Package 'fastQR'

November 10, 2026

Type Package

Title Fast QR Decomposition and Update

Version 1.1.5

Date 2026-11-10

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

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

Description Efficient algorithms for performing, updating, and removing rows or columns from the QR decomposition, R decomposition, or the inverse of the R decomposition of a matrix as rows or columns are added or removed. It also includes functions for solving linear systems of equations, normal equations for linear regression models, and normal equations for linear regression with a RIDGE penalty. For a detailed introduction to these methods, the monograph Matrix Computations (2013, <doi:10.1007/978-3-319-05089-8>) for complete introduction to the methods.

License GPL (>= 2)

Imports Rcpp (>= 1.0.10), RcppEigen, Rdpack

LinkingTo Rcpp, RcppArmadillo, RcppEigen

Encoding UTF-8

RoxygenNote 7.3.3

RdMacros Rdpack

SystemRequirements GNU make

NeedsCompilation yes

Repository CRAN

Date/Publication 2026-11-10 09:40:02 UTC

Contents

qr 2
qrchol 4

qrdowndate 4
qrls 8
qrmls 9
qrmridge 10
qrmridge_cv 12
qrridge 13
qrridge_cv 15
qrsolve 16
qrupdate 17
qr_coef 21
qr_fast 22
qr_fitted 25
qr_lm 26
qr_lse_coef 27
qr_lse_fitted 28
qr_lse_Qty 29
qr_lse_Qy 30
qr_lse_resid 31
qr_pivot2perm 32
qr_Q 33
qr_Qty 34
qr_Qy 35
qr_Q_full 36
qr_Q_reduced2full 37
qr_R 39
qr_resid 40
qr_thin 41
qr_X 42
rchol 44
rdowndate 45
rupdate 48

Index **52**

`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. Defines the number of block in the block recursive QR decomposition. See Golud and van Loan (2013).
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 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))
```

---

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 $X^\top X$.

---

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 $R_2 \in \mathbb{R}^{(n-p)\times p}$. 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

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

Arguments

Q	a $n \times n$ matrix.
R	a $n \times p$ upper triangular matrix.
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.

Value

A named list containing

Q the updated Q matrix.

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

```
## 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.

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 *RIDGE estimator for the linear multivariate regression model*

Description

qrmridge, 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

```
qrmridge(Y, X, lambda, 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.
lambda	a vector of lambdas.
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::qrmridge(Y = Y, X = X, lambda = 1, X_test = X_test, type = "QR")
output$coeff
```

`qrmridge_cv` *Cross-validation of the RIDGE estimator for the linear multivariate regression model*

Description

qrmridge_cv, 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

```
qrmridge_cv(Y, X, lambda, k = NULL, seed = NULL, 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.
`lambda`	a vector of lambdas.
`k`	an integer vector defining the number of groups for CV.
`seed`	ad integer number defining the seed for random number generation.
`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::qrmridge_cv(Y = Y, X = X, lambda = c(1,2),
                                 k = 5, seed = 12, X_test = X_test, type = "QR")
output$coeff
```

---

qrridge *RIDGE estimation for the linear regression model*

Description

lmridge, or RIDGE for linear regression models, solves the following penalized optimization problem

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

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

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

Arguments

y	a vector of length-n response vector.
X	an $(n \times p)$ matrix of predictors.
lambda	a vector of lambdas.
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

mean_y mean of the response variable.

mean_X a length-p vector containing the mean of each column of the design matrix.

path the whole path of estimated regression coefficients.

ess explained sum of squares for the whole path of estimated coefficients.

GCV generalized cross-validation for the whole path of lambdas.

GCV_min minimum value of GCV.

GCV_idx inded corresponding to the minimum values of GCV.

coeff a length-p vector containing the solution for the parameters β which corresponds to the minimum of GCV.

lambda the vector of lambdas.

scales the vector of standard deviations of each column of the design matrix.

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 of the RIDGE estimator for the linear regression model*

Description

qrridge_cv, 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

```
qrridge_cv(y, X, lambda, k = NULL, seed = NULL, 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.
lambda	a vector of lambdas.
k	an integer vector defining the number of groups for CV.
seed	ad integer number defining the seed for random number generation.
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 the matrix $X^\top X$.

XTy $X^\top y$.

sigma_hat estimated residual variance.

df degrees of freedom.

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

R 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.

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.

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))
```

---

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.

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.

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 $\hat{y}$.

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_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.

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.

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 $\hat{y}$.

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 $Q^\top y$.

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.

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.

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 $r = \text{rank}(X) < p$, 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 complete is TRUE, Q has n columns. If complete is FALSE, Q has p columns.

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_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 $Q^\top y$.

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.

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_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 $Q \in \mathbb{R}^{n \times n}$.

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 $n > p$).

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

Description

returns the R matrix of the full QR decomposition. If $r = \text{rank}(X) < p$, 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 complete is TRUE, R has n rows. If complete is FALSE, R has p rows.

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.

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.

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))
```

`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.

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, 21
qr_fast, 22
qr_fitted, 25
qr_lm, 26
qr_lse_coef, 27
qr_lse_fitted, 28
qr_lse_Qty, 29
qr_lse_Qy, 30
qr_lse_resid, 31
qr_pivot2perm, 32
qr_Q, 33
qr_Q_full, 36
qr_Q_reduced2full, 37
qr_Qty, 34
qr_Qy, 35
qr_R, 39
qr_resid, 40
qr_thin, 41
qr_X, 42
qrchol, 4
qrdowndate, 4
qrls, 8
qrmls, 9
qrmridge, 10
qrmridge_cv, 12
qrridge, 13
qrridge_cv, 15
qrsolve, 16
qrupdate, 17

rchol, 44
rdowndate, 45
rupdate, 48