a residual scaling diagnostic, not an unbiased ridge variance estimator.

df Legacy residual dimension: n-p when n>=p, and NA when p>n. This is not the effective degrees of freedom of the ridge fitted-value map.

R Backend-specific triangular factor; see Details.

Q Present only for type="QR"; orthogonal factor of the normalized penalized system.

QXTy For type="QR", Q transpose times X transpose y/n.

QTy For type="R", the intermediate right-hand side after solving the transposed triangular system.

R2 One minus the residual sum of squares divided by the centered total sum of squares, summing across responses in the multivariate case. NA when the total is zero.

predicted Predictions at X_test, present only when X_test is supplied.

lambda The supplied penalty grid.

mse Mean held-out squared errors, averaged equally across folds. Each fold error averages across observations and responses.

mse_min Smallest value of mse.

mse_min_ind Zero-based index of the first minimum, preserving the existing indexing convention.

lambda_min Selected penalty. Ties are resolved by the first supplied grid entry.

foldid Integer fold labels in the original observation order.

Examples

```
## generate sample data
set.seed(10)
n         <- 30
p         <- 6
X         <- matrix(rnorm(n * p, 1), n, p)
X[,1]     <- 1
eps       <- rnorm(n)
beta      <- rep(1, p)
y         <- X %*% beta + eps
X_test    <- matrix(rnorm(5 * p, 1), 5, p)
output    <- fastQR::qrridge_cv(y = y, X = X, lambda = c(1,2),
                                k = 5, seed = 12, X_test = X_test, type = "QR")
output$coeff
```

qrsolve *Solution of linear system of equations, via the QR decomposition.*

Description

solves systems of equations $Ax = b$, for $A \in \mathbb{R}^{n \times p}$ and $b \in \mathbb{R}^n$, via the QR decomposition.

Usage

```
qrsolve(A, b, type = NULL, nb = NULL)
```

Arguments

A	an $(n \times p)$ full column rank matrix.
b	a vector of dimension n.
type	either "QR" or "R". Specifies the type of decomposition to use: "QR" for the QR decomposition or "R" for the Cholesky factorization of $A^\top A$. The default is "QR".
nb	number of blocks for the recursive block QR decomposition, default is NULL.

Value

x a vector of dimension p that satisfies $Ax = b$.

References

Golub GH, Van Loan CF (2013). *Matrix computations*, Johns Hopkins Studies in the Mathematical Sciences, Fourth edition. Johns Hopkins University Press, Baltimore, MD. ISBN 978-1-4214-0794-4; 1-4214-0794-9; 978-1-4214-0859-0.

Björck Å (2015). *Numerical methods in matrix computations*, volume 59 of *Texts in Applied Mathematics*. Springer, Cham. ISBN 978-3-319-05088-1; 978-3-319-05098-8. doi:10.1007/9783319-050898.

Björck Å (2024). *Numerical Methods for Least Squares Problems: Second Edition*. Society for Industrial and Applied Mathematics, Philadelphia, PA. doi:10.1137/1.9781611977950. https://doi.org/10.1137/1.97816119779

Bernardi M, Busatto C, Cattelan M (2024). "Fast QR updating methods for statistical applications." 2412.05905, https://arxiv.org/abs/2412.05905.

Examples

```
## generate sample data
set.seed(1234)
n <- 10
p <- 4
A <- matrix(rnorm(n * p, 1), n, p)
b <- rnorm(n)
```

```
## solve the system of linear equations using qr
x1 <- fastQR::qrsolve(A = A, b = b)
x1

## solve the system of linear equations using rb qr
x2 <- fastQR::qrsolve(A = A, b = b, nb = 2)
x2

## check
round(x1 - solve(crossprod(A)) %*% crossprod(A, b), 5)
round(x2 - solve(crossprod(A)) %*% crossprod(A, b), 5)
```

`qrupdate` *Fast updating of the QR factorization*

Description

qrupdate provides the update of the QR factorization after the addition of $m > 1$ rows or columns to the matrix $X \in \mathbb{R}^{n \times p}$ with $n > p$. The QR factorization of the matrix X returns the matrices $Q \in \mathbb{R}^{n \times n}$ and $R \in \mathbb{R}^{n \times p}$ such that $X = QR$. The Q and R matrices are factorized as $Q = \begin{bmatrix} Q_1 & Q_2 \end{bmatrix}$ and $R = \begin{bmatrix} R_1 \\ R_2 \end{bmatrix}$, with $Q_1 \in \mathbb{R}^{n \times p}$, $Q_2 \in \mathbb{R}^{n \times (n-p)}$ such that $Q_1^\top Q_2 = Q_2^\top Q_1 = 0$ and $R_1 \in \mathbb{R}^{p \times p}$ upper triangular matrix and $R_2 \in \mathbb{R}^{(n-p) \times p}$. qrupdate accepts in input the matrices Q and either the complete matrix R or the reduced one, R_1. See Golub and Van Loan (2013) for further details on the method.

