lbfgsb3x: Using the 2011 version of L-BFGSB

John C Nash Telfer School of Management, University of Ottawa, nashjc@uottawa.ca

February 20, 2019

Abstract

In 2011 the authors of the L-BFGSB program published a correction and update to their 1995 code. The latter is the basis of the L-BFGS-B method of the `optim()` function in base-R. The package `lbfgsb3x` is a merging of CRAN packages `lfgsb3` and `lbfgsb3c`. This vignette gives a brief explanation of the offerings of this package.

Provenance of the R optim::L-BFGS-B and related solvers

The base-R code lbfgsb.c (at writing in R-3.5.2/src/appl/) is commented:

```
/* l-bfgs-b.f -- translated by f2c (version 19991025).

  From ?optim:
  The code for method '"L-BFGS-B"' is based on Fortran code by Zhu,
  Byrd, Lu-Chen and Nocedal obtained from Netlib (file 'opt/lbfgs_bcm.shar')

  The Fortran files contained no copyright information.

  Byrd, R. H., Lu, P., Nocedal, J. and Zhu, C.  (1995) A limited
  memory algorithm for bound constrained optimization.
  \emph{SIAM J. Scientific Computing}, \bold{16}, 1190--1208.
*/
```

The paper R. H. Byrd, Lu, Nocedal, and Zhu (1995a) builds on Lu et al. (1994). There have been a number of other workers who have followed-up on this work, but **R** code and packages seem to have largely stayed with codes derived from these original papers. Though the date of the paper is 1995, the ideas it embodies were around for a decade and a half at least, in particular in Nocedal80 and LiuN89. The definitive Fortran code was published as Zhu et al. (1997). This is available as `toms/778.zip` on www.netlib.org. A side-by-side comparison of the main subroutines in the two downloads from Netlib unfortunately shows a lot of differences. I have not tried to determine if these affect performance or are simply cosmetic.

More seriously perhaps, there were some deficiencies in the code(s), and in 2011 Nocedal's team published a Fortran code with some corrections (Morales and Nocedal (2011)). Since the **R** code predates this, I prepared package `lbfgsb3` (Nash et al. (2015)) to wrap the Fortran code. However, I did not discover any test cases where the `optim::L-BFGS-B` and `lbfgsb3` were different, though I confess to only running some limited tests. There are, in fact, more in this vignette.

In 2016, I was at a Fields Institute optimization conference in Toronto for the 70th birthday of Andy Conn. By sheer serendipity, Nocedal did not attend the conference, but sat down next to me at the conference dinner. When I asked him about the key changes, he said that the most important one was to fix the computation of the machine precision, which was not always correct in the 1995 code. Since **R** gets this number as `.Machine$double.eps`, the offending code is irrelevant.

Within Morales and Nocedal (2011), there is also reported an improvement in the subspace minimization that is applied in cases of bounds constraints. Since few of the tests I have applied imporse such constraints, it is reasonable that I will not have observed performance differences between the base-R `optim` code and my `lbfsgb3` package. More appropriate tests are welcome, and on my agenda.

Besides the ACM TOMS code, there are two related codes from the Northwestern team on NETLIB: http://netlib.org/opt/lbfgs_um.shar is for unconstrained minimization, while http://netlib.org/opt/lbfgs_bcm.shar handles bounds constrained problems. To these are attached references Liu and Nocedal (1989) and R. H. Byrd, Lu, Nocedal, and Zhu (1995b) respectively, most likely reflecting the effort required to implement the constraints.

The unconstrained code has been converted to **C** under the leadership of Naoaki Okazaki (see http://www.chokkan.org/software/liblbfgs/, or the fork at https://github.com/MIRTK/LBFGS). This has been wrapped for **R** as Coppola, Stewart, and Okazaki (2014) as the `lbfgs` package. This can be called from `optimx::optimr()`.

Using Rcpp (see Eddelbuettel and François (2011)) and the Fortran code in package `lbfgs3`, Matthew Fidler developed package `lbfgsb3c` (Fidler et al. (2018)). As this provides a more standard call and return than `lbfgsb3` Fidler and I are unified the two packages as `lbfgsb3x`.

