

# Nonlinear parameter optimization and modeling in R

John C. Nash  
Retired professor  
Telfer School of Management  
University of Ottawa  
[nashjc@uottawa.ca](mailto:nashjc@uottawa.ca)

# Why?

- Most common statistical tasks *should* have appropriate optimization embedded

BUT

- There may be constraints
- We may need to “adjust” the objective function
- We may want to check that methods actually do optimize as we understand they should
- New problems arise that have no package
- We may need to speed things up

# What?

- Outline the main problems we seek to solve
- Overview of (some) packages available and their strengths and weaknesses
- Review importance of getting the setup right
  - Functions, derivatives, constraints, starting points

But

- Keep tone relatively simple and interactive
- Main focus on fitting models & interpreting results
- Use examples

# Problem types

- Making model approximate the data in some fashion
  - Traditional least squares
  - Other loss functions (robust, approximations, ...)
- Operations research problems
  - Not traditional statistics, but may be part of our work

# The formal problem(s)

Find  $\mathbf{x} = \operatorname{argmin} f(\mathbf{x})$  s.t.  $\mathbf{q}(\mathbf{x}) \geq \mathbf{0}$  (NLP)

NOT very common (yet?) in statistics

General version not prominent in this tutorial

BUT variants / special cases of this problem are very important and common in statistics

Unconstrained problems or box-constrained ones

Special forms e.g., sums of squares (NLS)

Nonlinear equations  $\mathbf{g}(\mathbf{x}) = \mathbf{0}$

NOTE: We will “minimize” functions. Maximization is by  $\min(-f(\mathbf{x}))$

# Or ...

- Writing an objective function properly and checking it
- Scaling and constraints
- Starting values
- Understanding the output. Did I get a good answer?
- Common error messages and how to address them?
- Which package(s) should I use?

# Caveats

- Some (most?) of you know
  - R better than I, or
  - more stats than I, or
  - more about some optimization tools than I
- Our problems shape our skills and views
- I like to “make things work”
  - s/w design and usability as much as application
  - Prefer clean, clear methods (but sometimes one needs complicated, messy ones)

# Digression on expectations

- My experience is that tools in R are
  - Very good, but
  - Not nearly as reliable or foolproof as most users expect or hope
  - May work brilliantly on one problem and miserably on the next
  - Often badly- or mis-applied
- Paranoia and verification are **VERY** useful



# My R Interests

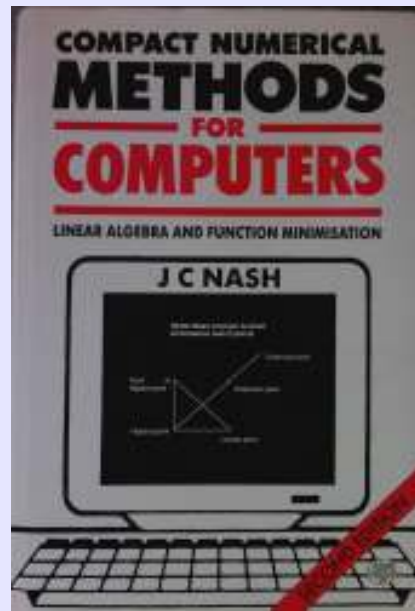
- Robust numerical and optimization tools
  - Keep trying, and trying, and trying, ...
  - May not be very fast
  - No misleading answers
- Informative software
  - Tells user what is happening, why, etc.
- Simplify the user experience
- Software that allows improvement / extension

# My background – in books

- 1979, 1990: CNM -- also Japanese & CD versions
- 1984: ESPSSC; also revised CD version
- 1987: JNMWS (1989 problem extensions)
- 2001: PFM
- 2014: NLPOR