Usage

```
qrupdate(Q, R, k, U, type = NULL, fast = NULL, complete = NULL)
```

Arguments

`Q`	a $n \times p$ matrix.
`R`	a $p \times p$ upper triangular matrix.
`k`	position where the columns or the rows are added.
`U`	either a $n \times m$ matrix or a $p \times m$ matrix of columns or rows to be added.
`type`	either 'row' of 'column', for adding rows or columns. Default is 'column'.
`fast`	fast mode: disable to check whether the provided matrices are valid inputs. Default is FALSE.
`complete`	logical expression of length 1. Indicates whether an arbitrary orthogonal completion of the Q matrix is to be made, or whether the R matrix is to be completed by binding zero-value rows beneath the square upper triangle.

Value

A named list containing

Q the updated Q matrix.

R the updated R matrix.

Note

Wide input factors and column additions producing more columns than rows are rejected even in fast mode.

References

Golub GH, Van Loan CF (2013). *Matrix computations*, Johns Hopkins Studies in the Mathematical Sciences, Fourth edition. Johns Hopkins University Press, Baltimore, MD. ISBN 978-1-4214-0794-4; 1-4214-0794-9; 978-1-4214-0859-0.

Björck Å (2015). *Numerical methods in matrix computations*, volume 59 of *Texts in Applied Mathematics*. Springer, Cham. ISBN 978-3-319-05088-1; 978-3-319-05098-8. doi:10.1007/9783319-050898.

Björck Å (2024). *Numerical Methods for Least Squares Problems: Second Edition*. Society for Industrial and Applied Mathematics, Philadelphia, PA. doi:10.1137/1.9781611977950. https://doi.org/10.1137/1.97816119779

Bernardi M, Busatto C, Cattelan M (2024). "Fast QR updating methods for statistical applications." 2412.05905, https://arxiv.org/abs/2412.05905.

Examples

```
## Add one column
## generate sample data
set.seed(1234)
n <- 12
p <- 5
X <- matrix(rnorm(n * p, 1), n, p)

## get the initial QR factorization
output <- qr(X, complete = TRUE)
Q      <- output$Q
R      <- output$R

## create column u to be added
k  <- p+1
u  <- matrix(rnorm(n), n, 1)
X1 <- cbind(X, u)

## update the QR decomposition
out <- fastQR::qrupdate(Q = Q, R = R,
                        k = k, U = u,
                        type = "column",
                        fast = FALSE,
                        complete = TRUE)
```

```
## check
round(out$Q %*% out$R - X1, 5)
max(abs(out$Q %*% out$R - X1))

## Add m columns
## create data: n > p
set.seed(1234)
n <- 10
p <- 5
X <- matrix(rnorm(n * p, 1), n, p)

## get the initial QR factorization
output <- fastQR::qr(X, complete = TRUE)
Q      <- output$Q
R      <- output$R

## create the matrix of two columns to be added
## in position 2
k  <- 2
m  <- 2
U  <- matrix(rnorm(n*m), n, m)
X1 <- cbind(X[,1:(k-1)], U, X[,k:p])

# update the QR decomposition
out <- fastQR::qrupdate(Q = Q, R = R,
                        k = k, U = U, type = "column",
                       fast = FALSE, complete = TRUE)

## check
round(out$Q %*% out$R - X1, 5)
max(abs(out$Q %*% out$R - X1))

## Add one row
## create data: n > p
set.seed(1234)
n <- 12
p <- 5
X <- matrix(rnorm(n * p, 1), n, p)

## get the initial QR factorization
output <- fastQR::qr(X, complete = TRUE)
Q      <- output$Q
R      <- output$R
R1     <- R[1:p,]

## create the row u to be added
u  <- matrix(data = rnorm(p), p, 1)
k  <- n+1
if (k<=n) {
  X1 <- rbind(rbind(X[1:(k-1), ], t(u)), X[k:n, ])
} else {
  X1 <- rbind(rbind(X, t(u)))
}
```

```
## update the QR decomposition
out <- fastQR::qrupdate(Q = Q, R = R,
                        k = k, U = u,
                        type = "row",
                        complete = TRUE)

## check
round(out$Q %*% out$R - X1, 5)
max(abs(out$Q %*% out$R - X1))

## Add m rows
## create data: n > p
set.seed(1234)
n <- 12
p <- 5
X <- matrix(rnorm(n * p, 1), n, p)

## get the initial QR factorization
output <- fastQR::qr(X, complete = TRUE)
Q      <- output$Q
R      <- output$R
R1     <- R[1:p,]

## create the matrix of rows U to be added:
## two rows in position 5
m  <- 2
U  <- matrix(data = rnorm(p*m), p, m)
k  <- 5
if (k<=n) {
  X1 <- rbind(rbind(X[1:(k-1), ], t(U)), X[k:n, ])
} else {
  X1 <- rbind(rbind(X, t(U)))
}

## update the QR decomposition
out <- fastQR::qrupdate(Q = Q, R = R,
                        k = k, U = U,
                        type = "row",
                        complete = FALSE)

## check
round(out$Q %*% out$R - X1, 5)
max(abs(out$Q %*% out$R - X1))
```