Functions in package `lbfgsb3x`

There are four optimizer functions in the package:

- `lbfgsb3`, which uses a `.Fortran` call of the compiled 2011 Fortran code. The object returned by this routine is NOT equivalent to the object returned by base-R `optim()` or by `optimx::optimr()`. Instead, it includes a structure `info` which contains the detailed diagnostic information of the Fortran code. For most users, this is not of interest, and I only recommend use of this function for those needing to examine how the optimization has been carried out.
- `lbfgsb3f()` wraps `lbfsgb3()` to produce a returned object of the same form as `optim()`.
- `lbfgsb3c()` uses Rcpp (Eddelbuettel (2013), Eddelbuettel and François (2011), Eddelbuettel and Balamuta (2017)) to streamline the call to the underlying Fortran.
- `lbfgsb3x()` is an alias of `lbfgsb3c()`

We recommend using the `lbfsgb3c()` call for most uses.

Comparison with optim::L-BFGS-B

```
# ref BT.RES in Nash and Walker-Smith (1987)

bt.f<-function(x){
 sum(x*x)
}

bt.g<-function(x){
  gg<-2.0*x
}

bt.badsetup<-function(n){
   x<-rep(0,n)
   lower<-rep(0,n)
   upper<-lower # to get arrays set
   bdmsk<-rep(1,n)
   bdmsk[(trunc(n/2)+1)]<-0
   for (i in 1:n) {
      x[i]<-2.2*i-n
      lower[i]<-1.0*(i-1)*(n-1)/n
      upper[i]<-1.0*i*(n+1)/n
   }
```

```
   result<-list(x=x, lower=lower, upper=upper, bdmsk=bdmsk)
}

bt.setup0<-function(n){
   x<-rep(0,n)
   lower<-rep(0,n)
   upper<-lower # to get arrays set
   bdmsk<-rep(1,n)
   bdmsk[(trunc(n/2)+1)]<-0
   for (i in 1:n) {
      lower[i]<-1.0*(i-1)*(n-1)/n
      upper[i]<-1.0*i*(n+1)/n
   }
   x<-0.5*(lower+upper)
   result<-list(x=x, lower=lower, upper=upper, bdmsk=bdmsk)
}

nn <- 4
baddy <- bt.badsetup(nn)
lo <- baddy$lower
up <- baddy$up
x0 <- baddy$x
baddy
```

```
## $x
## [1] -1.8  0.4  2.6  4.8
##
## $lower
## [1] 0.00 0.75 1.50 2.25
##
## $upper
## [1] 1.25 2.50 3.75 5.00
##
## $bdmsk
## [1] 1 1 0 1
```

```
## optim()
solbad0 <- optim(x0, bt.f, bt.g, lower=lo, upper=up, method="L-BFGS-B", control=list(trace=3))
```

```
## N = 4, M = 5 machine precision = 2.22045e-16
## The initial X is infeasible.  Restart with its projection.
## At X0, 2 variables are exactly at the bounds
## At iterate     0  f=       30.363  |proj g|=         2.55
##
## iterations 1
## function evaluations 2
## segments explored during Cauchy searches 3
## BFGS updates skipped 0
## active bounds at final generalized Cauchy point 4
## norm of the final projected gradient 0
## final function value 7.875
##
## F = 7.875
## final  value 7.875000
```

```
## converged
```

```
library(lbfgsb3x)
sol3c <- lbfgsb3x(x0, bt.f, bt.g, lower=lo, upper=up, control=list(trace=3))
```

```
##
## ============================================================================
## Before call f=17976931348623157081452742373170435679807056752584499659891747680315726078002853876058
## returned from lbfgsb3
## returned itask is 21 or "FG_START"
## computing f and g at prm=
## [1] 0.00 0.75 2.60 4.80
## At iteration 0 f=30.362500 max(abs(g))=0.000000
##
## ============================================================================
## Before call f=30.362500 task number 21, or "FG_START"
## returned from lbfgsb3
## returned itask is 20 or "FG_LNSRCH"
## computing f and g at prm=
## [1] 0.00 0.75 