Package 'lbfgsb3c'

October 6, 2026

Type Package

Title Limited Memory BFGS Minimizer with Bounds on Parameters with optim() 'C' Interface

Version 2026-3.5

Maintainer Matthew L Fidler <matthew.fidler@gmail.com>

Description Interfacing to Nocedal et al. L-BFGS-B.3.0 (See <http://users.iems.northwestern.edu/~nocedal/lbfgsb.html>) limited memory BFGS minimizer with bounds on parameters.
This is a fork of 'lbfgsb3'.
This registers a 'R' compatible 'C' interface to L-BFGS-B.3.0 that uses the same function types and optimization as the optim() function (see writing 'R' extensions and source for details). This package also adds more stopping criteria as well as allowing the adjustment of more tolerances.

License GPL-2

Depends R (>= 3.6)

Imports Rcpp (>= 0.12.3), numDeriv, methods

Suggests knitr, microbenchmark, optimx, RcppArmadillo, rmarkdown, testthat (>= 3.0.0), adagio

LinkingTo Rcpp (>= 0.12.3), RcppArmadillo

Encoding UTF-8

VignetteBuilder knitr

NeedsCompilation yes

URL https://nlmixr2.github.io/lbfgsb3c/, https://github.com/nlmixr2/lbfgsb3c

BugReports https://github.com/nlmixr2/lbfgsb3c/issues

Config/testthat/edition 3

Config/roxygen2/version 8.1.0

Author Matthew L Fidler [aut, cre] (ORCID: <https://orcid.org/0000-0001-8538-6691>), John C Nash [aut],

Ciyou Zhu [aut],
Richard Byrd [aut],
Jorge Nocedal [aut],
Jose Luis Morales [aut]

Repository CRAN

Date/Publication 2026-10-06 19:40:08 UTC

Contents

.lbfgsb3cPtr 2
lbfgsb3c 3

Index **7**

`.lbfgsb3cPtr` *Get the lbfgsb3c pointer*

Description

Get the lbfgsb3c pointer

Usage

```
.lbfgsb3cPtr()
```

Value

lbfgsb3c function pointer

Author(s)

Matthew L. Fidler

Examples

```
.lbfgsb3cPtr()
```

lbfgsb3c *Interfacing wrapper for the Nocedal - Morales LBFGSB3 (Fortran) limited memory BFGS solver.*

Description

Interfacing wrapper for the Nocedal - Morales LBFGSB3 (Fortran) limited memory BFGS solver.

Usage

```
lbfgsb3c(
  par,
  fn,
  gr = NULL,
  lower = -Inf,
  upper = Inf,
  control = list(),
  ...,
  rho = NULL
)

lbfgsb3(
  par,
  fn,
  gr = NULL,
  lower = -Inf,
  upper = Inf,
  control = list(),
  ...,
  rho = NULL
)

lbfgsb3f(
  par,
  fn,
  gr = NULL,
  lower = -Inf,
  upper = Inf,
  control = list(),
  ...,
  rho = NULL
)

lbfgsb3x(
  par,
  fn,
  gr = NULL,
```

```
  lower = -Inf,
  upper = Inf,
  control = list(),
  ...,
  rho = NULL
)
```

Arguments

par	A parameter vector which gives the initial guesses to the parameters that will minimize fn. This can be named, for example, we could use par=c(b1=1, b2=2.345, b3=0.123)
fn	A function that evaluates the objective function to be minimized. This can be a R function or a Rcpp function pointer.
gr	If present, a function that evaluates the gradient vector for the objective function at the given parameters computing the elements of the sum of squares function at the set of parameters start. This can be a R function or a Rcpp function pointer.
lower	Lower bounds on the parameters. If a single number, this will be applied to all parameters. Default -Inf.
upper	Upper bounds on the parameters. If a single number, this will be applied to all parameters. Default Inf.
control	An optional list of control settings. See below in details.
...	Any data needed for computation of the objective function and gradient.
rho	An Environment to use for function evaluation. If present the arguments in ... are ignored. Otherwise the ... are converted to an environment for evaluation.

Details

See the notes below for a general appreciation of this package.

The control list can contain:

- trace If positive, tracing information on the progress of the optimization is produced. Higher values may produce more tracing information: for method "L-BFGS-B" there are six levels of tracing. (To understand exactly what these do see the source code: higher levels give more detail.)
- factr controls the convergence of the "L-BFGS-B" method. Convergence occurs when the reduction in the objective is within this factor of the machine tolerance. Default is 1e7, that is a tolerance of about 1e-8.
- pgtol helps control the convergence of the "L-BFGS-B" method. It is a tolerance on the projected gradient in the current search direction. This defaults to zero, when the check is suppressed.
- abstol helps control the convergence of the "L-BFGS-B" method. It is an absolute tolerance difference in x values. This defaults to zero, when the check is suppressed.
- reltol helps control the convergence of the "L-BFGS-B" method. It is an relative tolerance difference in x values. This defaults to zero, when the check is suppressed.

