

Package: RcppTrust (via r-universe)

September 17, 2026

Type Package

Title Thread-Safe C++ Trust Region Optimizers

Version 0.2.0

Description Thread-safe C++ trust region optimizers, each exposed both as an R function and as a C-callable entry point (with function pointers registered for other packages to call without linking against this package's shared library) so they can be used from parallel C++ code such as in 'nlmixr2est'. Includes a port of the two-derivative trust region algorithm in the 'trust' package by Charles J. Geyer; a port of the Steihaug truncated conjugate-gradient trust region method from the 'basin' Rust crate (Steihaug (1983) <[doi:10.1137/0720042](https://doi.org/10.1137/0720042)>); and ports of M. J. D. Powell's derivative-free BOBYQA (Powell (2009), Report DAMTP 2009/NA06) and NEWUOA (Powell (2006) <[doi:10.1007/0-387-30065-1_16](https://doi.org/10.1007/0-387-30065-1_16)>) Fortran 77 code <<https://github.com/libprima/prima/tree/main/fortran/original>>, as modified and distributed in the 'minqa' package, whose R interface is reproduced.

License GPL-2

Copyright See inst/COPYRIGHTS for the sources, copyright holders and licenses of code ported from 'trust', 'basin', 'minqa' and Powell's Fortran.

URL <https://github.com/nlmixr2/RcppTrust/>,
<https://nlmixr2.github.io/RcppTrust/>,
<http://www.stat.umn.edu/geyer/trust/>

Encoding UTF-8

Depends R (>= 4.2.0)

Imports Rcpp

LinkingTo Rcpp, RcppArmadillo, BH

Suggests testthat (>= 3.0.0), stats, knitr, rmarkdown, trust, minqa, microbenchmark

Config/testthat/edition 3

SystemRequirements C++17
VignetteBuilder knitr
Config/roxygen2/version 8.1.0
Repository <https://nlmixr2.r-universe.dev>
Date/Publication 2026-09-17 21:21:07 UTC
RemoteUrl <https://github.com/nlmixr2/RcppTrust>
RemoteRef HEAD
RemoteSha 38fd778c4f7622c370163edba4d3781343e360dc

Contents

.RcppTrustPtr	2
bobyqa	3
newuoa	4
print.minqa	5
steihaug	5
trust	7
Index	9

.RcppTrustPtr	<i>Positionally-indexed function pointer table for downstream packages</i>
---------------	--

Description

Returns the list of native function pointers a consuming package (e.g. `nlmixr2est`) resolves via the header-only registration pattern in `inst/include/RcppTrust.h` – the same pattern `rxode2/n1qn1c/lbfgsb3c` use, so no linker dependency on this package’s shared library is required. Not intended to be called directly by end users.

Usage

`.RcppTrustPtr()`

Value

An unclassified list of external pointers.

Description

Thread-safe C++ port of `minqa::bobyqa()`: minimizes a function of many variables subject to box constraints by a trust region method that forms quadratic models by interpolation, using M. J. D. Powell's BOBYQA algorithm. Arguments, defaults, and return value are those of `minqa::bobyqa()`; see its documentation for details.

Usage

```
bobyqa(par, fn, lower = -Inf, upper = Inf, control = list(), ...)
```

Arguments

<code>par</code>	numeric vector of starting parameters.
<code>fn</code>	function to be minimized; its first argument must be the parameter vector and it must return a scalar numeric value.
<code>lower, upper</code>	numeric vectors of lower and upper bounds (recycled when of length 1).
<code>control</code>	a list of control settings: <code>npt</code> , <code>rhobeg</code> , <code>rhoend</code> , <code>iprint</code> , <code>maxfun</code> , <code>obstop</code> and <code>force.start</code> , as in <code>minqa::bobyqa()</code> .
<code>...</code>	further arguments passed to <code>fn</code> .

Value

A list of class `c("bobyqa", "minqa")` with components `par`, `fval`, `feval`, `ierr` and `msg`.

Source

M. J. D. Powell's original Fortran 77 BOBYQA code, archived at <https://github.com/libprima/prima/tree/main/fortran/original/bobyqa>; R interface following the **minqa** package.