1.50 2.25
## At iteration 2 f=7.875000 max(abs(g))=0.000000
##
## ============================================================================
## Before call f=7.875000 task number 20, or "FG_LNSRCH"
## returned from lbfgsb3
## returned itask is 1 or "NEW_X"
##
## ============================================================================
## Before call f=7.875000 task number 1, or "NEW_X"
## returned from lbfgsb3
## returned itask is 7 or "CONVERGENCE: NORM_OF_PROJECTED_GRADIENT_<=_PGTOL"
```

```
library(microbenchmark)
tbad0 <- microbenchmark(optim(x0, bt.f, bt.g, lower=lo, upper=up, method="L-BFGS-B"))
t3c <- microbenchmark(lbfgsb3x(x0, bt.f, bt.g, lower=lo, upper=up))
tbad0
```

```
## Unit: microseconds
##                                                                  expr    min
##  optim(x0, bt.f, bt.g, lower = lo, upper = up, method = "L-BFGS-B") 39.775
##       lq     mean  median     uq     max neval
##  41.9215 46.02499 43.0405 47.144 131.552   100
```

```
t3c
```

```
## Unit: microseconds
##                                           expr     min      lq     mean
##  lbfgsb3x(x0, bt.f, bt.g, lower = lo, upper = up) 188.734 195.187 209.7215
##   median       uq     max neval
##  202.325 217.3165 444.283   100
```

```
library(optimx)
meths <- c("L-BFGS-B", "lbfgsb3c") # Note: lbfgsb3x not yet in optimx
allbt0 <- opm(x0, bt.f, bt.g, lower=lo, upper=up, method=meths)
```

```
## Warning in optimr(par, fn, gr, hess = hess, method = meth, lower = lower, : Parameter(s) changed to
```

```
## Warning in optimr(par, fn, gr, hess = hess, method = meth, lower = lower, : Parameter(s) changed to r
```

```
summary(allbt0, order=value)
```

```
##          p1   p2  p3   p4 value fevals gevals convergence  kkt1 kkt2 xtime
## L-BFGS-B  0 0.75 1.5 2.25 7.875      2      2           0 FALSE TRUE 0.004
## lbfgsb3c  0 0.75 1.5 2.25 7.875      2      2           0 FALSE TRUE 0.006
```

```
# candlestick function
# J C Nash 2011-2-3
cstick.f<-function(x,alpha=100){
  x<-as.vector(x)
  r2<-crossprod(x)
  f<-as.double(r2+alpha/r2)
  return(f)
}

cstick.g<-function(x,alpha=100){
  x<-as.vector(x)
  r2<-as.numeric(crossprod(x))
  g1<-2*x
  g2 <- (-alpha)*2*x/(r2*r2)
  g<-as.double(g1+g2)
  return(g)
}

nn <- 2
x0 <- c(10,10)
lo <- c(1, 1)
up <- c(10,10)

meths <- c("L-BFGS-B", "lbfgsb3c", "Rvmmin", "Rcgmin", "Rtnmin")
require(optimx)
print(x0)
```

```
## [1] 10 10
```

```
cstick2a <- opm(x0, cstick.f, cstick.g, method=meths, upper=up, lower=lo, control=list(kkt=FALSE))
```

```
## trace= 0
```

```
print(summary(cstick2a, par.select=1:2, order=value))
```

```
##              p1     p2 value fevals gevals convergence kkt1 kkt2 xtime
## Rcgmin   2.2361 2.2361    20     10      6           0   NA   NA 0.002
## L-BFGS-B 2.2361 2.2361    20     14     14           0   NA   NA 0.015
## lbfgsb3c 2.2361 2.2361    20     14     14           0   NA   NA 0.001
## Rvmmin   1.0000 1.0000    52      2      2           2   NA   NA 0.004
## Rtnmin   1.0000 1.0000    52      2      2           0   NA   NA 0.004
```

```
lo <- c(4, 4)
cstick2b <- opm(x0, cstick.f, cstick.g, method=meths, upper=up, lower=lo, control=list(kkt=FALSE))
```

```
## trace= 0
```

```
print(summary(cstick2b, par.select=1:2, order=value))
```

```
##          p1 p2  value fevals gevals convergence kkt1 kkt2 xtime
```

