Package ‘marqLevAlg’

March 9, 2022

Type Package

Title A Parallelized General-Purpose Optimization Based on Marquardt-Levenberg Algorithm

Version 2.0.6

Date 2022-03-08

Author Viviane Philipps, Cecile Proust-Lima, Melanie Prague, Boris Hejblum, Daniel Commenges, Amadou Diakite

Maintainer Viviane Philipps <viviane.philipps@u-bordeaux.fr>

Depends R (>= 3.5.0)

Suggests microbenchmark, knitr, rmarkdown, rticles, ggplot2, viridis, patchwork, xtable

LazyLoad yes

LazyData true

Description This algorithm provides a numerical solution to the problem of unconstrained local minimization (or maximization). It is particularly suited for complex problems and more efficient than the Gauss-Newton-like algorithm when starting from points very far from the final minimum (or maximum). Each iteration is parallelized and convergence relies on a stringent stopping criterion based on the first and second derivatives. See Philipps et al, 2021 <doi:10.32614/RJ-2021-089>.

License GPL (>= 2.0)

BugReports https://github.com/VivianePhilipps/marqLevAlgParallel/issues

Imports doParallel, foreach

VignetteBuilder knitr

NeedsCompilation yes

RoxygenNote 7.1.1

Repository CRAN

Date/Publication 2022-03-09 15:20:07 UTC
marqLevAlg-package

A parallelized general-purpose optimization based on Marquardt-Levenberg algorithm

Description

This algorithm provides a numerical solution to the problem of unconstrained local minimization/maximization. This is more efficient than the Gauss-Newton-like algorithm when starting from points very far from the final minimum/maximum. A new convergence test is implemented (RDM) in addition to the usual stopping criterion: stopping rule is when the gradients are small enough in the parameters metric (GH^-1G).

Details

<table>
<thead>
<tr>
<th>Package</th>
<th>marqLevAlg</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>Package</td>
</tr>
<tr>
<td>Version</td>
<td>2.0.6</td>
</tr>
<tr>
<td>Date</td>
<td>2022-03-08</td>
</tr>
<tr>
<td>License</td>
<td>GPL (&gt;= 2.0)</td>
</tr>
<tr>
<td>LazyLoad</td>
<td>yes</td>
</tr>
</tbody>
</table>

This algorithm provides a numerical solution to the problem of optimizing a function. This is more efficient than the Gauss-Newton-like algorithm when starting from points very far from the final maximum. A new convergence test is implemented (RDM) in addition to the usual stopping criterion: stopping rule is when the gradients are small enough in the parameters metric (GH-1G).

Author(s)

Viviane Philipps, Cécile Proust-Lima, Boris Hejblum, Melanie Prague, Daniel Commenges, Amadou Diakite
dataEx

References

marqLevAlg Algorithm
Convergence criteria : Relative distance to Maximum

dataEx Simulated dataset

Description
Sample of 500 subjects simulated according to a linear mixed model. The fixed part of the model included an intercept, 2 natural cubic splines on time and their interactions with time-independent covariates X1, X2 and X3. A random intercept and a independant error term were also inclued.

Format
A data frame with 2429 observations on the following 6 variables.
i subject identification number
t time of measurement
X1 binary covariate
X2 continous Gaussian standard variable
X3 continous Gaussian variable
Y longitudinal outcome

deriva Numerical derivatives

Description
The function computes the first derivates and the information score matrix. Central finite-differences and forward finite-differences are used for the first and second derivatives respectively.

Usage
deriva(nproc = 1, b, funcpa, .packages = NULL, ...)
deriva_grad

Arguments

nproc number of processors for parallel computing
b value of parameters to be optimized over
funcpa function to be minimized (or maximized), with argument the vector of parameters over which minimization is to take place. It should return a scalar result.
.packages character vector of packages that funcpa depends on
... other arguments of the funcpa function

Value

v vector containing the upper part of the information score matrix and the first derivatives
r1 the value of the funcpa function at point b

Author(s)

Viviane Philipps, Boris Hejblum, Cecile Proust-Lima, Daniel Commenges

References


Examples

b <- 0.1
f <- function(b){return((2*b[1]**2+3*b[1]))}
d <- deriva(b=b,funcpa=f)

deriva_grad Numerical derivatives of the gradient function

Description

The function computes the information score matrix in the case where the first derivatives of the function to optimize are analytically known. Therefore, minus the derivatives of the gradient are computed by central finite differences.

