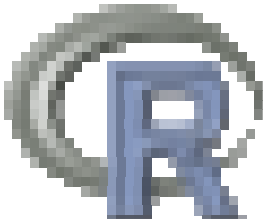
Updating & Improving optim(): Unifying optimization algorithms in R for smooth, nonlinear problems

John C. Nash
University of Ottawa, Telfer School of Management
Ottawa, Canada

Ravi Varadhan
The Center on Aging and Health,
Johns Hopkins University, Baltimore, USA.



1. Our work addresses 4 different areas: (a) unification of existing tools (optimx), (b) updating optim, (c) providing guidance to useRs on choosign appropriate algorithms, as well as on problem formulation, scaling etc. (GUIDED), and (d) benchmarking algorithms and comparative performance evaluation (runopt)

2. Given that unification of optimziation tools (for box-constrained problems) is the main theme for our talk, our focus is primarily on optimx(). The main question(s) to address is (are): why is optimx() needed? How can useRs benefit from it? We should demonstrate, with a couple of examples, why optimx() is the "go-to" place for smooth, box-constrained nonlinear optimization.

3. After demonstrating the utility of optimx(), we can sketch our plans for the future, i.e. how we are planning to address 1(b) - 1(d).

4. Finally, we can close with a slide or two soliciting suggestions/ideas on how to address some specific critical issues with regards to 1(a) - 1(d).

Why?

- R offers a very powerful and convenient interface to many statistical tools
- Including optimization, nonlinear least squares, and nonlinear equations

BUT ...

- Too many, very similar tools
- “Old” tools or at least old implementations
- Confusion over which to use. Which is “better”?

e.g. variable metric method

- optim(..., method='BFGS')
  - Nash (1990) Pascal --> C (B. Ripley), no bounds (but ...)
- optim(..., method='L-BFGS-B')
  - Byrd _et. al._ (1995), not same algorithm, bounds
- ucminf( )
  - Nielsen/Mortensen. Seems to be Fortran VM code.

And for good measure, some new ones --

- Rvmmin0 and Rvmminb
  - my own “all R” versions without and with bounds

A (partial?) list

optim == Nelder-Mead, BFGS, L-BFGS-B, CG (with FR, PR, BS variants), SANN

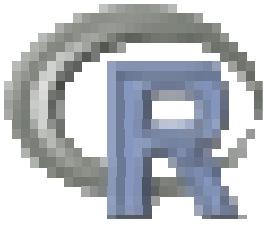
nlm, nlminb, powell, ucminf, MaxLik, BB::spg

trust, cleversearch, DEoptim, rgenoud

Rdonlp2, ConstrOptim

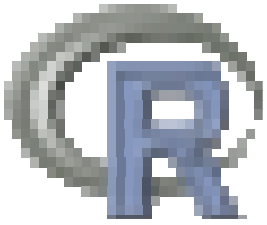
---

and many local choices



Why so many choices?

- Details can be important
- Problems are often “nasty”
  - scale varies hugely across parameter space
  - singularity of Hessian
- Problems vary in size and complexity
  - scaling
  - constraints
  - availability of derivative information
- Human nature



Our Objectives

- unify optimization tools in R
  - priority: smooth, nonlinear, box-constrained optimization problems;
- provide "guidance" to users (automated?)
  - choosing appropriate tools (inc. nls etc. if indicated)
  - setting up function and call
- need evidence as basis for advice (runopt)
- update/extend R optimization methods & tools
  - especially tools / interfaces

optimx() Illustration

- Petran-Ratkowsky problem **
  - difficulties with convergence of nls and optim
  - Which method to use?

```
ourres<-optimx(par=c(10,0.01,4,10),fn=RSS,method=c('BFGS','spg','nlm'))
ourres

par    fvalues method
1 10.05291619, 0.04146914, 4.15651522, 9.83597299 0.02468974   BFGS
2 16.29893736, 0.03223693, 4.16816366, 9.84166213 0.02465116    nlm
```

  - Quick and dirty way to try different methods on possibly difficult problem

** Marie Laure Delignette-Muller

Another example

ans<-optimx(start,fn=broydt.f,gr= broydt.g,
method=c('nlm','nlminb','BFGS',"Nelder","CG","ucminf","SANN","L-BFGS-B","spg"))

```
> ans
                                                                      par
5                                                                4, 4, 4, 4, 4, 4
6 5.2260723, 1.5111481, -0.4172068, -0.9248976, -0.8926013, -0.5756397
7 5.2260677, 1.5111560, -0.4172003, -0.9248946, -0.8926011, -0.5756397
1 5.2260728, 1.5111500, -0.4172051, -0.9248977, -0.8926029, -0.5756406
2 5.2260730, 1.5111498, -0.4172052, -0.9248978, -0.8926030, -0.5756406
4 5.2260730, 1.5111498, -0.4172052, -0.9248978, -0.8926030, -0.5756406
3 5.2260730, 1.5111498, -0.4172052, -0.9248978, -0.8926030, -0.5756406
       fvalues   method   fns grs itns conv  KKT1 KKT2
5          206     SANN 10000  NA NULL    0 FALSE TRUE
6 6.750998e-11 L-BFGS-B    47  47 NULL    0 FALSE TRUE
7 6.154014e-11      spg   158  NA  143    0 FALSE TRUE
1 7.022921e-14      nlm    NA  NA   51    0  TRUE TRUE
2 8.030451e-16   nlminb    53  43   42    0  TRUE TRUE
4 8.515444e-18   ucminf    44  44 NULL    0  TRUE TRUE
3 2.531347e-19     BFGS   109  51 NULL    0  TRUE TRUE
```