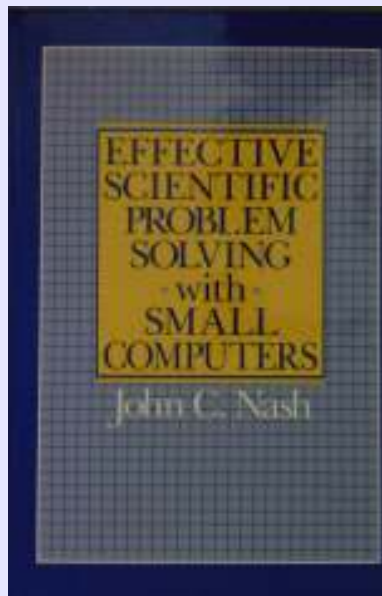
1979



1990



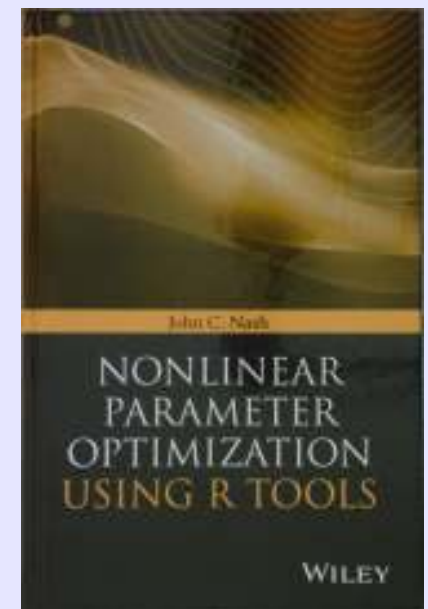
1987



1984



2001



2014

# Example problems (likely oversimplified)

- 1) Hobbs weed infestation
- 2) Two straight lines
- 3) Multiple exponentials
- 4) Maximum likelihood – Ajay Shah has a useful intro.

<http://www.mayin.org/ajayshah/KB/R/documents/mle/mle.html>

- 5) Polynomial approximation

But different side conditions change problems

- Minimax vs. Least squares
- Constraints
- Need to solve many similar problems

# Example 1: Hobbs weed infestation

- A real problem
- Has least squares and likelihood versions
- Nasty! Yet looks straightforward
- Illustrates issues with Hessian

==> example “hobbs”

# Example 2: Two straight lines

- Real and artificial possibilities
- Various approaches to solution

==> example “twosl”

# Example 3: Multiple exponentials

- A classic problem
- Exquisitely difficult to get “true” parameters to artificial problem (Lanczos)
- May present in several ways

==> example “lanczos”

# Example 4: Maximum Likelihood

- Importance of getting function correct
  - Test it!
  - Negative log likelihood is minimized
- Gradients?
- Scaling and constraints need attention

==> example “maxlik”



# Example 5: Polynomial approximation

- Find a polynomial approximation to a function or set of data
- Least squares  $\implies$  linear problem
- BUT approximation is usually minimax

$\implies$  example “polyfit”

# Other examples

- Distribution fitting
- ?? others to add – provide files for examples
- ?? add vignettes

# Conclusions

for those wanting to be first to the bar

- Most methods work most of the time
  - Generally quite easy to use, but setup of objective function often very user-specific
  - BUT ... all of them fail sometimes
  - Swapping methods takes work (BUT see ***optimx***)
  - Needed “extra” computations may be missing and awkward to supply, especially “nicely”
  - AND ... everyone wants to make a better optimizer (but only for the problem at hand)
  - So there are (too!) many choices

# Conclusions 2

for those wanting to be first to the bar

- Methods are iterative
  - Choice of start can be critical
    - To performance
    - To success – getting the “right” answer
  - Convergence vs. Termination
    - Algorithms converge
    - Programs terminate
    - Neither may be at the “right” solution
  - The “right” answer is usually the one the user knows is right

# My Own View

- Optimization tools are extremely useful
- But take work and need a lot of caution
- R is the best framework I have found for exploring and using optimization tools
  - I prefer it to MATLAB, GAMS, etc.
  - No problem has yet proved impossible to approach in R, but much effort is needed
- Still plenty of room for improvement in R
  - Methods; Interfaces, Documentation; User Ed.

