

# Package ‘rminqa’

May 9, 2026

**Type** Package

**Title** Derivative-Free Optimization in R using C++

**Version** 0.3.1

**Date** 2026-01-11

**Description** Perform derivative-free optimization algorithms in R using C++.

A wrapper interface is provided to call C function of the 'bobyqa' implementation  
(See <<https://github.com/emmt/Algorithms/tree/master/bobyqa>>).

**License** GPL (>= 2)

**Encoding** UTF-8

**Imports** Rcpp (>= 1.0.7)

**LinkingTo** Rcpp (>= 1.0.7)

**Depends** R (>= 3.5.0)

**RoxygenNote** 7.3.2

**Suggests** minqa

**NeedsCompilation** yes

**Author** Sam Watson [aut, cre],

Yi Pan [aut],

Éric Thiébaud [aut],

Mike Powell [aut]

**Maintainer** Sam Watson <S.I.Watson@bham.ac.uk>

**Repository** CRAN

**Date/Publication** 2026-01-12 06:10:21 UTC

## Contents

bobyqa_rosen_test1 . . . . .	2
bobyqa_rosen_test2 . . . . .	2
bobyqa_rosen_x1 . . . . .	3
bobyqa_rosen_x1e . . . . .	4

<b>Index</b>	<b>5</b>
--------------	----------

---

bobyqa\_rosen\_test1      *Example 0a: Minimize Rosenbrock function using bobyqa*

---

### Description

Minimize Rosenbrock function using bobyqa and expect a normal exit from bobyqa.

### Usage

```
bobyqa_rosen_test1()
```

### Value

No return value, called for side effects.

### Examples

```
fr <- function(x) { ## Rosenbrock Banana function
  x1 <- x[1]
  x2 <- x[2]
  100 * (x2 - x1 * x1)^2 + (1 - x1)^2
}
(x1 <- minqa::bobyqa(c(1, 2), fr, lower = c(0, 0), upper = c(4, 4)))
## => optimum at c(1, 1) with fval = 0
str(x1) # see that the error code and msg are returned

## corresponding C++ implementation:
bobyqa_rosen_test1()
```

---

bobyqa\_rosen\_test2      *Example 0b: Minimize Rosenbrock function using bobyqa*

---

### Description

Minimize Rosenbrock function using bobyqa and expect a normal exit from bobyqa. Allows for user input.

### Usage

```
bobyqa_rosen_test2(start, lower, upper)
```

### Arguments

start	Starting values of the algorithm.
lower	Lower bounds of the parameters.
upper	Upper bounds of the parameters.

**Value**

No return value, called for side effects.

**Examples**

```
fr <- function(x) { ## Rosenbrock Banana function
  x1 <- x[1]
  x2 <- x[2]
  100 * (x2 - x1 * x1)^2 + (1 - x1)^2
}
(x1 <- minqa::bobyqa(c(1, 2), fr, lower = c(0, 0), upper = c(4, 4)))
## => optimum at c(1, 1) with fval = 0
str(x1) # see that the error code and msg are returned

## corresponding C++ implementation:
bobyqa_rosen_test2(c(1,2),c(0,0),c(4,4))
```

---

bobyqa\_rosen\_x1

*Example 1a: Minimize Rosenbrock function using bobyqa*


---

**Description**

Minimize Rosenbrock function using bobyqa and expect a normal exit from bobyqa.

**Usage**

```
bobyqa_rosen_x1()
```

**Value**

No return value, called for side effects.

**Examples**

```
fr <- function(x) { ## Rosenbrock Banana function
  x1 <- x[1]
  x2 <- x[2]
  100 * (x2 - x1 * x1)^2 + (1 - x1)^2
}
(x1 <- minqa::bobyqa(c(1, 2), fr, lower = c(0, 0), upper = c(4, 4)))
## => optimum at c(1, 1) with fval = 0
str(x1) # see that the error code and msg are returned

## corresponding C++ implementation:
bobyqa_rosen_x1()
```

---

`bobyqa_rosen_x1e`*Example 1b: Minimize Rosenbrock function using bobyqa*

---

**Description**

Minimize Rosenbrock function using bobyqa and expect a normal exit from bobyqa.

**Usage**

```
bobyqa_rosen_x1e()
```

**Value**

No return value, called for side effects.

**Examples**

```
fr <- function(x) { ## Rosenbrock Banana function
  x1 <- x[1]
  x2 <- x[2]
  100 * (x2 - x1 * x1)^2 + (1 - x1)^2
}
# check the error exits
# too many iterations
x1e <- minqa::bobyqa(c(1, 2), fr, lower = c(0, 0), upper = c(4, 4), control = list(maxfun=50))
str(x1e)

## corresponding C++ implementation:
bobyqa_rosen_x1e()
```

# Index

bobyqa\_rosen\_test1, [2](#)

bobyqa\_rosen\_test2, [2](#)

bobyqa\_rosen\_x1, [3](#)

bobyqa\_rosen\_x1e, [4](#)