optimx() outline

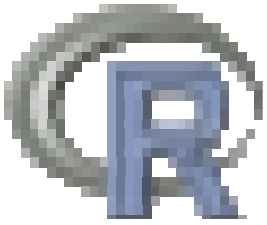
- Checks
  - parameter structure, and, if supplied, bounds, gradients, and Hessians
  - methods suitable to inputs e.g., bounds
- Methods – multiple via a list
- Post solution analysis – KKT conditions
- Attempt to make things “nice” for user while keeping optim()-style calling syntax

Niceties

- control$maximize – if TRUE maximizes
  - avoids “fiddle” of fnscale = -1
- control$follow.on – if TRUE use last set of parameters of one method as start in next
  - allows polyalgorithms tailored to needs
- KKT post-solution analysis
  - Are we “there” yet?
  - “Termination” not “convergence”

Possibilities?

- Hooks for calling local R source methods
  - allows bleeding-edge tools to be applied
  - allows local modificaitons for special purposes, such as special constraints or instrumentation of method
  - helps standardize calling syntax (inc. our own methods!)
  - already have prototype working
- Include other tools – trust?, cleversearch?
- Include “new” tools (Rcgmin, BOBYQA)



Related activity -> other goals

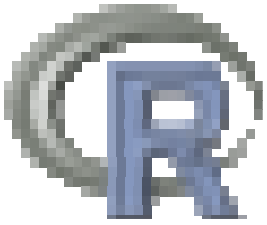
- Wiki to share work in progress – ask for access
- R-forge “OptimizeR” for packages that “work”
- funcheck / funtest packages (already working)
  - same functionality, depending on function & file names
  - “standardized” test file structure
  - NISTnls functions partially converted to this structure
  - supports optimx() with common R code where suited
- runopt() - for performance data (alpha stage)
  - MUST be simple, and gather date from many machines
  - issues of platform / problem / method characteristics

Purpose of runopt()

- Build base on which to give advice
  - Automatic gathering / Public data repository
- Make tools more consistent and easier to use
- Provide a framework for continuous improvement
  - Easier/unified interface to existing tools
  - But without causing upset to legacy applications
  - Gradually add interface features
  - Look for tools that will help users get good results more easily e.g., Automatic Differentiation for gradient function

Issues – UseR input?

- Scaling – via parscale or explicit in R code?
  - parscale may impose inefficiencies / error-prone
  - Can tools help generate scaled code automatically?
- Mandatory bounds (box-constraints)
  - Force user to think of the scaling and "reality" -- avoids inadmissible answers
  - "Number of grain elevators in Saskatchewan"



Issues – UseR input (2) ?

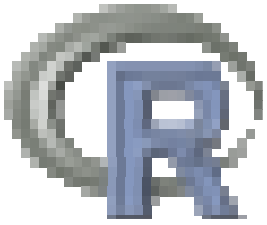
- Development & tuning of a 'GUIDED' method
  - GUI programming / Can it be simply scripted?
  - Starting values
  - Links to nls etc.– focus on problems, not methods
  - More templates and examples
- runopt() -- as mentioned, need participants
- Fortran/C/etc. -- less ad hoc, more review
- Feedback and continuous improvement
- Derivatives / automate function building

Progress report

- optimx() is at beta stage
- runopt() at alpha
  - But we need more, and better checked, test files
  - And we need to ensure automated data gathering is bullet-proof
  - Profiling still in early stages
  - Documentation and facilities for allowing data to be analysed – and results reported
- funcheck and funtest – alpha/beta boundary

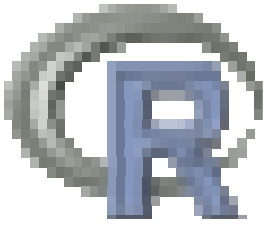
Progress report (2)

- Wiki up and running and some external participations (reminder: ask for access)
- OptimizeR is up, but so far mostly pre-existing packages have been loaded
  - Optimx beta up; runopt, etc. “soon”
- Some interaction with others on ideas relating to Automatic and Symbolic Differentiation
  - Still a long way to go to get easy-to-use tools



Progress report (3)

- Getting new Powell BOBYQA to run with R
  - Fortran 77 code, Uses local printing, lots of local storage
  - Help welcome! Want to build general “how to”.
- Several all-R codes running in alpha state
- “GUIDED” tools outlined, but not programmed
- So far have not done much re:
  - ensuring underlying apps. (nnet, arima, etc.) can benefit
  - generalizing to include nls etc. i.e., more problem focus



THANKS!