---

rchol *Cholesky decomposition via R factorization.*

Description

rchol, provides the Cholesky decomposition of the symmetric and positive definite matrix $X^\top X \in \mathbb{R}^{p\times p}$, where $X \in \mathbb{R}^{n\times p}$ is the input matrix.

Usage

```
rchol(X)
```

Arguments

X	an $(n \times p)$ matrix, with $n \geq p$. If $n < p$ an error message is returned.

Value

an upper triangular matrix of dimension $p \times p$ which represents the Cholesky decomposition of $X^\top X$.

References

Golub GH, Van Loan CF (2013). *Matrix computations*, Johns Hopkins Studies in the Mathematical Sciences, Fourth edition. Johns Hopkins University Press, Baltimore, MD. ISBN 978-1-4214-0794-4; 1-4214-0794-9; 978-1-4214-0859-0.

Björck Å (2015). *Numerical methods in matrix computations*, volume 59 of *Texts in Applied Mathematics*. Springer, Cham. ISBN 978-3-319-05088-1; 978-3-319-05098-8. doi:10.1007/9783319-050898.

Björck Å (2024). *Numerical Methods for Least Squares Problems: Second Edition*. Society for Industrial and Applied Mathematics, Philadelphia, PA. doi:10.1137/1.9781611977950. https://doi.org/10.1137/1.97816119779

Bernardi M, Busatto C, Cattelan M (2024). "Fast QR updating methods for statistical applications." 2412.05905, https://arxiv.org/abs/2412.05905.

Examples

```
set.seed(1234)
n <- 10
p <- 6
X <- matrix(rnorm(n * p, 1), n, p)

## compute the Cholesky decomposition of X^TX
S <- fastQR::rchol(X = X)
S

## check
round(S - chol(crossprod(X)), 5)
```

`rdowndate` *Fast downdating of the R matrix*

Description

rdowndate provides the update of the thin R matrix of the QR factorization after the deletion of $m \geq 1$ rows or columns to the matrix $X \in \mathbb{R}^{n \times p}$ with $n > p$. The R factorization of the matrix X returns the upper triangular matrix $R \in \mathbb{R}^{p \times p}$ such that $X^\top X = R^\top R$. See Golub and Van Loan (2013) for further details on the method.

Usage

```
rdowndate(R, k = NULL, m = NULL, U = NULL, fast = NULL, type = NULL)
```

Arguments

`R`	a $p \times p$ upper triangular matrix.
`k`	position where the columns or the rows are removed.
`m`	number of columns or rows to be removed. It is not required when deleting columns. If NULL, it defaults to the number of columns in U.
`U`	a $p \times m$ matrix of rows to be removed. It should only be provided when rows are being removed.
`fast`	fast mode: disable to check whether the provided matrices are valid inputs. Default is FALSE.
`type`	either 'row' of 'column', for removing rows or columns.

Value

R the updated R matrix.

Note

Require a square reduced R factor; rectangular factors, including wide factors, are rejected even in fast mode.

References

Golub GH, Van Loan CF (2013). *Matrix computations*, Johns Hopkins Studies in the Mathematical Sciences, Fourth edition. Johns Hopkins University Press, Baltimore, MD. ISBN 978-1-4214-0794-4; 1-4214-0794-9; 978-1-4214-0859-0.

Björck Å (2015). *Numerical methods in matrix computations*, volume 59 of *Texts in Applied Mathematics*. Springer, Cham. ISBN 978-3-319-05088-1; 978-3-319-05098-8. doi:10.1007/9783319-050898.