```
## L-BFGS-B  4  4 35.125      2      2           0   NA   NA 0.001
## lbfgsb3c  4  4 35.125      2      2           0   NA   NA 0.001
## Rtnmin    4  4 35.125      2      2           0   NA   NA 0.000
## Rvmmin    4  4 35.125      2      2           2   NA   NA 0.001
## Rcgmin    4  4 35.125      3      2           0   NA   NA 0.001
```

```
nn <- 100
x0 <- rep(10, nn)
up <- rep(10, nn)
lo <- rep(1e-4, nn)
cstickc0 <- opm(x0, cstick.f, cstick.g, method=meths, upper=up, lower=lo, control=list(kkt=FALSE))
```

```
## trace= 0
```

```
print(summary(cstickc0, par.select=1:5, order=value))
```

```
##               p1      p2      p3      p4      p5 value fevals gevals
## L-BFGS-B 0.31623 0.31623 0.31623 0.31623 0.31623    20     23     23
## lbfgsb3c 0.31623 0.31623 0.31623 0.31623 0.31623    20     23     23
## Rvmmin   0.31623 0.31623 0.31623 0.31623 0.31623    20     47     23
## Rcgmin   0.31623 0.31623 0.31623 0.31623 0.31623    20     24     18
## Rtnmin   0.31623 0.31623 0.31623 0.31623 0.31623    20     18     18
##          convergence kkt1 kkt2 xtime
## L-BFGS-B           0   NA   NA 0.002
## lbfgsb3c           0   NA   NA 0.003
## Rvmmin             0   NA   NA 0.072
## Rcgmin             0   NA   NA 0.018
## Rtnmin             0   NA   NA 0.003
```

```
lo <- rep(1, nn)
cstickca <- opm(x0, cstick.f, cstick.g, method=meths, upper=up, lower=lo, control=list(kkt=FALSE))
```

```
## trace= 0
```

```
print(summary(cstickca, par.select=1:5, order=value))
```

```
##          p1 p2 p3 p4 p5 value fevals gevals convergence kkt1 kkt2 xtime
## L-BFGS-B  1  1  1  1  1   101      2      2           0   NA   NA 0.001
## lbfgsb3c  1  1  1  1  1   101      2      2           0   NA   NA 0.001
## Rvmmin    1  1  1  1  1   101      2      2           2   NA   NA 0.018
## Rcgmin    1  1  1  1  1   101      3      2           0   NA   NA 0.000
## Rtnmin    1  1  1  1  1   101      2      2           0   NA   NA 0.000
```

```
lo <- rep(4, nn)
cstickcb <- opm(x0, cstick.f, cstick.g, method=meths, upper=up, lower=lo, control=list(kkt=FALSE))
```

```
## trace= 0
```

```
print(summary(cstickcb, par.select=1:5, order=value))
```

```
##          p1 p2 p3 p4 p5  value fevals gevals convergence kkt1 kkt2 xtime
## L-BFGS-B  4  4  4  4  4 1600.1      2      2           0   NA   NA 0.000
## lbfgsb3c  4  4  4  4  4 1600.1      2      2           0   NA   NA 0.001
## Rvmmin    4  4  4  4  4 1600.1      2      2           2   NA   NA 0.019
## Rcgmin    4  4  4  4  4 1600.1      3      2           0   NA   NA 0.010
## Rtnmin    4  4  4  4  4 1600.1      2      2           0   NA   NA 0.001
```

```
require(funconstrain)
```

```
## Loading required package: funconstrain
```

```
exrosen <- ex_rosen()
exrosenf <- exrosen$fn
exroseng <- exrosen$gr
meths <- c("L-BFGS-B", "lbfgsb3c", "Rvmmin", "Rcgmin", "Rtnmin")
require(optimx)


for (n in seq(2,12, by=2)) {
  cat("ex_rosen try for n=",n,"\n")
  x0 <- exrosen$x0(n)
  lo <- rep(.5, n)
  up <- rep(3, n)
  print(x0)
  erfg <- opm(x0, exrosenf, exroseng, method=meths, lower=lo, upper=up)
  print(summary(erfg, par.select=1:2, order=value))
}
```

```
## ex_rosen try for n= 2
## [1] -1.2  1.0
```

```
## Warning in optimr(par, fn, gr, hess = hess, method = meth, lower = lower, : Parameter(s) changed to r
```

