Package 'Rcgmin'

February 19, 2015

Type Package

Title Conjugate Gradient Minimization of Nonlinear Functions

Version 2013-2.21

Date 2013-02-21

Author John C. Nash

Maintainer John C. Nash <nashjc@uottawa.ca>

Description Conjugate gradient minimization of nonlinear functions with box constraints incorporating the Dai/Yuan update. This implementation should be used in place of the ``CG" algorithm of the optim() function.

Imports numDeriv

License GPL (>= 2)

NeedsCompilation no

Repository CRAN

Date/Publication 2014-12-06 00:21:13

R topics documented:

`Rcgmin` *An R implementation of a nonlinear conjugate gradient algorithm with the Dai / Yuan update and restart. Based on Nash (1979) Algorithm 22 for its main structure.*

Description

The purpose of Rcgmin is to minimize an unconstrained or bounds (box) and mask constrained function of many parameters by a nonlinear conjugate gradients method. This code is entirely in R to allow users to explore and understand the method. It also allows bounds (or box) constraints and masks (equality constraints) to be imposed on parameters.

Rcgmin is a wrapper that calls Rcgminu for unconstrained problems, else Rcgminb.

Usage

```
Rcgmin(par, fn, gr, lower, upper, bdmsk, control = list(), ...)
```

Arguments

par	A numeric vector of starting estimates.
fn	A function that returns the value of the objective at the supplied set of parameters par using auxiliary data in .... The first argument of fn must be par.
gr	A function that returns the gradient of the objective at the supplied set of parameters par using auxiliary data in .... The first argument of fn must be par. This function returns the gradient as a numeric vector. If gr is not provided or is NULL, then grad() from package numDeriv is used. However, we strongly recommend carefully coded and checked analytic derivatives for Rcgmin.
lower	A vector of lower bounds on the parameters.
upper	A vector of upper bounds on the parameters.
bdmsk	An indicator vector, having 1 for each parameter that is "free" or unconstrained, and 0 for any parameter that is fixed or MASKED for the duration of the optimization.
control	An optional list of control settings.
...	Further arguments to be passed to fn.

Details

Functions fn must return a numeric value.

The control argument is a list.

maxit A limit on the number of iterations (default 500). Note that this is used to compute a quantity maxfeval<-round(sqrt(n+1)*maxit) where n is the number of parameters to be minimized.

trace Set 0 (default) for no output, >0 for trace output (larger values imply more output).

eps Tolerance used to calculate numerical gradients. Default is 1.0E-7. See source code for Rcgmin for details of application.

dowarn = TRUE if we want warnings generated by optimx. Default is TRUE.

tol Tolerance used in testing the size of the square of the gradient. Default is 0 on input, which uses a value of tolgr = npar*npar*.Machine$double.eps in testing if crossprod(g) <= tolgr * (abs(fmin) + reltest). If the user supplies a value for tol that is non-zero, then that value is used for tolgr.

reltest=100 is only alterable by changing the code. fmin is the current best value found for the function minimum value.

Note that the scale of the gradient means that tests for a small gradient can easily be mismatched to a given problem. The defaults in Rcgmin are a "best guess".

The source code `Rcgmin` for R is likely to remain a work in progress for some time, so users should watch the console output.

As of 2011-11-21 the following controls have been REMOVED

usenumDeriv There is now a choice of numerical gradient routines. See argument `gr`.

maximize To maximize user_function, supply a function that computes (-1)*user_function. An alternative is to call Rcgmin via the package optimx, where the MAXIMIZE field of the OPCON structure in package `optfntools` is used.

Value

A list with components:

`par`	The best set of parameters found.
`value`	The value of the objective at the best set of parameters found.
`counts`	A two-element integer vector giving the number of calls to 'fn' and 'gr' respectively. This excludes those calls needed to compute the Hessian, if requested, and any calls to 'fn' to compute a finite-difference approximation to the gradient.
`convergence`	An integer code. '0' indicates successful convergence. '1' indicates that the function evaluation count 'maxfeval' was reached. '2' indicates initial point is infeasible.
`message`	A character string giving any additional information returned by the optimizer, or 'NULL'.
`bdmsk`	Returned index describing the status of bounds and masks at the proposed solution. Parameters for which bdmsk are 1 are unconstrained or "free", those with bdmsk 0 are masked i.e., fixed. For historical reasons, we indicate a parameter is at a lower bound using -3 or upper bound using -1.

