The minpack.lm Package
April 19, 2005

Version 1.0-5
Date Mon Jan 10 11:50:00 MSK 2005
Title R interface for MINPACK least squares optimization library
Author Timur V. Elzhov
Maintainer Timur V. Elzhov <Timur.Elzhov@jinr.ru>

Description Provides R interface for two functions from MINPACK library, solving nonlinear least squares problem by modification of the Levenberg-Marquardt algorithm.

License GPL (version 2 or higher)

R topics documented:

nls.lm .................................................. 1

Index 6

nls.lm Solving NLS problem by Levenberg-Marquardt algorithm

Description

The purpose of nls.lm is to minimize the square sum of the vector returned by fn function, by a modification of the Levenberg-Marquardt algorithm. The user may also provide jac function which calculates the Jacobian.

Usage

nls.lm(par, fn, jac = NULL, control = list(), ...)

Arguments

par A list or numeric vector of starting estimates.
fn A function, least squares of which is to be minimized, with first argument the list of parameters over which minimization is to take place.
jac A function to return the Jacobian for the fn function.
control A list of control parameters. See Details.
... Further arguments to be passed to fn and jac.
Details

Both functions \( fn \) and \( jac \) (if provided) must return numeric vectors. Length of the vector, returned by \( fn \), must not be lower than that of \( par \). Vector, that \( jac \) will return, must have length equal to \( \text{length}(fn(par,\ldots)) \cdot \text{length}(par) \).

The `control` argument is a list that can supply any of the following components:

- **ftol** non-negative numeric. Termination occurs when both the actual and predicted relative reductions in the sum of squares are at most \( \text{ftol} \). Therefore, \( \text{ftol} \) measures the relative error desired in the sum of squares.
- **ptol** non-negative numeric. Termination occurs when the relative error between two consecutive iterates is at most \( \text{ptol} \). Therefore, \( \text{ptol} \) measures the relative error desired in the approximate solution.
- **gtol** non-negative numeric. Termination occurs when the cosine of the angle between result of \( fn \) evaluation \( fvec \) and any column of the Jacobian is at most \( \text{gtol} \) in absolute value. Therefore, \( \text{gtol} \) measures the orthogonality desired between the function vector and the columns of the Jacobian.
- **diag** a list or numeric vector containing positive entries that serve as multiplicative scale factors for the parameters. Length of \( \text{diag} \) should be equal to that of \( \text{par} \). If not, user-provided \( \text{diag} \) is ignored and \( \text{diag} \) is internally set.
- **epsfcn** (used if \( jac \) is not provided) is a numeric used in determining a suitable step for the forward-difference approximation. This approximation assumes that the relative errors in the functions are of the order of \( \text{epsfcn} \). If \( \text{epsfcn} \) is less than the machine precision, it is assumed that the relative errors in the functions are of the order of the machine precision.
- **factor** positive numeric, used in determining the initial step bound. This bound is set to the product of \( \text{factor} \) and the \( |\text{diag} \cdot \text{par}| \) if nonzero, or else to \( \text{factor} \) itself. In most cases \( \text{factor} \) should lie in the interval \((0.1,100)\). 100 is a generally recommended value.
- **maxfev** positive integer. Termination occur when the number of calls to \( fn \) has reached \( \text{maxfev} \).
- **nprint** is an integer, that enables controlled printing of iterates if it is positive. In this case, estimates of \( \text{par} \) are printed at the beginning of the first iteration and every \( \text{nprint} \) iterations thereafter and immediately prior to return. If \( \text{nprint} \) is not positive, no tracing information on the progress of the optimization is produced.

Successful completion.

The accuracy of \( nls.lm \) is controlled by the convergence parameters \( \text{ftol}, \text{ptol}, \text{and gtol} \). These parameters are used in tests which make three types of comparisons between the approximation \( \text{par} \) and a solution \( \text{par}_0 \). \( nls.lm \) terminates when any of the tests is satisfied. If any of the convergence parameters is less than the machine precision, then \( nls.lm \) only attempts to satisfy the test defined by the machine precision. Further progress is not usually possible. The tests assume that \( fn \) as well as \( jac \) are reasonably well behaved. If this condition is not satisfied, then \( nls.lm \) may incorrectly indicate convergence. The validity of the answer can be checked, for example, by rerunning \( nls.lm \) with tighter tolerances.

**First convergence test.**

If \( |z| \) denotes the Euclidean norm of a vector \( z \), then this test attempts to guarantee that

\[
|fvec| < (1 + \text{ftol})|fvec_0|,
\]

where \( fvec_0 \) denotes the result of \( fn \) function evaluated at \( par_0 \). If this condition is satisfied with \( \text{ftol} \approx 10^{-8} \), then the final residual norm \( |fvec| \) has \( k \) significant decimal digits and \( \text{info} \) is set to 1 (or to 3 if the second test is also satisfied). Unless high precision solutions are required, the
recommended value for \( ftol \) is the square root of the machine precision.

**Second convergence test.**

If \( D \) is the diagonal matrix whose entries are defined by the array \( \text{diag} \), then this test attempt to guarantee that

\[
|D(par - par_0)| < ptol |D| \text{par}_0|,
\]

If this condition is satisfied with \( ptol \approx 10^{-k} \), then the larger components of \( (D\text{par}) \) have \( k \) significant decimal digits and \( \text{info} \) is set to 2 (or to 3 if the first test is also satisfied). There is a danger that the smaller components of \( (D\text{par}) \) may have large relative errors, but if \( \text{diag} \) is internally set, then the accuracy of the components of \( \text{par} \) is usually related to their sensitivity. Unless high precision solutions are required, the recommended value for \( ptol \) is the square root of the machine precision.