```
## Warning in optimr(par, fn, gr, hess = hess, method = meth, lower = lower, : Parameter(s) changed to r
```

```
## Warning in optimr(par, fn, gr, hess = hess, method = meth, lower = lower, : Parameter(s) changed to r
```

```
## Warning in Rvmmin(par = spar, fn = efn, gr = egr, lower = slower, upper =
## supper, : Parameter out of bounds has been moved to nearest bound
```

```
## trace= 0
```

```
## Warning in optimr(par, fn, gr, hess = hess, method = meth, lower = lower, : Parameter(s) changed to r
```

```
## Warning in Rcgminb(par = spar, fn = efn, gr = egr, lower = slower, upper =
## supper, : x[1], set -1.2 to lower bound = 0.5
```

```
## Warning in optimr(par, fn, gr, hess = hess, method = meth, lower = lower, : Parameter(s) changed to r
```

```
##           p1  p2      value fevals gevals convergence  kkt1  kkt2 xtime
## Rcgmin   1.0 1.0 9.3483e-15    119     72           0  TRUE  TRUE 0.004
## Rtnmin   1.0 1.0 6.5821e-13     20     20           0  TRUE  TRUE 0.003
## lbfgsb3c 1.0 1.0 4.6379e-11     19     19           0  TRUE  TRUE 0.003
## L-BFGS-B 1.0 1.0 4.6379e-11     19     19           0  TRUE  TRUE 0.001
## Rvmmin   0.5 0.5 6.5000e+00      3      3           2 FALSE FALSE 0.002
## ex_rosen try for n= 4
## [1] -1.2  1.0 -1.2  1.0
```

```
## Warning in optimr(par, fn, gr, hess = hess, method = meth, lower = lower, : Parameter(s) changed to r
```

```
## Warning in optimr(par, fn, gr, hess = hess, method = meth, lower = lower, : Parameter(s) changed to r
```

```
## Warning in optimr(par, fn, gr, hess = hess, method = meth, lower = lower, : Parameter(s) changed to r
```

```
## Warning in Rvmmin(par = spar, fn = efn, gr = egr, lower = slower, upper =
## supper, : Parameter out of bounds has been moved to nearest bound
```

```
## trace= 0
```

```
## Warning in optimr(par, fn, gr, hess = hess, method = meth, lower = lower, : Parameter(s) changed to r

## Warning in Rcgminb(par = spar, fn = efn, gr = egr, lower = slower, upper =
## supper, : x[1], set -1.2 to lower bound = 0.5

## Warning in Rcgminb(par = spar, fn = efn, gr = egr, lower = slower, upper =
## supper, : x[3], set -1.2 to lower bound = 0.5

## Warning in optimr(par, fn, gr, hess = hess, method = meth, lower = lower, : Parameter(s) changed to r

##           p1  p2      value fevals gevals convergence  kkt1  kkt2 xtime
## Rcgmin   1.0 1.0 1.8697e-14    119     72           0  TRUE  TRUE 0.005
## Rtnmin   1.0 1.0 6.7977e-13     19     19           0  TRUE  TRUE 0.003
## lbfgsb3c 1.0 1.0 9.2757e-11     19     19           0  TRUE  TRUE 0.003
## L-BFGS-B 1.0 1.0 9.2757e-11     19     19           0  TRUE  TRUE 0.001
## Rvmmin   0.5 0.5 1.3000e+01      3      3           2 FALSE FALSE 0.002
## ex_rosen try for n= 6
## [1] -1.2  1.0 -1.2  1.0 -1.2  1.0

## Warning in optimr(par, fn, gr, hess = hess, method = meth, lower = lower, : Parameter(s) changed to r

## Warning in optimr(par, fn, gr, hess = hess, method = meth, lower = lower, : Parameter(s) changed to r

## Warning in optimr(par, fn, gr, hess = hess, method = meth, lower = lower, : Parameter(s) changed to r

## Warning in Rvmmin(par = spar, fn = efn, gr = egr, lower = slower, upper =
## supper, : Parameter out of bounds has been moved to nearest bound

## trace= 0

## Warning in optimr(par, fn, gr, hess = hess, method = meth, lower = lower, : Parameter(s) changed to r

## Warning in Rcgminb(par = spar, fn = efn, gr = egr, lower = slower, upper =
## supper, : x[1], set -1.2 to lower bound = 0.5

## Warning in Rcgminb(par = spar, fn = efn, gr = egr, lower = slower, upper =
## supper, : x[3], set -1.2 to lower bound = 0.5

## Warning in Rcgminb(par = spar, fn = efn, gr = egr, lower = slower, upper =
## supper, : x[5], set -1.2 to lower bound = 0.5

## Warning in optimr(par, fn, gr, hess = hess, method = meth, lower = lower, : Parameter(s) changed to r

##           p1  p2      value fevals gevals convergence  kkt1  kkt2 xtime
## Rcgmin   1.0 1.0 2.8417e-14    119     72           0  TRUE  TRUE 0.005
## Rtnmin   1.0 1.0 6.7659e-12     20     20           0  TRUE  TRUE 0.005
## L-BFGS-B 1.0 1.0 1.3914e-10     19     19           0  TRUE  TRUE 0.001
## lbfgsb3c 1.0 1.0 1.3914e-10     19     19           0  TRUE  TRUE 0.002
## Rvmmin   0.5 0.5 1.9500e+01      3      3           2 FALSE FALSE 0.002
## ex_rosen try for n= 8
## [1] -1.2  1.0 -1.2  1.0 -1.2  1.0 -1.2  1.0

## Warning in optimr(par, fn, gr, hess = hess, method = meth, lower = lower, : Parameter(s) changed to r

## Warning in optimr(par, fn, gr, hess = hess, method = meth, lower = lower, : Parameter(s) changed to r

## Warning in optimr(par, fn, gr, hess = hess, method = meth, lower = lower, : Parameter(s) changed to r

## Warning in Rvmmin(par = spar, fn = efn, gr = egr, lower = slower, upper =
## supper, : Parameter out of bounds has been moved to nearest bound

## trace= 0
```

```
## Warning in optimr(par, fn, gr, hess = hess, method = meth, lower = lower, : Parameter(s) changed to r

## Warning in Rcgminb(par = spar, fn = efn, gr = egr, lower = slower, upper =
## supper, : x[1], set -1.2 to lower bound = 0.5

## Warning in Rcgminb(par = spar, fn = efn, gr = egr, lower = slower, upper =
## supper, : x[3], set -1.2 to lower bound = 0.5

## Warning in Rcgminb(par = spar, fn = efn, gr = egr, lower = slower, upper =
## supper, : x[5], set -1.2 to lower bound = 0.5

## Warning in Rcgminb(par = spar, fn = efn, gr = egr, lower = slower, upper =
## supper, : x[7], set -1.2 to lower bound = 0.5

## Warning in optimr(par, fn, gr, hess = hess, method = meth, lower = lower, : Parameter(s) changed to r

##           p1  p2      value fevals gevals convergence  kkt1  kkt2 xtime
## L-BFGS-B 1.0 1.0 1.6073e-16     20     20           0  TRUE  TRUE 0.001
## lbfgsb3c 1.0 1.0 1.6073e-16     20     20           0  TRUE  TRUE 0.003
## Rcgmin   1.0 1.0 3.7282e-14    119     72           0  TRUE  TRUE 0.006
## Rtnmin   1.0 1.0 9.0218e-12     20     20           0  TRUE  TRUE 0.004
## Rvmmin   0.5 0.5 2.6000e+01      3      3           2 FALSE FALSE 0.003
## ex_rosen try for n= 10
##  [1] -1.2  1.0 -1.2  1.0 -1.2  1.0 -1.2  1.0 -1.2  1.0

## Warning in optimr(par, fn, gr, hess = hess, method = meth, lower = lower, : Parameter(s) changed to r

## Warning in optimr(par, fn, gr, hess = hess, method = meth, lower = lower, : Parameter(s) changed to r

## Warning in optimr(par, fn, gr, hess = hess, method = meth, lower = lower, : Parameter(s) changed to r

## Warning in Rvmmin(par = spar, fn = efn, gr = egr, lower = slower, upper =
## supper, : Parameter out of bounds has been moved to nearest bound

## trace= 0