Björck Å (2024). *Numerical Methods for Least Squares Problems: Second Edition*. Society for Industrial and Applied Mathematics, Philadelphia, PA. doi:10.1137/1.9781611977950. https://doi.org/10.1137/1.97816119779

Bernardi M, Busatto C, Cattelan M (2024). "Fast QR updating methods for statistical applications." 2412.05905, https://arxiv.org/abs/2412.05905.

Examples

```
## Remove one column
## generate sample data
set.seed(10)
n      <- 10
p      <- 6
X      <- matrix(rnorm(n * p, 1), n, p)

## get the initial QR factorization
output <- fastQR::qr(X, type = "householder",
                     nb = NULL,
                     complete = TRUE)
Q      <- output$Q
R      <- output$R
R1     <- R[1:p,]

## select the column to be deleted from X and update X
k  <- 2
X1 <- X[, -k]

## downdate the R decomposition
R2 <- fastQR::rdowndate(R = R1, k = k,
                        m = 1, type = "column")

## check
max(abs(crossprod(R2) - crossprod(X1)))

## Remove m columns
## generate sample data
set.seed(10)
n      <- 10
p      <- 6
X      <- matrix(rnorm(n * p, 1), n, p)

## get the initial QR factorization
output <- fastQR::qr(X, type = "householder",
                     nb = NULL,
                     complete = TRUE)
Q      <- output$Q
R      <- output$R
R1     <- R[1:p,]

## select the column to be deleted from X and update X
k  <- 2
X1 <- X[, -c(k,k+1)]

## downdate the R decomposition
R2 <- fastQR::rdowndate(R = R1, k = k,
                        m = 2, type = "column")

## check
max(abs(crossprod(R2) - crossprod(X1)))
```

```
## Remove one row
## generate sample data
set.seed(10)
n      <- 10
p      <- 6
X      <- matrix(rnorm(n * p, 1), n, p)

## get the initial QR factorization
output <- fastQR::qr(X, type = "householder",
                     nb = NULL,
                     complete = TRUE)
Q      <- output$Q
R      <- output$R
R1     <- R[1:p,]

# select the row to be deleted from X and update X
k  <- 5
X1 <- X[-k,]
U  <- as.matrix(X[k,], p, 1)

## downdate the R decomposition
R2 <-  rdowndate(R = R1, k = k, m = 1,
                 U = U, fast = FALSE, type = "row")

## check
max(abs(crossprod(R2) - crossprod(X1)))

## Remove m rows
## create data: n > p
set.seed(10)
n      <- 10
p      <- 6
X      <- matrix(rnorm(n * p, 1), n, p)
output <- fastQR::qr(X, type = "householder",
                     nb = NULL,
                     complete = TRUE)
Q      <- output$Q
R      <- output$R
R1     <- R[1:p,]

## select the rows to be deleted from X and update X
k  <- 2
m  <- 2
X1 <- X[-c(k,k+m-1),]
U  <- t(X[k:(k+m-1), ])

## downdate the R decomposition
R2 <- rdowndate(R = R1, k = k, m = m,
                U = U, fast = FALSE, type = "row")

## check
max(abs(crossprod(R2) - crossprod(X1)))
```

`rupdate` *Fast updating of the R matrix*

Description

updates the R factorization when $m \geq 1$ rows or columns are added to the matrix $X \in \mathbb{R}^{n \times p}$, where $n > p$. The R factorization of X produces an upper triangular matrix $R \in \mathbb{R}^{p \times p}$ such that $X^\top X = R^\top R$. For more details on this method, refer to Golub and Van Loan (2013). Columns can only be added in positions $p+1$ through $p+m$, while the position of added rows does not need to be specified.

Usage

```
rupdate(X, R, U, fast = NULL, type = NULL)
```

Arguments

`X`	the current $n \times p$ matrix, prior to the addition of any rows or columns.
`R`	a $p \times p$ upper triangular matrix.
`U`	either a $n \times m$ matrix or a $p \times m$ matrix of columns or rows to be added.
`type`	either 'row' of 'column', for adding rows or columns.
`fast`	fast mode: disable to check whether the provided matrices are valid inputs. Default is FALSE.

Value

R the updated R matrix.

References

Golub GH, Van Loan CF (2013). *Matrix computations*, Johns Hopkins Studies in the Mathematical Sciences, Fourth edition. Johns Hopkins University Press, Baltimore, MD. ISBN 978-1-4214-0794-4; 1-4214-0794-9; 978-1-4214-0859-0.