- lmm is an integer giving the number of BFGS updates retained in the "L-BFGS-B" method, It defaults to 5.
- maxit maximum number of iterations.
- iprint Provided only for compatibility with older codes. This control is no longer active.)
- info a boolean to indicate if more optimization information is captured and output in a $info list
- engine which implementation of L-BFGS-B to use: `"fortran"` (the default) is the original Fortran code; `"cpp"` is a thread-safe C++ port of the same code (also available to C/C++ callers as `lbfgsb3Cts` through `.lbfgsb3cPtr()` and `lbfgsb3ptr.h`).

Value

A list of the following items

- par The best set of parameters found.
- value The value of fn corresponding to par.
- counts A two-element integer vector giving the number of calls to fn and gr respectively. This excludes any calls to fn to compute a finite-difference approximation to the gradient.
- convergence An integer code. 0 indicates successful completion

Note

This package is a wrapper to the Fortran code released by Nocedal and Morales. This poses several difficulties for an R package. While the `.Fortran()` tool exists for the interfacing, we must be very careful to align the arguments with those of the Fortran subroutine, especially in type and storage.

A more annoying task for interfacing the Fortran code is that Fortran WRITE or PRINT statements must all be replaced with calls to special R-friendly output routines. Unfortunately, the Fortran is full of output statements. Worse, we may wish to be able to suppress such output, and there are thus many modifications to be made. This means that an update of the original code cannot be simply plugged into the R package `src` directory.

Finally, and likely because L-BFGS-B has a long history, the Fortran code is far from well-structured. For example, the number of function and gradient evaluations used is returned as the 34'th element of an integer vector. There does not appear to be an easy way to stop the program after some maximum number of such evaluations have been performed.

On the other hand, the version of L-BFGS-B in `optim()` is a `C` translation of a now-lost Fortran code. It does not implement the improvements Nocedal and Morales published in 2011. Hence, despite its deficiencies, this wrapper has been prepared.

In addition to the above reasons for the original lbfgsb3 package, this additional package allows C calling of L-BFGS-B 3.0 by a program as well as adjustments to the tolerances that were not present in the original CRAN package. Also adjustments were made to have outputs conform with R's optim routine.

Author(s)

Matthew Fidler (move to C and add more options for adjustments), John C Nash <nashjc@uottawa.ca> (of the wrapper and edits to Fortran code to allow R output) Ciyou Zhu, Richard Byrd, Jorge Nocedal, Jose Luis Morales (original Fortran packages)

References

Morales, J. L.; Nocedal, J. (2011). "Remark on 'algorithm 778: L-BFGS-B: Fortran subroutines for large-scale bound constrained optimization' ". ACM Transactions on Mathematical Software 38: 1.

Byrd, R. H.; Lu, P.; Nocedal, J.; Zhu, C. (1995). "A Limited Memory Algorithm for Bound Constrained Optimization". SIAM J. Sci. Comput. 16 (5): 1190-1208.

Zhu, C.; Byrd, Richard H.; Lu, Peihuang; Nocedal, Jorge (1997). "L-BFGS-B: Algorithm 778: L-BFGS-B, FORTRAN routines for large scale bound constrained optimization". ACM Transactions on Mathematical Software 23 (4): 550-560.

See Also

Packages optim and optimx.

Examples

```
# Rosenbrock's banana function
n=3; p=100

fr = function(x)
{
    f=1.0
    for(i in 2:n) {
        f=f+p*(x[i]-x[i-1]**2)**2+(1.0-x[i])**2
    }
    f
}

grr = function(x)
{
    g = double(n)
    g[1]=-4.0*p*(x[2]-x[1]**2)*x[1]
    if(n>2) {
        for(i in 2:(n-1)) {
            g[i]=2.0*p*(x[i]-x[i-1]**2)-4.0*p*(x[i+1]-x[i]**2)*x[i]-2.0*(1.0-x[i])
        }
    }
    g[n]=2.0*p*(x[n]-x[n-1]**2)-2.0*(1.0-x[n])
    g
}
x = c(a=1.02, b=1.02, c=1.02)
(opc <- lbfgsb3c(x,fr, grr))
(op <- lbfgsb3(x,fr, grr, control=list(trace=1)))
(opx <- lbfgsb3x(x,fr, grr))
(opf <- lbfgsb3f(x,fr, grr))
```

Index

∗ **nonlinear**
    lbfgsb3c, 3
∗ **optimization**
    lbfgsb3c, 3
∗ **parameter**
    lbfgsb3c, 3
.lbfgsb3cPtr, 2

lbfgsb3 *(*lbfgsb3c*)*, 3
lbfgsb3c, 3
lbfgsb3f *(*lbfgsb3c*)*, 3
lbfgsb3x *(*lbfgsb3c*)*, 3

optim, *6*