## Warning in optimr(par, fn, gr, hess = hess, method = meth, lower = lower, : Parameter(s) changed to r

## Warning in Rcgminb(par = spar, fn = efn, gr = egr, lower = slower, upper =
## supper, : x[1], set -1.2 to lower bound = 0.5

## Warning in Rcgminb(par = spar, fn = efn, gr = egr, lower = slower, upper =
## supper, : x[3], set -1.2 to lower bound = 0.5

## Warning in Rcgminb(par = spar, fn = efn, gr = egr, lower = slower, upper =
## supper, : x[5], set -1.2 to lower bound = 0.5

## Warning in Rcgminb(par = spar, fn = efn, gr = egr, lower = slower, upper =
## supper, : x[7], set -1.2 to lower bound = 0.5

## Warning in Rcgminb(par = spar, fn = efn, gr = egr, lower = slower, upper =
## supper, : x[9], set -1.2 to lower bound = 0.5

## Warning in optimr(par, fn, gr, hess = hess, method = meth, lower = lower, : Parameter(s) changed to r

##           p1  p2      value fevals gevals convergence  kkt1  kkt2 xtime
## lbfgsb3c 1.0 1.0 2.0092e-16     20     20           0  TRUE  TRUE 0.002
## L-BFGS-B 1.0 1.0 2.0092e-16     20     20           0  TRUE  TRUE 0.002
## Rcgmin   1.0 1.0 4.7018e-14    119     72           0  TRUE  TRUE 0.005
## Rtnmin   1.0 1.0 1.1278e-11     20     20           0  TRUE  TRUE 0.004
## Rvmmin   0.5 0.5 3.2500e+01      3      3           2 FALSE FALSE 0.003
## ex_rosen try for n= 12
```

```
##  [1] -1.2  1.0 -1.2  1.0 -1.2  1.0 -1.2  1.0 -1.2  1.0 -1.2  1.0
## Warning in optimr(par, fn, gr, hess = hess, method = meth, lower = lower, : Parameter(s) changed to r
## Warning in optimr(par, fn, gr, hess = hess, method = meth, lower = lower, : Parameter(s) changed to r
## Warning in optimr(par, fn, gr, hess = hess, method = meth, lower = lower, : Parameter(s) changed to r
## Warning in Rvmmin(par = spar, fn = efn, gr = egr, lower = slower, upper =
## supper, : Parameter out of bounds has been moved to nearest bound
## trace= 0
## Warning in optimr(par, fn, gr, hess = hess, method = meth, lower = lower, : Parameter(s) changed to r
## Warning in Rcgminb(par = spar, fn = efn, gr = egr, lower = slower, upper =
## supper, : x[1], set -1.2 to lower bound = 0.5
## Warning in Rcgminb(par = spar, fn = efn, gr = egr, lower = slower, upper =
## supper, : x[3], set -1.2 to lower bound = 0.5
## Warning in Rcgminb(par = spar, fn = efn, gr = egr, lower = slower, upper =
## supper, : x[5], set -1.2 to lower bound = 0.5
## Warning in Rcgminb(par = spar, fn = efn, gr = egr, lower = slower, upper =
## supper, : x[7], set -1.2 to lower bound = 0.5
## Warning in Rcgminb(par = spar, fn = efn, gr = egr, lower = slower, upper =
## supper, : x[9], set -1.2 to lower bound = 0.5
## Warning in Rcgminb(par = spar, fn = efn, gr = egr, lower = slower, upper =
## supper, : x[11], set -1.2 to lower bound = 0.5
## Warning in optimr(par, fn, gr, hess = hess, method = meth, lower = lower, : Parameter(s) changed to r
##           p1  p2      value fevals gevals convergence  kkt1  kkt2 xtime
## L-BFGS-B 1.0 1.0 2.4110e-16     20     20           0  TRUE  TRUE 0.002
## lbfgsb3c 1.0 1.0 2.4110e-16     20     20           0  TRUE  TRUE 0.003
## Rcgmin   1.0 1.0 5.6031e-14    119     72           0  TRUE  TRUE 0.007
## Rtnmin   1.0 1.0 1.3529e-11     21     21           0  TRUE  TRUE 0.004
## Rvmmin   0.5 0.5 3.9000e+01      3      3           2 FALSE FALSE 0.000
```