# The formal problem (reminder)

Find  $\mathbf{x} = \operatorname{argmin} f(\mathbf{x})$  s.t.  $\mathbf{q}(\mathbf{x}) \geq \mathbf{0}$  (NLP)

# Characterizations of problems

## Number of constraints:

Lots: Math programming

Few: Function minimization (my main experience)

## Number of solutions

none (no feasible solution due to constraints)



one and only one



several



many



plateaux and saddles



Multiple real or "near"  
minima very frequently  
a source of difficulties  
==> global optimization

# Characterizations of problems (2)

By smoothness or reproducibility of function

By math / algorithmic approach to solution

- Descent method (gradient based)

- Newton approach (Hessian based)

- Direct search, but “derivative-free” methods may implicitly use gradient ideas

By statistical versus optimization viewpoint

(not so?) Side-note: Deciding if we have a solution

- KKT conditions

- Other indicators



# KKT conditions (local min.)

## Gradient is zero

Hilltop, valley bottom or saddle?

How small is zero?

## Hessian is positive-definite

Curvature is “upwards”  $\implies$  valley bottom

How do we decide? Posi

Issues are tolerances for small gradient and negative/zero eigenvalues of Hessian

Complications of scaling

# Other indicators

- Grid search
  - e.g., cleversearch in svcm package
    - But see code – max [3D], references, 2009 date?
- Axial search ( + / - steps only)
  - Tilt and curvature
- Dispersion estimates. With constraints?
- Room for more work on good “indicators”

*“Are we there yet?”*

# Global optima

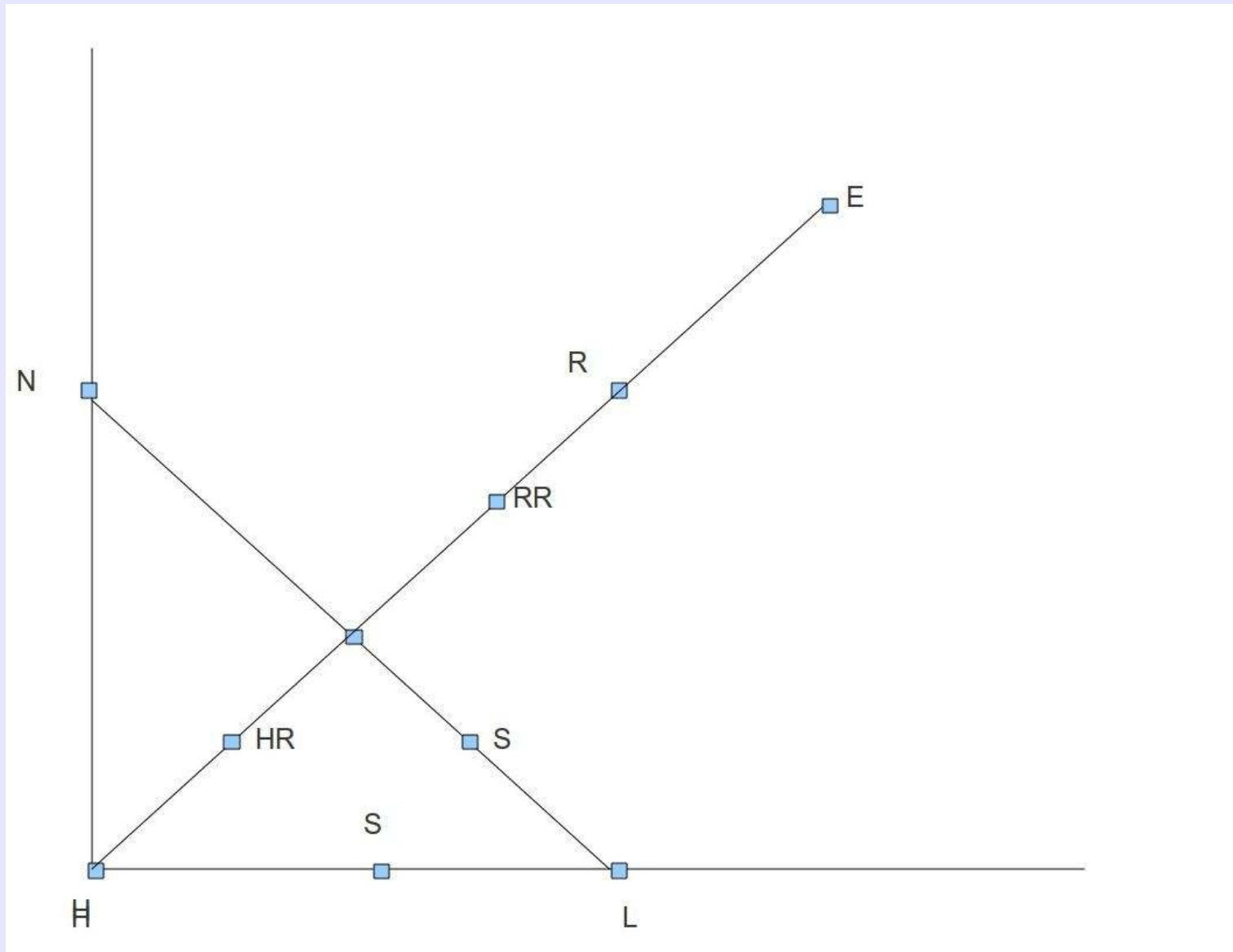
- What users want
- What they (almost) never get!
- Mathematical conditions for global optimum rarely available, and computational implementations even less so, i.e., Lipschitz conditions using bounds on gradients