Usage

deriva_grad(nproc = 1, b, grad, .packages = NULL, ...)
**gradLMM**

**Arguments**

- `nproc` number of processors for parallel computing
- `b` value of parameters to be optimized over
- `grad` the gradient of the function to be minimized (or maximized)
- `.packages` character vector of packages that grad depends on
- `...` other arguments of the grad function

**Value**

- `hessian` vector containing the upper part of the information score matrix

**Author(s)**

Viviane Philipps, Boris Hejblum, Cecile Proust-Lima, Daniel Commenges

---

| gradLMM | Gradient of the log-likelihood of a linear mixed model with random intercept |

**Description**

Gradient of the log-likelihood of a linear mixed model with random intercept

**Usage**

`gradLMM(b, Y, X, ni)`

**Arguments**

- `b` numeric vector specifying the parameter’s values in the following order: first the fixed effects and then the standard deviation of the random intercept and of the independent error
- `Y` numeric vector including the dependent outcome vector ordered by subject
- `X` numeric matrix including the covariates
- `ni` integer vector giving the number of repeated measures for each subject

**Value**

a vector containing the gradient of the log-likelihood of the linear mixed model at point b
loglikLMM

Log-likelihood of a linear mixed model with random intercept

Description

Log-likelihood of a linear mixed model with random intercept

Usage

loglikLMM(b, Y, X, ni)

Arguments

b	numeric vector specifying the parameter’s values in the following order: first
the fixed effects and then the standard deviation of the random intercept and of
the independent error

Y
numeric vector including the dependent outcome vector ordered by subject

X
numeric matrix including the covariates

ni
integer vector giving the number of repeated measures for each subject

Value

the log-likelihood of the linear mixed model at point b

marqLevAlg

A parallelized general-purpose optimization based on Marquardt-Levenberg algorithm

Description

This algorithm provides a numerical solution to the problem of unconstrained local optimization. This is more efficient than the Gauss-Newton-like algorithm when starting from points very far from the final maximum. A new convergence test is implemented (RDM) in addition to the usual stopping criterion: stopping rule is when the gradients are small enough in the parameters metric (GH-1G).

Usage

marqLevAlg(
    b,
    m = FALSE,
    fn,
    gr = NULL,
    hess = NULL,
    maxiter = 500,
)
marqLevAlg

    epsa = 1e-04,
    epsb = 1e-04,
    epsd = 1e-04,
    partialH = NULL,
    digits = 8,
    print.info = FALSE,
    blinding = TRUE,
    multipleTry = 25,
    nproc = 1,
    clustertype = NULL,
    file = "",
    .packages = NULL,
    minimize = TRUE,

)

mla(
    b,
    m = FALSE,
    fn,
    gr = NULL,
    hess = NULL,
    maxiter = 500,
    epsa = 1e-04,
    epsb = 1e-04,
    epsd = 1e-04,
    partialH = NULL,
    digits = 8,
    print.info = FALSE,
    blinding = TRUE,
    multipleTry = 25,
    nproc = 1,
    clustertype = NULL,
    file = "",
    .packages = NULL,
    minimize = TRUE,

)

Arguments

b        an optional vector containing the initial values for the parameters. Default is 0.1
         for every parameter.

m        number of parameters. Optional if b is specified.

fn       the function to be optimized, with first argument the vector of parameters over
         which optimization is to take place (argument b). It should return a scalar result.

gr       a function to return the gradient value for a specific point. If missing, finite-
         difference approximation will be used.
marqLevAlg

hess  a function to return the hessian matrix for a specific point. If missing, finite-difference approximation will be used.

maxiter  optional maximum number of iterations for the marqLevAlg iterative algorithm. Default is 500.

epsa  optional threshold for the convergence criterion based on the parameter stability. Default is 0.0001.

epsb  optional threshold for the convergence criterion based on the objective function stability. Default is 0.0001.

epsd  optional threshold for the relative distance to maximum. This criterion has the nice interpretation of estimating the ratio of the approximation error over the statistical error, thus it can be used for stopping the iterative process whatever the problem. Default is 0.0001.