References

Byrd, Richard H., Peihuang Lu, Jorge Nocedal, and Ci You Zhu. 1995a. "A Limited Memory Algorithm for Bound Constrained Optimization." *SIAM Journal on Scientific Computing* 16 (5): 1190–1208.

Byrd, Richard H., Peihuang Lu, Jorge Nocedal, and Ciyou Zhu. 1995b. "A Limited Memory Algorithm for Bound Constrained Optimization." *SIAM J. Sci. Comput.* 16 (5). Philadelphia, PA, USA: Society for Industrial; Applied Mathematics: 1190–1208. https://doi.org/10.1137/0916069.

Coppola, Antonio, Brandon Stewart, and Naoaki Okazaki. 2014. *Lbfgs: Limited-Memory Bfgs Optimization.* https://CRAN.R-project.org/package=lbfgs.

Eddelbuettel, Dirk. 2013. *Seamless R and C++ Integration with Rcpp.* New York: Springer. https://doi.org/10.1007/978-1-4614-6868-4.

Eddelbuettel, Dirk, and James Joseph Balamuta. 2017. "Extending extitR with extitC++: A Brief Introduction to extitRcpp." *PeerJ Preprints* 5 (August): e3188v1. https://doi.org/10.7287/peerj.preprints.3188v1.

Eddelbuettel, Dirk, and Romain François. 2011. "Rcpp: Seamless R and C++ Integration." *Journal of Statistical Software* 40 (8): 1–18. https://doi.org/10.18637/jss.v040.i08.

Fidler, Matthew L, John C Nash, Ciyou Zhu, Richard Byrd, Jorge Nocedal, and Jose Luis Morales. 2018. *lbfgsb3c: Limited Memory Bfgs Minimizer with Bounds on Parameters with Optim() 'c' Interface*. https://CRAN.R-project.org/package=lbfgsb3c.

Liu, Dong C., and Jorge Nocedal. 1989. "On the Limited Memory BFGS Method for Large Scale Optimization." *Math. Program.* 45 (1-3): 503–28. https://doi.org/10.1007/BF01589116.

Lu, Peihuang, Jorge Nocedal, Ciyou Zhu, and Richard H. Byrd. 1994. "A Limited-Memory Algorithm for Bound Constrained Optimization." *SIAM Journal on Scientific Computing* 16: 1190–1208.

Morales, José Luis, and Jorge Nocedal. 2011. "Remark on Algorithm 778: L-BFGS-B: Fortran subroutines for large-scale bound constrained optimization." *ACM Trans. Math. Softw.* 38 (1). New York, NY, USA: ACM: 7:1–7:4. http://doi.acm.org/10.1145/2049662.2049669.

Nash, John C, Ciyou Zhu, Richard Byrd, Jorge Nocedal, and Jose Luis Morales. 2015. *lbfgsb3: Limited Memory Bfgs Minimizer with Bounds on Parameters*. https://CRAN.R-project.org/package=lbfgsb3.

Zhu, Ciyou, Richard H. Byrd, Peihuang Lu, and Jorge Nocedal. 1997. "Algorithm 778: L-BFGS-B: Fortran Subroutines for Large-Scale Bound-Constrained Optimization." *ACM Trans. Math. Softw.* 23 (4). New York, NY, USA: ACM: 550–60. https://doi.org/10.1145/279232.279236.