References

M. J. D. Powell (2009), "The BOBYQA algorithm for bound constrained optimization without derivatives", Report No. DAMTP 2009/NA06, Centre for Mathematical Sciences, University of Cambridge.

See Also

[`newuoa()`], and [`minqa_c_api`] for calling the solvers from parallel C/C++ code.

Examples

```
fr <- function(x) 100 * (x[2] - x[1]^2)^2 + (1 - x[1])^2
bobyqa(c(1, 2), fr, lower = c(0, 0), upper = c(4, 4))
```

newuoa

*Unconstrained derivative-free optimization (NEWUOA)***Description**

Thread-safe C++ port of `minqa::newuoa()`: minimizes a function of many variables by a trust region method that forms quadratic models by interpolation, using M. J. D. Powell's NEWUOA algorithm.

Usage

```
newuoa(par, fn, control = list(), ...)
```

Arguments

<code>par</code>	numeric vector of starting parameters.
<code>fn</code>	function to be minimized; its first argument must be the parameter vector and it must return a scalar numeric value.
<code>control</code>	a list of control settings: <code>npt</code> , <code>rhobeg</code> , <code>rhoend</code> , <code>iprint</code> and <code>maxfun</code> , as in <code>minqa::newuoa()</code> .
<code>...</code>	further arguments passed to <code>fn</code> .

Value

A list of class `c("newuoa", "minqa")` with components `par`, `fval`, `feval`, `ierr` and `msg`.

Source

M. J. D. Powell's original Fortran 77 NEWUOA code, archived at <https://github.com/libprima/prima/tree/main/fortran/original/newuoa>; R interface following the **minqa** package.

References

M. J. D. Powell (2006), "The NEWUOA software for unconstrained optimization without derivatives", in *Large-Scale Nonlinear Optimization*, Springer, 255-297. doi:10.1007/0387300651_16

See Also

`[bobyqa()]`, `[minqa_c_api]`

Examples

```
fr <- function(x) 100 * (x[2] - x[1]^2)^2 + (1 - x[1])^2
newuoa(c(1, 2), fr)
```

print.minqa	<i>Print method for minqa objects</i>
-------------	---------------------------------------

Description

Print method for minqa objects

Usage

```
## S3 method for class 'minqa'
print(x, digits = max(3, getOption("digits") - 3), ...)
```

Arguments

x	an object inheriting from class minqa.
digits	unused; kept for compatibility with minqa.
...	unused.

Value

invisible(x)

steihaug	<i>Steihaug truncated-CG trust-region Newton minimization</i>
----------	---

Description

Thread-safe C++ port of the trust-region Newton minimizer with the Steihaug truncated conjugate-gradient subproblem from the Rust crate **basin** (TrustRegion with its default Steihaug subproblem, driven by basin's executor). Each iteration approximately minimizes the quadratic model $m(p) = f + g^T p + \frac{1}{2} p^T B p$ over $\|p\| \leq \Delta$ by conjugate gradients, stopping at the boundary on negative curvature. Unlike [trust()] it never factorizes the Hessian, and with hessvec it never forms it at all (basin's matrix-free mode).

Usage

```
steihaug(par, fn, gr, hess = NULL, hessvec = NULL, control = list(), ...)
```