References

Dai, Y. H. and Y. Yuan (2001). An efficient hybrid conjugate gradient method for unconstrained optimization. Annals of Operations Research 103 (1-4), 33–47.

Nash JC (1979). Compact Numerical Methods for Computers: Linear Algebra and Function Minimisation. Adam Hilger, Bristol. Second Edition, 1990, Bristol: Institute of Physics Publications.

Nash, J. C. and M. Walker-Smith (1987). Nonlinear Parameter Estimation: An Integrated System in BASIC. New York: Marcel Dekker. See http://www.nashinfo.com/nlpe.htm for a downloadable version of this plus some extras.

See Also

`optim`

Examples

```
#####################
require(numDeriv)
## Rosenbrock Banana function
fr <- function(x) {
    x1 <- x[1]
    x2 <- x[2]
    100 * (x2 - x1 * x1)^2 + (1 - x1)^2
}

grr <- function(x) { ## Gradient of 'fr'
    x1 <- x[1]
    x2 <- x[2]
    c(-400 * x1 * (x2 - x1 * x1) - 2 * (1 - x1),
       200 *      (x2 - x1 * x1))
}

grn<-function(x){
    gg<-grad(fr, x)
}


ansrosenbrock0 <- Rcgmin(fn=fr,gr=grn, par=c(1,2))
print(ansrosenbrock0) # use print to allow copy to separate file that
#    can be called using source()
#####################
# Simple bounds and masks test
bt.f<-function(x){
 sum(x*x)
}

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

n<-10
xx<-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
}
xx<-0.5*(lower+upper)
ansbt<-Rcgmin(xx, bt.f, bt.g, lower, upper, bdmsk, control=list(trace=1))

print(ansbt)

#####################
genrose.f<- function(x, gs=NULL){ # objective function
```

```
## One generalization of the Rosenbrock banana valley function (n parameters)
n <- length(x)
        if(is.null(gs)) { gs=100.0 }
fval<-1.0 + sum (gs*(x[1:(n-1)]^2 - x[2:n])^2 + (x[2:n] - 1)^2)
        return(fval)
}
genrose.g <- function(x, gs=NULL){
# vectorized gradient for genrose.f
# Ravi Varadhan 2009-04-03
n <- length(x)
        if(is.null(gs)) { gs=100.0 }
gg <- as.vector(rep(0, n))
tn <- 2:n
tn1 <- tn - 1
z1 <- x[tn] - x[tn1]^2
z2 <- 1 - x[tn]
gg[tn] <- 2 * (gs * z1 - z2)
gg[tn1] <- gg[tn1] - 4 * gs * x[tn1] * z1
gg
}

# analytic gradient test
xx<-rep(pi,10)
lower<-NULL
upper<-NULL
bdmsk<-NULL
genrosea<-Rcgmin(xx,genrose.f, genrose.g, gs=10)
genrosenn<-Rcgmin(xx,genrose.f, gs=10) # use local numerical gradient
cat("genrosea uses analytic gradient\n")
print(genrosea)
cat("genrosenn uses numDeriv gradient\n")
print(genrosenn)



cat("timings B vs U\n")
lo<-rep(-100,10)
up<-rep(100,10)
bdmsk<-rep(1,10)
tb<-system.time(ab<-Rcgminb(xx,genrose.f, genrose.g, lower=lo, upper=up, bdmsk=bdmsk))[1]
tu<-system.time(au<-Rcgminu(xx,genrose.f, genrose.g))[1]
cat("times U=",tu,"   B=",tb,"\n")
cat("solution Rcgminu\n")
print(au)
cat("solution Rcgminb\n")
print(ab)
cat("diff fu-fb=",au$value-ab$value,"\n")
cat("max abs parameter diff = ", max(abs(au$par-ab$par)),"\n")



maxfn<-function(x) {
       n<-length(x)
```

```
ss<-seq(1,n)
f<-10-(crossprod(x-ss))^2
f<-as.numeric(f)
return(f)
}

gmaxfn<-function(x) {
     gg<-grad(maxfn, x)
}


negmaxfn<-function(x) {
f<-(-1)*maxfn(x)
return(f)
}



cat("test that maximize=TRUE works correctly\n")

n<-6
xx<-rep(1,n)
ansmax<-Rcgmin(xx,maxfn, control=list(maximize=TRUE,trace=1))
print(ansmax)

cat("using the negmax function should give same parameters\n")
ansnegmax<-Rcgmin(xx,negmaxfn, control=list(trace=1))
print(ansnegmax)


#####################  From Rvmmin.Rd
cat("test bounds and masks\n")
nn<-4
startx<-rep(pi,nn)
lo<-rep(2,nn)
up<-rep(10,nn)
grbds1<-Rcgmin(startx,genrose.f, gr=genrose.g,lower=lo,upper=up)
print(grbds1)

cat("test lower bound only\n")
nn<-4
startx<-rep(pi,nn)
lo<-rep(2,nn)
grbds2<-Rcgmin(startx,genrose.f, gr=genrose.g,lower=lo)
print(grbds2)

cat("test lower bound single value only\n")
nn<-4
startx<-rep(pi,nn)
lo<-2
up<-rep(10,nn)
grbds3<-Rcgmin(startx,genrose.f, gr=genrose.g,lower=lo)
print(grbds3)
```

```
cat("test upper bound only\n")
nn<-4
startx<-rep(pi,nn)
lo<-rep(2,nn)
up<-rep(10,nn)
grbds4<-Rcgmin(startx,genrose.f, gr=genrose.g,upper=up)
print(grbds4)

cat("test upper bound single value only\n")
nn<-4
startx<-rep(pi,nn)
grbds5<-Rcgmin(startx,genrose.f, gr=genrose.g,upper=10)
print(grbds5)



cat("test masks only\n")
nn<-6
bd<-c(1,1,0,0,1,1)
startx<-rep(pi,nn)
grbds6<-Rcgmin(startx,genrose.f, gr=genrose.g,bdmsk=bd)
print(grbds6)

cat("test upper bound on first two elements only\n")
nn<-4
startx<-rep(pi,nn)
upper<-c(10,8, Inf, Inf)
grbds7<-Rcgmin(startx,genrose.f, gr=genrose.g,upper=upper)
print(grbds7)


cat("test lower bound on first two elements only\n")
nn<-4
startx<-rep(0,nn)
lower<-c(0,1.1, -Inf, -Inf)
grbds8<-Rcgmin(startx,genrose.f,genrose.g,lower=lower, control=list(maxit=2000))
print(grbds8)

cat("test n=1 problem using simple squares of parameter\n")

sqtst<-function(xx) {
   res<-sum((xx-2)*(xx-2))
}

gsqtst<-function(xx) {
    gg<-2*(xx-2)
}

######### One dimension test
nn<-1
startx<-rep(0,nn)
onepar<-Rcgmin(startx,sqtst,  gr=gsqtst,control=list(trace=1))
```

```
print(onepar)

cat("Suppress warnings\n")
oneparnw<-Rcgmin(startx,sqtst,  gr=gsqtst,control=list(dowarn=FALSE,trace=1))
print(oneparnw)
```