Contact info:

nashjc _at__ uottawa.ca

RVaradhan __at__ jhmi.edu

Questions?

Extra topics

The following slides are intended to augment the brief exposition in the main presentation.

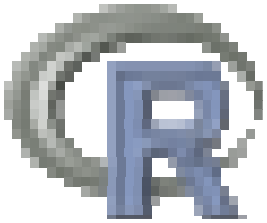
Scaling and why it “hurts”

hobbs.r: 12 data points to be fitted to

$$y \sim x_1/(1+x_2*\exp(-x_3*t)) \quad \text{(3 parameter logistic)}$$

Function base = 23520.58 at 1 1 1
Percent changes for 1 % change in each parameter are *0.03503 0.00020 0.00046*

Function base = 2.587542 at 196.5079544 49.1138533 0.3133611
Percent changes for 1 % change in each parameter are *94.117 39.695 391.27*
Hessian eigenvalues -- unscaled function
At start: 41.618914 16.635191 -3.700846 (INDEFINITE) Ratio -11.24579
At solution: 2.047414e+06 4.252238e-01 4.376540e-03 Ratio 467815596



“Simple” rescaling

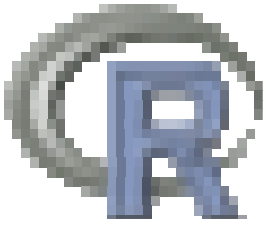
$$y \sim 100\, x_1/(1+10\, x_2 * \exp(-0.1\, x_3 * t))$$

Function base = 23520.58 at [1] 0.01 0.10 10.00
Percent changes for 1 % change in each parameter are *0.03503 0.00020 0.00046*

Function base = 2.587543 at 1.965080 4.911385 3.133611
Percent changes for 1 % change in each parameter are *94.112 39.698 391.26*

No change. This is as it should be!

Hessian eigenvalues -- scaled function
At start: 223294.0 .5599862 -204.9109 (INDEFINITE) Ratio 398749.1
At solution: 33859.37019 76.55200 14.70142 Ratio 2303.137

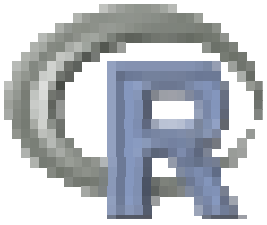


Scaling in optim() ?

- parscale and fnscale (<0 maximizes function)
- Quite a lot of overhead in code
  - argument passing, many multiplications and divisions
  - adds to work of package builders
  - more difficult to make optimx call other minimizers
  - Should we not get the user to scale his/her function?
- How can we help with such scaling?
- OR ... put the scaling in functions explicitly
  - Still need to provide help

Guiding the user

- Scale so solution has parameters in [1,10] in magnitude
- Provide bounds (even if we don't use them)
  - avoid nasties (negative grain elevators in Saskatchewan)
  - users almost always can get us within an order of magnitude
- If derivatives available, provide them
  - Make SURE they are correct (now part of optimx)
  - Important for dispersion information



The getting of wisdom

- runopt.R – a first try to get performance data
- Use scponly tool on server to allow uploads
  - ssh keypairs, 1 for each “user”
  - no shell functions, only scp
  - build scp calls into runopt.R
- runopt.R set up to execute various tests
  - many details still being developed
- Need to get machine/OS data
- Nice to get performance profiles

Performance profiles

- ref??
- example??

runopt() infrastructure

- Sets of test functions (with data)
- name.R is file, name.xxx provides functions

  xxx takes on

  - f, res, jac, g, h, rsd, fgh, doc, setup, test
  - function, residual, jacobian, gradient, hessian, residual second derivatives, combined function+gradient+hessian, documentation, setup, testing
  - Not all functions for all test functions (some are not sums of squares)
- funcheck.R – test such files using numDeriv

funtest.R

- Similar to funcheck.R, but for cases where functions have naming scheme other than that in our “standard”, i.e., runopt() compatible, test functions
- Intended to help users build functions
- Would like to extend to (G)UI that helps construct the appropriate R file and calls

Methods in R code only

- C and Fortran may hide what is going on
- And are more difficult to “document”
- May win with eventual R compiler (??)
- May already be fast enough, especially for vectorized code
    - BUT we need to test such ideas

All-R methods

- new Rcgmin with Yuan-Dai restart patch
  - example timings
- Rvmmin – to test different line search strategies
- SNewton – safeguarded Newton method
  - Should be “like” nlm, but ...
- xxxxxxb versions to handle bounds
- Important to provide extensive commentary if we want to learn/improve