e.g., [http://ab-initio.mit.edu/wiki/index.php/NLopt\\_Algorithms](http://ab-initio.mit.edu/wiki/index.php/NLopt_Algorithms)

# Minimization methods

- Heuristic
- Stochastic
- Newton based
- Descent based
  
- Mixtures of the above

# Heuristic – e.g. Nelder Mead



# Stochastic

- DEOptim, optim(SANN)
  - Caution about termination!
- Generally combine with heuristics
- When are we there?
- 
- 
- ?? ref 2 nlpor??

# Newton based

- At  $\mathbf{x}$  find  $\mathbf{g}$  and  $\mathbf{H}$ , solve  $\mathbf{H} \mathbf{delta} = -\mathbf{g}$
- Then  $\mathbf{x}_{new} = \mathbf{x} + \mathbf{delta}$
- BUT
  - $\mathbf{H}$  is costly of space, time, human effort
  - Quadratic convergence if “close enough”
  - Else line search, acceptable point, trust region
- Try to approximate  $\mathbf{H}$ 
  - Gauss Newton, Quasi Newton, Truncated ...

# Newton based 2

- Quasi-Newton / Variable Metric
- Start with approx  $\mathbf{H}$  or its inverse
- Take a step then “update” -- use gradients only
- MANY variants (inverse, direct, update formulae, line searches, heuristics, ....)
- BFGS uses inverse and acceptable point
- L-BFGS-B uses inverse in different way
- Not sure of nlm or nlminb internals



# Descent

- Cauchy – steepest descents
- Want “conjugate” search directions
  - Gram Schmidt ideas
  - CG ideas, especially “restarting”
  - Spectral gradient (spg from BB)
- Combine with Newton ideas ...

# R view of optimization problems

## Expressions (as in `nls`)

$y \sim a1 / (1 + a2 * \exp(- a3 * t))$  [parameters `a1,a2,a3`]

Mainly least squares problems.

