

# Package: RSO (via r-universe)

July 6, 2026

**Title** Ridge Selection Operator for Sparse Linear Regression

**Version** 1.0.0

**Description** Implements the Ridge Selection Operator (RSO) for variable selection in linear regression as proposed by Wu (2021) [doi:10.1080/00401706.2020.1791254](https://doi.org/10.1080/00401706.2020.1791254). The RSO method extends classical ridge regression by using individually penalized ridge parameters, inducing sparsity through reciprocal penalty parameters. This package provides a fast C++ implementation ('RSOFast') using 'Armadillo' linear algebra routines. The fast implementation precomputes matrix products, uses Cholesky factorization with primal/dual switching, and performs golden-section search for coordinate optimization.

**License** GPL (>= 3)

**Encoding** UTF-8

**RoxygenNote** 7.3.2

**LinkingTo** Rcpp, RcppArmadillo

**Imports** Rcpp

**NeedsCompilation** yes

**Author** Murat Genc [aut, cre], Adewale Lukman [aut]

**Maintainer** Murat Genc <mgenc@cu.edu.tr>

**Repository** <https://cran.r-universe.dev>

**Date/Publication** 2026-07-06 14:39:12 UTC

**RemoteUrl** <https://github.com/cran/RSO>

**RemoteRef** HEAD

**RemoteSha** 9de59d1f429827a96e5ce12f147b30fb1af11323

## Contents

ridgereg . . . . .	2
ridgereg_df . . . . .	2
ridgeRegPrecomp . . . . .	3
RSOFast . . . . .	4

---

ridgereg	<i>Individually Penalized Ridge Regression</i>
----------	--

---

**Description**

Computes the sum of squared residuals (SSR) for individually penalized ridge regression without explicitly forming the hat matrix. Uses Cholesky decomposition for speed and numerical stability.

**Usage**

```
ridgereg(x, y, gam, penalty.factor = rep(1, length(gam)))
```

**Arguments**

x	Centered predictor matrix (n x p)
y	Centered response vector (n x 1)
gam	Ridge penalty parameters vector (p x 1) where $\text{gam}_j = 1/\text{nu}_j$
penalty.factor	Adaptive penalty factors (p x 1), default = 1 for RSO

**Value**

Sum of squared residuals (SSR)

**Examples**

```
n <- 100; p <- 10
x <- matrix(rnorm(n*p), n, p)
x <- scale(x, scale = FALSE)
y <- rnorm(n)
y <- y - mean(y)
gam <- runif(p)
result <- ridgereg(x, y, gam, rep(1, p))
```

---

ridgereg_df	<i>Ridge Regression with Degrees of Freedom</i>
-------------	---

---

**Description**

Computes the sum of squared residuals (SSR), degrees of freedom (trace of hat matrix), and coefficient estimates for ridge regression. Uses Cholesky decomposition for speed and numerical stability.

**Usage**

```
ridgereg_df(x, y, gam, penalty.factor = rep(1, length(gam)))
```

**Arguments**

`x`                    Centered predictor matrix (n x p)  
`y`                    Centered response vector (n x 1)  
`gam`                  Ridge penalty parameters vector (p x 1) where  $\text{gam}_j = 1/\text{nu}_j$   
`penalty.factor`      Adaptive penalty factors (p x 1), default = 1 for RSO

**Value**

A list containing:

`ssr`                  Sum of squared residuals  
`df`                    Degrees of freedom (trace of hat matrix)  
`coef`                 Coefficient estimates (p x 1)

**Examples**

```
n <- 100; p <- 10
x <- matrix(rnorm(n*p), n, p)
x <- scale(x, scale = FALSE)
y <- rnorm(n)
y <- y - mean(y)
gam <- runif(p)
result <- ridgereg_df(x, y, gam, rep(1, p))
```

---

ridgeRegPrecomp

*Individually Penalized Ridge Regression with Precomputed Matrices*


---

**Description**

Computes the sum of squared residuals (SSR) for individually penalized ridge regression using precomputed  $X'X$  and  $X'y$  matrices. This is optimized for repeated calls with the same  $x$  matrix but different  $\text{gam}$  parameters (e.g., during grid search).

**Usage**

```
ridgeRegPrecomp(x, y, gam, penalty.factor = rep(1, length(gam)), precomp)
```

**Arguments**

<code>x</code>	Centered predictor matrix (n x p)
<code>y</code>	Centered response vector (n x 1)
<code>gam</code>	Ridge penalty parameters vector (p x 1) where $\text{gam}_j = 1/\text{nu}_j$
<code>penalty.factor</code>	Adaptive penalty factors (p x 1), default = 1 for RSO
<code>precomp</code>	List containing precomputed values: <ul style="list-style-type: none"> <li>• <code>xtx</code>: <math>X'X</math> matrix (p x p)</li> <li>• <code>xty</code>: <math>X'y</math> vector (p x 1)</li> </ul>

**Value**

Sum of squared residuals (SSR)

**Examples**

```
n <- 100; p <- 10
x <- matrix(rnorm(n*p), n, p)
x <- scale(x, scale = FALSE)
y <- rnorm(n)
y <- y - mean(y)
precomp <- list(xtx = crossprod(x), xty = crossprod(x,y))
gam <- runif(p)
result <- ridgeRegPrecomp(x, y, gam, rep(1, p), precomp)
```

---

 RSOFast

*Fast RSO (Ridge Selection Operator)*


---

**Description**

Implements the modified coordinate descent algorithm for RSO as described in Wu (2021). Solves for optimal lambda parameters that minimize SSR subject to  $\text{sum}(\text{lambda}) = \text{tau}$  and  $\text{lambda} \geq 0$ .

**Usage**

```
RSOFast(x, y, tau, penalty.factor = rep(1, ncol(x)), gaminit = NULL)
```

**Arguments**

<code>x</code>	Centered predictor matrix (n x p)
<code>y</code>	Centered response vector (n x 1)
<code>tau</code>	Regularization parameter (sum of lambda)
<code>penalty.factor</code>	Adaptive penalty factors (p x 1), default = 1 for RSO
<code>gaminit</code>	Initial lambda values. If NULL, uses $\text{tau}/p$ for all.

**Value**

A list containing:

gamma	Optimal lambda vector (p x 1)
lambda	Alias for gamma
coefficients	Coefficient estimates (p x 1)
coef	Alias for coefficients
df	Degrees of freedom
ssr	Sum of squared residuals
iterations	Number of iterations
converged	Convergence status
n_selected	Number of selected variables
tau	Original tau parameter

**References**

Wu, Y. (2021). Can't ridge regression perform variable selection?. *Technometrics*, 63(2), 263-271.  
[doi:10.1080/00401706.2020.1791254](https://doi.org/10.1080/00401706.2020.1791254)

**Examples**

```
n <- 100; p <- 10
x <- matrix(rnorm(n*p), n, p)
x <- scale(x, scale = FALSE)
y <- rnorm(n)
y <- y - mean(y)
result <- RSOFast(x, y, tau = 1.0, rep(1, p))
```

# Index

ridgereg, [2](#)  
ridgereg\_df, [2](#)  
ridgeRegPrecomp, [3](#)  
RSOFast, [4](#)