partialH  optional vector giving the indexes of the parameters to be dropped from the Hessian matrix to define the relative distance to maximum/minimum. If specified, this option will only be considered at iterations where the two first convergence criteria are satisfied (epsa and epsb) and if the total Hessian is not invertible. By default, no partial Hessian is defined.

digits  number of digits to print in outputs. Default value is 8.

print.info  logical indicating if information about computation should be reported at each iteration. Default value is FALSE.

blinding  logical. Equals to TRUE if the algorithm is allowed to go on in case of an infinite or not definite value of function. Default value is FALSE.

multipleTry  integer, different from 1 if the algorithm is allowed to go for the first iteration in case of an infinite or not definite value of gradients or hessian. As many tries as requested in multipleTry will be done by changing the starting point of the algorithm. Default value is 25.

nproc  number of processors for parallel computing

clustertype  one of the supported types from makeCluster

file  optional character giving the name of the file where the outputs of each iteration should be written (if print.info=TRUE).

.packages  for parallel setting only, packages used in the fn function

minimize  logical indicating if the fn function should be minimized or maximized. By default minimize=TRUE, the function is minimized.

...  other arguments of the fn, gr and hess functions

Details

Convergence criteria are very strict as they are based on derivatives of the objective function in addition to the parameter and objective function stability. In some cases, the program may not converge and reach the maximum number of iterations fixed at 500. In this case, the user should check that parameter estimates at the last iteration are not on the boundaries of the parameter space. If the parameters are on the boundaries of the parameter space, the identifiability of the model should be assessed. If not, the program should be run again with other initial values, with a higher maximum number of iterations or less strict convergence tolerances. An alternative is to remove some parameters from the Hessian matrix.
marqLevAlg

Value

- **cl**: summary of the call to the function marqLevAlg.
- **ni**: number of marqLevAlg iterations before reaching stopping criterion.
- **istop**: status of convergence: =1 if the convergence criteria were satisfied, =2 if the maximum number of iterations was reached, =4 if the algorithm encountered a problem in the function computation.
- **v**: vector containing the upper triangle matrix of variance-covariance estimates at the stopping point.
- **grad**: vector containing the gradient at the stopping point.
- **fn.value**: function evaluation at the stopping point.
- **b**: stopping point value.
- **ca**: convergence criteria for parameters stabilisation.
- **cb**: convergence criteria for function stabilisation.
- **rdm**: convergence criteria on the relative distance to minimum (or maximum).
- **time**: a running time.

Author(s)

Melanie Prague, Viviane Philipps, Cecile Proust-Lima, Boris Hejblum, Daniel Commenges, Amadou Diakite

References

**marqLevAlg Algorithm**

Donald W. marquardt

**Convergence criteria : Relative distance to Minimum (or Maximum)**


Examples

```r
### example 1
### initial values
b <- c(8,9)
### your function
f1 <- function(b){
  return(-4*(b[1]-5)^2-(b[2]-6)^2)
}
### gradient
g1 <- function(b){
  return(c(-8*(b[1]-5),-2*(b[2]-6)))
}
## Call
```
test1 <- mla(b=b, fn=f1, minimize=FALSE)

## Not run:
microbenchmark::microbenchmark(mla(b=b, fn=f1, minimize=FALSE),
    mla(b=b, fn=f1, minimize=FALSE, nproc=2),
    mla(b=b, fn=f1, gr=g1, minimize=FALSE),
    mla(b=b, fn=f1, gr=g1, minimize=FALSE, nproc=2),
    times=10)

## End(Not run)

### example 2
## initial values
b <- c(3,-1,0,1)
## your function
f2 <- function(b){
    return(-((b[1]+10*b[2])^2+5*(b[3]-b[4])^2+(b[2]-2*b[3])^4+10*(b[1]-b[4])^4))
}
## Call
test2 <- mla(b=b, fn=f2, minimize=FALSE)
test2$b

test2_par <- mla(b=b, fn=f2, minimize=FALSE, nproc=2)
test2_par$b

## Not run:
microbenchmark::microbenchmark(mla(b=b, fn=f2, minimize=FALSE),
    mla(b=b, fn=f2, minimize=FALSE, nproc=2),
    times=10)