BUT: Not all sums of squares are from expressions

## Functions (as in `optim` and `descendents`)

```
objfn <- function(x, ...) {  
    (code)  
    something <-.....  
    return( something )  
}
```

# Strategic issues in methods

## Single algorithms, possibly safeguarded

allows package developers to match to problems

“user-developed” specialized polyalgorithms

requires user knowledge for best use

## All-purpose polyalgorithms (“always work”)

possibly approach of nlminb and nlm creators

difficult to understand / debug

may be best suited to experience/style of the creators

*JN leans more to first style, esp. for R packages*

# Tactical choices for R

- Interface existing codes in Fortran or C or ...
  - R  $\leftrightarrow$  C ( $\leftrightarrow$  Fortran) for each fn evaluation!!!
  - Difficult to debug
  - .Call, .Fortran, SEXP's etc.
  - Rcpp for C++ (another layer)
- Or all in R
  - Need to vectorize for best performance
  - Improvements to R interpreter/compiler?
  - Easier to understand and debug if all in R
  - Not necessarily so slow (but more tests needed)

# Some problems

- **Hobbs** – where I came in (1974) -- HV.Rnw
- Cobb Douglas – CD.Rnw
- Exponential fits: shifted and multiple
  - optex.R, expontest.Rnw, bvlstest.R
- Distribution fits – PoissLikJO.Rnw
- Large n problems
  - Rayleigh Quotient eigprob.Rnw
  - Artificial tests artificial.Rnw
- Non-smooth / imprecise

# Looking at problems: an idealized agenda for each

- The obvious approach
- Can we do better?
- Discussion of the issues?
- Recommendations
- Preferably avoiding “you ought to use ....”
- Hopefully using methods that are familiar, at least in how they are used and what they return

# Hobbs problem

- Weed infestation over 12 seasons
- Initial request for 3 parameter logistic

$y \sim b1 / (1 + b2 * \exp(-b3 * t))$  where

```
y<-c(5.308, 7.24, 9.638, 12.866, 17.069, 23.192,  
31.443, 38.558, 50.156, 62.948, 75.995, 91.972)
```

```
t<-1:12
```

- But `nls` often fails unless we have “good” starting parameters, while optimization methods mostly get “near” the solution -- HV vignette

# Scaling

Want parameters  $x[i]$  all between 1 and 10

Zero parameters may not be “there”

Also give rise to scaling issues.

Part of the overall issue of *reparametrization*

$$\mathbf{x}_{new} = \mathbf{z}(\mathbf{x}) \quad (\mathbf{z} \text{ vector valued, invertible})$$

Try simple case

$$\mathbf{x}_{new} = \mathbf{Z} \mathbf{x} \qquad \mathbf{x} = \mathbf{Z}^{-1} \mathbf{x}_{new}$$

$\mathbf{Z}$  is a simple non-singular diagonal matrix



# Scaling (cont.)

$$f(\mathbf{x}, \dots) = f(\mathbf{Z}^{-1} \mathbf{x}_{new}, \dots) = f_{new}(\mathbf{x}_{new}, \dots)$$

But we often end up doing the algebra, and it can be error prone, particularly for the derivatives.

$$\partial f_{new}(\mathbf{x}_{new}, \dots) / \partial x_{new_i} = \partial f(\mathbf{x}, \dots) / \partial x_i * \partial x_i / \partial x_{new_i} = \partial f(\mathbf{x}, \dots) / \partial x_i \mathbf{Z}_{ii}^{-1}$$

Hobbs:  $\mathbf{x} \leftarrow c(100, 10, .1)$   $\mathbf{x}_{new} \leftarrow c(1, 1, 1)$

$$\mathbf{Z}^{-1} = \text{diag}(100, 10, .1)$$

$$\mathbf{g}(\mathbf{x}) = c(-100.9131, 783.5327, -82341.5897)$$

$$\mathbf{g}_{new}(\mathbf{x}_{new}) = c(-10091.312, 7835.327, -8234.159)$$

# Why Bad Scaling “Hurts”

hobbs.r: 12 data points to be fitted to  
 $y \sim x_1 / (1 + x_2 \exp(-x_3 t))$  (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

Hence scale check in `optimx()`.