**Third convergence test.**

This test is satisfied when the cosine of the angle between the result of \( \text{fn} \) evaluation \( fvec \) and any column of the Jacobian at \( \text{par} \) is at most \( gtol \) in absolute value. There is no clear relationship between this test and the accuracy of \( \text{nls.lm} \), and furthermore, the test is equally well satisfied at other critical points, namely maximizers and saddle points. Therefore, termination caused by this test (\( \text{info} = 4 \)) should be examined carefully. The recommended value for \( gtol \) is zero.

**Unsuccessful completion.**

Unsuccessful termination of \( \text{nls.lm} \) can be due to improper input parameters, arithmetic interrupts, or an excessive number of function evaluations.

**Improper input parameters.**

\( \text{info} \) is set to 0 if \( \text{length} (\text{par}) = 0 \), or \( \text{length}(fvec) < \text{length} (\text{par}) \), or \( \text{ftol} < 0 \), or \( \text{ptol} < 0 \), or \( \text{gtol} < 0 \), or \( \text{maxfev} \leq 0 \), or \( \text{factor} \leq 0 \).

**Arithmetic interrupts.**

If these interrupts occur in the \( \text{fn} \) function during an early stage of the computation, they may be caused by an unacceptable choice of \( \text{par} \) by \( \text{nls.lm} \). In this case, it may be possible to remedy the situation by rerunning \( \text{nls.lm} \) with a smaller value of \( \text{factor} \).

**Excessive number of function evaluations.**

A reasonable value for \( \text{maxfev} \) is \( 100 \cdot (\text{length}(\text{par}) + 1) \). If the number of calls to \( \text{fn} \) reaches \( \text{maxfev} \), then this indicates that the routine is converging very slowly as measured by the progress of \( fvec \) and \( \text{info} \) is set to 5. In this case, it may be helpful to force \( \text{diag} \) to be internally set.

**Value**

A list with components:

- **par**
  The best set of parameters found.

- **hessian**
  A symmetric matrix giving an estimate of the Hessian at the solution found.

- **fvec**
  The result of the last \( \text{fn} \) evaluation.

- **info, message**
  \( \text{info} \) is an integer code indicating status of convergence. Explanation of convergence code is stored in the \text{message} component.

  0  Improper input parameters.

  1  Both actual and predicted relative reductions in the sum of squares are at most \( \text{ftol} \).
Relative error between two consecutive iterates is at most ptol.
Conditions for info = 1 and info = 2 both hold.
The cosine of the angle between fvec and any column of the Jacobian is at most gtol in absolute value.
Number of calls to fn has reached maxfev.
ftol is too small. No further reduction in the sum of squares is possible.
ptol is too small. No further improvement in the approximate solution par is possible.
gtol is too small. fvec is orthogonal to the columns of the Jacobian to machine precision.

The result list of diag. See Details.

Note
The public domain Fortran sources of MINPACK package by J.J. More, implementing Levenberg-Marquardt algorithm were downloaded from http://ftp.netlib.org/minpack, and left unchanged.
The contents of this manual page is almost entirely extracted from the comments of MINPACK sources.

References

See Also
optim, nls

Examples
```r
f <- function(T, tau, N0, a, f0) {
  expr <- expression(N0*exp(-T/tau)*(1 + a*cos(f0*T)))
  eval(expr)
}
j <- function(T, tau, N0, a, f0) {
  expr <- expression(N0*exp(-T/tau)*(1 + a*cos(f0*T)))
  c(eval(D(expr, "tau")),
     eval(D(expr, "N0")),
     eval(D(expr, "a")),
     eval(D(expr, "f0")))
}
T <- seq(0, 8, len=501)
p <- c("tau" = 2.2, "N0" = 1000, "a" = 0.25, "f0" = 8)
N <- do.call("f", c(list(T = T), as.list(p)))
N <- rnorm(length(N), mean=N, sd=sqrt(N))
plot(T, N, bg = "black", pch = 21, cex = 0.5)
fcn <- function(p, T, N, N.Err, fcall, jcall) {
  (N - do.call("fcall", c(list(T = T), as.list(p))))/N.Err
}
fcn.jac <- function(p, T, N, N.Err, fcall, jcall) {
```

```
N.Err <- rep(N.Err, length(p))
    -do.call("jcall", c(list(T = T), as.list(p)))/N.Err

guess <- c("tau" = 2.2, "N0" = 1500, "a" = 0.25, "f0" = 10)

out <- nls.lm(par = guess, fn = fcn, #jac = fcn.jac,
    fcall = f, jcall = j,
    T = T, N = N, N.Err = sqrt(N),
    control = list(nprint = 3, diag = numeric()))

N1 <- do.call("f", c(list(T = T), out$par))  # N1 == N - sqrt(N)*out$fvec

lines(T, N1, col="blue", lwd=2)
str(out)

print(sqrt(diag(solve(out$hessian))))  # calculating of SSE
#rm(f, j, fcn, fcn.jac, T, p, guess, N, N1, out)
Index

*Topic nonlinear
   nls.lm, 1
*Topic optimize
   nls.lm, 1

nls, 4
nls.lm, 1

optim, 4