Rcgminb — *An R implementation of a bounded nonlinear conjugate gradient algorithm with the Dai / Yuan update and restart. Based on Nash (1979) Algorithm 22 for its main structure. CALL THIS VIA* Rcgmin *AND DO NOT USE DIRECTLY.*

Description

The purpose of Rcgminb is to minimize a bounds (box) and mask constrained function of many parameters by a nonlinear conjugate gradients method. This code is entirely in R to allow users to explore and understand the method. It allows bounds (or box) constraints and masks (equality constraints) to be imposed on parameters.

This code should be called through Rcgmin which selects Rcgminb or Rcgminu according to the presence of bounds and masks.

Usage

```
Rcgminb(par, fn, gr, lower, upper, bdmsk, control = list(), ...)
```

Arguments

par	A numeric vector of starting estimates.
fn	A function that returns the value of the objective at the supplied set of parameters par using auxiliary data in .... The first argument of fn must be par.
gr	A function that returns the gradient of the objective at the supplied set of parameters par using auxiliary data in .... The first argument of fn must be par. This function returns the gradient as a numeric vector. The use of numerical gradients for Rcgminb is STRONGLY discouraged.
lower	A vector of lower bounds on the parameters.
upper	A vector of upper bounds on the parameters.
bdmsk	An indicator vector, having 1 for each parameter that is "free" or unconstrained, and 0 for any parameter that is fixed or MASKED for the duration of the optimization.
control	An optional list of control settings.
...	Further arguments to be passed to fn.

Details

Functions `fn` must return a numeric value.

The `control` argument is a list.

maxit A limit on the number of iterations (default 500). Note that this is used to compute a quantity `maxfeval`<-round(sqrt(n+1)*maxit) where n is the number of parameters to be minimized.

trace Set 0 (default) for no output, >0 for trace output (larger values imply more output).

eps Tolerance used to calculate numerical gradients. Default is 1.0E-7. See source code for `Rcgminb` for details of application.