# “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**

# Reparametrization

- D Bates version

$$y \sim c1 / (1 + \exp((c2 - t) / c3))$$

- Still needs care in starting
- Has parameters that can be interpreted
  - Asymptote is c1
  - Time t at midpoint is c2
  - Sharpness of “stepup” inversely related to c3 Self-starting “models” extremely useful
- Sometimes use linear approximations
- Note selfStart – and work to create such tools

# Estimating start

- Guess asymptote; scale:  $yy = y/(1.05*\max(y))$
- Linearize:  $z = \log(yy/(1-yy)) \sim (t - c2) / c3$
- Get  $c2, c3$  from  $\text{lm}(t \sim z)$
- Use `nls(..., algorithm="plinear")` to get  $c1$   
and refine  $c2, c3$

# Summary of results: original model

$$y \sim b1 / (1 + b2 * \exp(-b3 * t))$$

- Unscaled: 196.1862582 49.0916390 0.3135697  
S.E.s 11.307 1.6884 0.0068633  
by Hess 8.0230 1.1983 0.0048600
- Scaled: 1.961863 4.909164 3.135697  
S.E.s 0.11307 0.16884 0.068633  
by Hess 0.080230 0.11983 0.048600

S.E. Estimates:  $\sqrt{\text{RSS} * \text{diag}(\text{solve}(\text{t}(\text{J}) \%*\% \text{J})) / (\text{n} - \text{npar})}$

By Hessian:  $\sqrt{\text{RSS} * \text{diag}(\text{solve}(\text{Hessian}))} / (\text{n} - \text{npar})$

J (jacobian) and Hessian evaluated at parameter estimates.

# Summary of results: Bates model

Formula:  $yy \sim 100 * c1 / (1 + 10 * \exp(0.1 * (c2 - tt) / c3))$

	Estimate	Std. Error	t value	Pr(> t )
c1	1.96186	0.11307	17.35	3.17e-08
c2	5.07416	0.18714	27.11	6.11e-10
c3	0.31891	0.00698	45.69	5.77e-12

Formula:  $yy \sim c1 / (1 + \exp((c2 - tt) / c3))$

c1	196.1863	11.3069	17.35	3.17e-08
c2	12.4173	0.3346	37.11	3.72e-11
c3	3.1891	0.0698	45.69	5.77e-12

RSS for all models  $\approx 2.587$

# Hobbs: lessons

- Because “raw” problem has near singularities in Hessian, NM may succeed when Newton fails, and provide starting values
- Good starting values – `nls()` needs them
- Scaling helps
- Reparametrization helps more, but users may want the original model
- How to get dispersion estimates for model parameters? ??? put in success tests??



# Why things go wrong

- Objective function set up badly
  - Just plain wrong – mistakes in design or coding
  - Poorly scaled
  - Overparametrized
  - No control of inadmissible inputs  
(...)/0; log(0) or log(negative); sqrt(negative)  
exp(big) or  $x^{\text{big}}$   
 $x > 2$  and  $x < 1$  style infeasibilities

*I am currently trying to solve a Maximum Likelihood optimization problem in R. Below you can find the output from R, when I use the "BFGS" method. The problem is that the parameters that I get are very unreasonable, ....*

*(some code)*

*(response)*

*Two possible problems:*

*(a) If you're working with a normal likelihood---and it seems that you are---the exponent should be squared.*

*(b) lag may not be working like you think it should. Consider this silly example ...*

# Why things go wrong - 2

- “Solutions” to math, not to real-world problem
  - Try to build in “admissibility”, but that is difficult!
- Programs, including R packages, have too many control settings for even a small subset of possibilities to have been tested
  - “Tests” are only an infinitesimal subsample of the possible domain
- Some problems are ill-posed (e.g. Hassan18.2)

*There is a contradiction between what the help page says and what constrOptim actually does with the constraints. The issue is what happens on the boundary.*

*The help page says*

*The feasible region is defined by  $ui \%*\% \theta - ci \geq 0$ ,  
but the R code for constrOptim reads  
if (any(ui \%\*\% theta - ci <= 0))  
stop("initial value not feasible")*

# Why things go wrong - 3

- Gradients mis-specified (if at all)
  - </home/john/R-optimtest/2010tutorial/Rhelp-gradient-091130.txt>
  - Sometimes we DO need analytic derivatives ??? fix ref
- “Bad” starting values
  - `genrostart100708.R`
  - Infeasible start
- Bad control settings – check iteration limits
  - Sometimes we're “almost” there, but ...
  - Different controls in different methods
  - `optimx()` tries to unify, but ...