Arguments

<code>par</code>	numeric vector of starting parameters.
<code>fn</code>	objective function of the parameter vector returning a scalar.
<code>gr</code>	gradient function returning a numeric vector of length <code>length(par)</code> .
<code>hess</code>	Hessian function returning a symmetric <code>length(par)</code> by <code>length(par)</code> matrix. May be NULL when <code>hessvec</code> is supplied.
<code>hessvec</code>	optional Hessian-vector product function(<code>x</code> , <code>v</code>) returning $H(x)v$. When supplied, the matrix-free mode is used and <code>hess</code> is never called.
<code>control</code>	a list of control settings (defaults in brackets; a NULL tolerance or budget is disabled): rinit initial trust region radius [1] rmax maximum trust region radius [100] eta step acceptance threshold on the reduction ratio, in $[0, 1/4)$ [0.125] maxInner radius reductions (re-solves) per outer iteration [10] cgMaxit CG iteration cap per subproblem; 0 means <code>length(par)</code> [0] kappa, theta CG forcing rule: stop when $\ r\ < \min(\kappa, \ g\ ^\theta)\ g\ $ [0.5, 0.5] maxit maximum outer iterations [100] maxCostEvals, maxGradEvals evaluation budgets [NULL] targetCost stop once the objective is at or below this [NULL] gradTol absolute gradient tolerance on $\ g\ _2$ [1e-8] relGradTol gradient tolerance relative to the initial gradient norm [NULL] stepTol, relStepTol absolute / relative step tolerance [NULL] costTol, relCostTol absolute / relative objective change tolerance [NULL] trace if TRUE, return a per-attempt trace [FALSE] ... further arguments passed to <code>fn</code> , <code>gr</code> , <code>hess</code> and <code>hessvec</code> .

Details

Evaluation pattern (identical to `basin`): `fn` and `gr` once at the start; `hess` once per outer iteration (exact mode) or `hessvec` once per CG product (matrix-free mode); `fn` at each trial point; `gr` after each accepted step. `fn` may return `Inf` (or `NaN`) at a trial point to reject it.

Value

A list with components `par`, `value`, `gradient`, `iterations`, `converged`, `status` (integer), `message` (the basin termination reason, e.g. "GradientTolerance", "MaxIter"), `radius` (final trust radius), `counts` (evaluations of `fn`, `gr`, `hess`, `hessvec`), and, when `control$trace` is TRUE, `trace`, a data frame with one row per subproblem attempt.

References

Steihaug, T. (1983). The conjugate gradient method and trust regions in large scale optimization. *SIAM Journal on Numerical Analysis*, 20(3), 626-637.

Nocedal, J. and Wright, S. J. (2006). *Numerical Optimization* (2nd ed.), Algorithms 4.1 and 7.2. Springer.

See Also

[trust()], [bobyqa()], [newuoa()]

Examples

```
fr <- function(x) 100 * (x[2] - x[1]^2)^2 + (1 - x[1])^2
grr <- function(x) c(-400 * x[1] * (x[2] - x[1]^2) - 2 * (1 - x[1]),
                    200 * (x[2] - x[1]^2))
hr <- function(x) matrix(c(1200 * x[1]^2 - 400 * x[2] + 2, -400 * x[1],
                        -400 * x[1], 200), 2, 2)
steihaug(c(-1.2, 1), fr, grr, hr)

# matrix-free: only Hessian-vector products
hv <- function(x, v) drop(hr(x) %**% v)
steihaug(c(-1.2, 1), fr, grr, hessvec = hv)$counts
```

trust

*Non-Linear Optimization by a Trust Region Algorithm***Description**

Thread-safe C++ port of `trust::trust()` (Geyer). Carries out a minimization or maximization of a function using a trust region algorithm; see the original package's documentation for the algorithm and argument semantics, which this function reproduces exactly.

Usage

```
trust(
  objfun,
  parinit,
  rinit,
  rmax,
  parscale,
  iterlim = 100,
  fterm = sqrt(.Machine$double.eps),
  mterm = sqrt(.Machine$double.eps),
  minimize = TRUE,
  blather = FALSE,
  ...
)
```

Arguments

<code>objfun</code>	an R function computing value, gradient, and Hessian; see <code>trust::trust</code> for the exact contract.
<code>parinit</code>	starting parameter values.
<code>rinit</code>	starting trust region radius.

rmax	maximum allowed trust region radius.
parscale	optional parameter scale vector.
iterlim	maximum number of iterations.
fterm	function-value termination tolerance.
mterm	model-decrease termination tolerance.
minimize	if TRUE minimize, if FALSE maximize.
blather	if TRUE return extra per-iteration information.
...	additional arguments passed to objfun.

Value

A list with the same components as `trust::trust()`.

Index

`.RcppTrustPtr`, [2](#)

`bobyqa`, [3](#)

`newuoa`, [4](#)

`print.minqa`, [5](#)

`steihaug`, [5](#)

`trust`, [7](#)