Björck Å (2015). *Numerical methods in matrix computations*, volume 59 of *Texts in Applied Mathematics*. Springer, Cham. ISBN 978-3-319-05088-1; 978-3-319-05098-8. doi:10.1007/9783319-050898.

Björck Å (2024). *Numerical Methods for Least Squares Problems: Second Edition*. Society for Industrial and Applied Mathematics, Philadelphia, PA. doi:10.1137/1.9781611977950. https://doi.org/10.1137/1.97816119779

Bernardi M, Busatto C, Cattelan M (2024). "Fast QR updating methods for statistical applications." 2412.05905, https://arxiv.org/abs/2412.05905.

Examples

```
## Add one column
## generate sample data
set.seed(1234)
n <- 12
p <- 5
X <- matrix(rnorm(n * p, 1), n, p)

## get the initial QR factorization
output <- fastQR::qr(X, complete = TRUE)
Q      <- output$Q
R      <- output$R
R1     <- R[1:p,]

## create column to be added
u  <- matrix(rnorm(n), n, 1)
X1 <- cbind(X, u)

## update the R decomposition
R2 <- fastQR::rupdate(X = X, R = R1, U = u,
                      fast = FALSE, type = "column")

## check
max(abs(crossprod(R2) - crossprod(X1)))

## Add m columns
## generate sample data
set.seed(1234)
n <- 10
p <- 5
X <- matrix(rnorm(n * p, 1), n, p)

## get the initial QR factorization
output <- fastQR::qr(X, complete = TRUE)
Q      <- output$Q
R      <- output$R
R1     <- R[1:p,]

## create the matrix of columns to be added
m  <- 2
U  <- matrix(rnorm(n*m), n, m)
X1 <- cbind(X, U)

# QR update
R2 <- fastQR::rupdate(X = X, R = R1, U = U,
                      fast = FALSE, type = "column")

## check
max(abs(crossprod(R2) - crossprod(X1)))

## Add one row
## generate sample data
```

```
set.seed(1234)
n <- 12
p <- 5
X <- matrix(rnorm(n * p, 1), n, p)

## get the initial QR factorization
output <- fastQR::qr(X, complete = TRUE)
Q      <- output$Q
R      <- output$R
R1     <- R[1:p,]

## create the row u to be added
u  <- matrix(data = rnorm(p), p, 1)
k  <- 5
if (k<=n) {
  X1 <- rbind(rbind(X[1:(k-1), ], t(u)), X[k:n, ])
} else {
  X1 <- rbind(rbind(X, t(u)))
}

## update the R decomposition
R2 <- fastQR::rupdate(R = R1, X = X,
                      U = u,
                      type = "row")

## check
max(abs(crossprod(R2) - crossprod(X1)))

## Add m rows
## generate sample data
set.seed(1234)
n <- 12
p <- 5
X <- matrix(rnorm(n * p, 1), n, p)

## get the initial QR factorization
output <- fastQR::qr(X, complete = TRUE)
Q      <- output$Q
R      <- output$R
R1     <- R[1:p,]

## create the matrix of rows to be added
m  <- 2
U  <- matrix(data = rnorm(p*m), p, m)
k  <- 5
if (k<=n) {
  X1 <- rbind(rbind(X[1:(k-1), ], t(U)), X[k:n, ])
} else {
  X1 <- rbind(rbind(X, t(U)))
}

## update the R decomposition
R2 <- fastQR::rupdate(R = R1, X = X,
```

```
                        U = U,
                        fast = FALSE,
                        type = "row")

## check
max(abs(crossprod(R2) - crossprod(X1)))
```

Index

qr, 2
qr_coef, 4
qr_fast, 5
qr_fitted, 8
qr_lapack, 10
qr_lm, 11
qr_lse_coef, 12
qr_lse_fitted, 13
qr_lse_Qty, 14
qr_lse_Qy, 15
qr_lse_resid, 16
qr_pivot2perm, 17
qr_Q, 18
qr_Q_full, 20
qr_Q_reduced2full, 21
qr_Qty, 22
qr_Qy, 23
qr_R, 24
qr_resid, 26
qr_thin, 27
qr_X, 28
qrchol, 30
qrdowndate, 30
qrls, 34
qrmls, 35
qrmridge, 37
qrmridge_cv, 38
qrridge, 41
qrridge_cv, 43
qrsolve, 45
qrupdate, 46

rchol, 49
rdowndate, 51
rupdate, 54