*I am doing a optimization problem using nlminb. It seems to me that the result is kind of sensitive to the starting value.*

---

*I have constructed the function mml2 (below) based on the likelihood function described in the minimal latex I have pasted below for anyone who wants to look at it. This function finds parameter estimates for a basic Rasch (IRT) model. Using the function without the gradient, using either nlminb or optim returns the correct parameter estimates and, in the case of optim, the correct standard errors.*

*By correct, I mean they match another software program as well as the rasch() function in the ltm package.*

*Your function named 'gradient' is not the correct gradient.*

# Annoyances

- Structuring of the problem input/output
  - How functions / expressions must be provided
  - Names / availability of outputs not consistent
  - Attributes vs. Regular returned values
- Getting at the ancillary information “easily”
- Finding information about the methods and approaches e.g., How are SE's computed?
- Everything a little more difficult than we like!
- Options, Options, Options! -- WHY?

# Complaint!

## Tutorial proposal suggested covering

- “Common error messages and how to address them”
- Not easy to do
- Especially when code is *not* in R
- Do any .Rd files include a list of error messages?
  - A mirror would show me one culprit.



# Cobb-Douglas models ??

- Model of production as function of labour (L) and Capital(K)

- $Y \sim \text{beta1} * L^{\text{beta2}} * K^{\text{beta3}}$

- Issue: What should be the loss function

$$Y = \text{beta1} * L^{\text{beta2}} * K^{\text{beta3}} + \text{add\_error}$$

or

$$\log Y = \log(\text{beta1}) + \text{beta2} * \log(L) + \text{beta3} * \log(K) + \text{mult\_error}$$

- “Standard errors”

# Cobb-Douglas Examples

Data from Pedro Arroyo – coef from log model has different sign from unlogged one

Data from <http://www.sts.uzh.ch/data/cobb.html> and André Oliviera very similar

All show parameters with high estimated dispersion

3D graphs may be helpful to see sparsity of data  
– But don't give much help with estimation