`dowarn` = TRUE if we want warnings generated by optimx. Default is TRUE.

The source code `Rcgminb` for R is likely to remain a work in progress for some time, so users should watch the console output.

As of 2011-11-21 the following controls have been REMOVED

usenumDeriv There is now a choice of numerical gradient routines. See argument `gr`.

maximize To maximize user_function, supply a function that computes (-1)*user_function. An alternative is to call Rcgmin via the package optimx.

Value

A list with components:

`par`	The best set of parameters found.
`value`	The value of the objective at the best set of parameters found.
`counts`	A two-element integer vector giving the number of calls to 'fn' and 'gr' respectively. This excludes those calls needed to compute the Hessian, if requested, and any calls to 'fn' to compute a finite-difference approximation to the gradient.
`convergence`	An integer code. '0' indicates successful convergence. '1' indicates that the function evaluation count 'maxfeval' was reached. '2' indicates initial point is infeasible.
`message`	A character string giving any additional information returned by the optimizer, or 'NULL'.
`bdmsk`	Returned index describing the status of bounds and masks at the proposed solution. Parameters for which bdmsk are 1 are unconstrained or "free", those with bdmsk 0 are masked i.e., fixed. For historical reasons, we indicate a parameter is at a lower bound using -3 or upper bound using -1.

References

See `Rcgmin` documentation. Note that bounds and masks were adapted from the work by Nash and Walker-Smith(1987).

See Also

`optim`

Rcgminu — *An R implementation of an unconstrained nonlinear conjugate gradient algorithm with the Dai / Yuan update and restart. Based on Nash (1979) Algorithm 22 for its main structure. CALL THIS VIA* Rcgmin *AND DO NOT USE DIRECTLY.*

Description

The purpose of Rcgminu is to minimize an unconstrained function of many parameters by a nonlinear conjugate gradients method. This code is entirely in R to allow users to explore and understand the method.

This code should be called through Rcgmin which selects Rcgminb or Rcgminu according to the presence of bounds and masks.

Usage

```
Rcgminu(par, fn, gr, control = list(), ...)
```

Arguments

par	A numeric vector of starting estimates.
fn	A function that returns the value of the objective at the supplied set of parameters par using auxiliary data in .... The first argument of fn must be par.
gr	A function that returns the gradient of the objective at the supplied set of parameters par using auxiliary data in .... The first argument of fn must be par. This function returns the gradient as a numeric vector. The use of numerical gradients for Rcgminu is STRONGLY discouraged.
control	An optional list of control settings.
...	Further arguments to be passed to fn.

Details

Functions fn must return a numeric value.

The control argument is a list.

maxit A limit on the number of iterations (default 500). Note that this is used to compute a quantity maxfeval<-round(sqrt(n+1)*maxit) where n is the number of parameters to be minimized.

trace Set 0 (default) for no output, >0 for trace output (larger values imply more output).

eps Tolerance used to calculate numerical gradients. Default is 1.0E-7. See source code for Rcgminu for details of application.

dowarn = TRUE if we want warnings generated by optimx. Default is TRUE.

The source code Rcgminu for R is likely to remain a work in progress for some time, so users should watch the console output.

As of 2011-11-21 the following controls have been REMOVED

usenumDeriv There is now a choice of numerical gradient routines. See argument gr.

maximize To maximize user_function, supply a function that computes (-1)*user_function. An alternative is to call Rcgmin via the package optimx.

Value

A list with components:

par	The best set of parameters found.
value	The value of the objective at the best set of parameters found.
counts	A two-element integer vector giving the number of calls to 'fn' and 'gr' respectively. This excludes those calls needed to compute the Hessian, if requested, and any calls to 'fn' to compute a finite-difference approximation to the gradient.
convergence	An integer code. '0' indicates successful convergence. '1' indicates that the function evaluation count 'maxfeval' was reached. '2' indicates initial point is infeasible.
message	A character string giving any additional information returned by the optimizer, or 'NULL'.
bdmsk	Returned index describing the status of bounds and masks at the proposed solution. Parameters for which bdmsk are 1 are unconstrained or "free", those with bdmsk 0 are masked i.e., fixed. For historical reasons, we indicate a parameter is at a lower bound using -3 or upper bound using -1.

References

See Rcgmin documentation.

See Also

optim

Index