## End(Not run)

## Not run:
### example 3 : a linear mixed model
## the log-likelihood is implemented in the loglikLMM function
## the gradient is implemented in the gradLMM function

## data
Y <- dataEx$Y
X <- as.matrix(cbind(1, dataEx[, c("t","X1","X3")], dataEx$t*dataEx$X1))
ni <- as.numeric(table(dataEx$i))

## initial values
binit <- c(0,0,0,0,0,1,1)

## estimation in sequential mode, with numeric derivatives
estim <- marqLevAlg(b=binit, fn=loglikLMM, minimize=FALSE, X=X, Y=Y, ni=ni)
## estimation in parallel mode, with numeric derivatives
estim2 <- marqLevAlg(b=binit, fn=loglikLMM, minimize=FALSE, X=X, Y=Y, ni=ni, nproc=2, clustertype="FORK")
## estimation in sequential mode, with analytic gradient
estim3 <- marqLevAlg(b=binit, fn=loglikLMM, gr=gradLMM, minimize=FALSE, X=X, Y=Y, ni=ni)
## estimation in parallel mode, with analytic gradient
estim4 <- marqLevAlg(b=binit, fn=loglikLMM, gr=gradLMM, minimize=FALSE, X=X, Y=Y, ni=ni, nproc=2, clustertype="FORK")
## End(Not run)

---

**print.marqLevAlg**  
*Summary of a marqLevAlg object*

### Description

The function provides a summary of a marqLevAlg optimisation.

### Usage

```r
## S3 method for class 'marqLevAlg'
print(x, digits = 8, ...)
```

### Arguments

- `x`  
a marqLevAlg object.
- `digits`  
Number of digits to print in outputs. Default value is 8.
- `...`  
other (unused) arguments.

### Author(s)

V. Philipps, C. Proust-Lima, B. Hejblum, D. Commenges, M. Prague, A. Diakite

### See Also

link{summary.marqLevAlg}

### Examples

```r
f1 <- function(b){
  return(4*(b[1]-5)^2+(b[2]-6)^2)
}

test.marq <- marqLevAlg(b=c(8,9), m=2, maxiter=100, epsa=0.001, epsb=0.001, epsd=0.001, fn=f1)

test.marq
```
Summary of optimization

Description
A short summary of parameters estimates by marqLevAlg algorithm.

Usage
```r
## S3 method for class 'marqLevAlg'
summary(object, digits = 8, loglik = FALSE, ...)
```

Arguments
- **object**: a marqLevAlg object
- **digits**: Number of digits to print in outputs. Default value is 8.
- **loglik**: Logical indicating if the objective function is a log-likelihood. By default, loglik=FALSE.
- **...**: other (unused) arguments

Value
A data frame containing as many rows as estimated parameters. If loglik=FALSE, it includes one column containing the estimated parameters values. If loglik=TRUE, it includes 6 columns: the estimated parameters, their standard errors, the corresponding Wald statistic, the associated p-value and the boundaries of the 95% confidence interval.

Author(s)
V. Philipps, C. Proust-Lima, B. Hejblum, D. Commenges, M. Prague, A. Diakite

See Also
- marqLevAlg
- print.marqLevAlg

Examples
```r
f1 <- function(b){
    return(4*(b[1]-5)^2+(b[2]-6)^2)
}

test.marq <- marqLevAlg(b=c(8,9),m=2,maxiter=100,epsa=0.001,epsb=0.001,
                         epsd=0.001,fn=f1)

summary(test.marq)
```
Index

* **algorithm**
  marqLevAlg-package, 2

* **datasets**
  dataEx, 3

* **marqLevAlg**
  marqLevAlg-package, 2

* **maximisation**
  marqLevAlg-package, 2

* **optimization**
  marqLevAlg-package, 2

* **package**
  marqLevAlg-package, 2

* **print**
  print.marqLevAlg, 11

* **summary**
  summary.marqLevAlg, 12

dataEx, 3
deriva, 3
deriva_grad, 4

gradLMM, 5

loglikLMM, 6

makeCluster, 8
marqLevAlg, 6, 12
marqLevAlg-package, 2
mla (marqLevAlg), 6

print.marqLevAlg, 11, 12
summary.marqLevAlg, 12