# Partial linearity: sum of exponentials

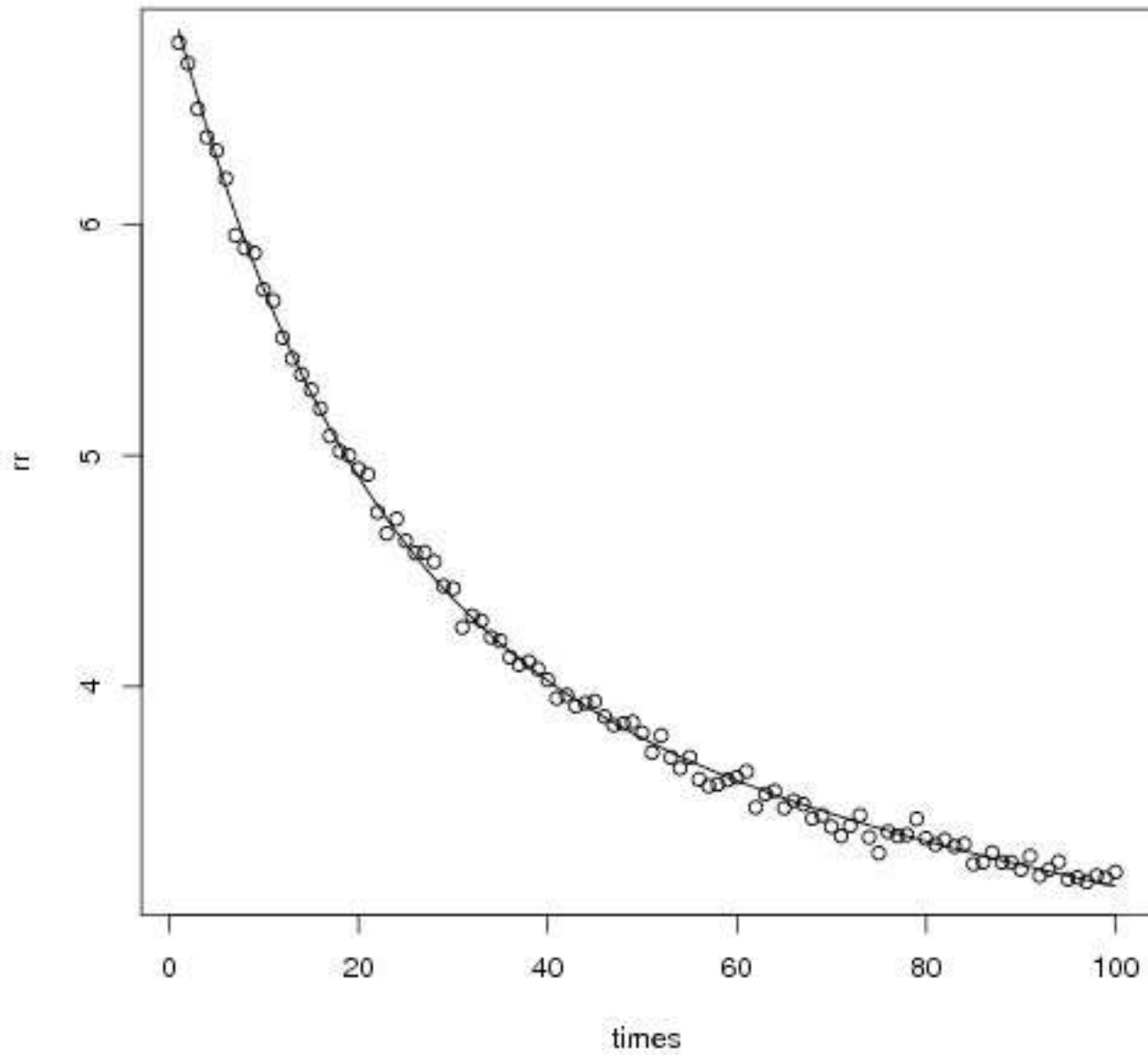
- Sum of exponentials ALWAYS difficult
  - Lanczos (1956) shows why;
    - other refs in Nash and Walker-Smith, 1987,  
<http://macnash.telfer.uottawa.ca/nlpe/>
  - Is this a realistic problem? e.g., chemical kinetics
- NIST Lanczos problems – useful tests or not?
- Plain approach to optimization does very poorly
  - Need to have VERY close starting values
  - Pays to use the partial linearity (gradient?)

# Example – simplified Lanczos

Model:  $y \sim b3 * \exp(-b1 * t) + b4 * \exp(-b2 * t)$

- put parameters = c(0.05, 0.0025, 3, 4)
- Generate t = 1:100 and calculate model values
- Input sd (default 0.04) and set.seed(918273645)
- Generate rnorm(100) and add to model values
- This is similar to competing chemical reactions data.

*optex.R example script*



# Approaches

`nls(rr ~ l1 * exp(-e1*times) + l2 * exp(-e2*times), ...`

This does better than `optim( )` i.e., Nelder Mead.

Even better is

`nls(rr ~ cbind(exp(-e1 * times), exp(-e2 * times)),, data = mydata,  
algorithm='plinear', ....`

BUT zero residual problems cause `nls()` difficulty

`nls.lm` from `minpack.lm` may do somewhat better

And `optim()` does “not too badly” when only nonlinear parameters are varied, i.e., solve for linear model as per 'plinear'.

Problem gets increasingly difficult with number of exponentials

Methods may get ordering of parameters swapped.

# Opinions

- Avoid multiple exponentials problem if you can!
  - Guaranteed multiple minima
  - Inherently ill-conditioned – small changes in inputs --> big changes in outputs (parameters)
- Use the linearity
  - Note that there may be efficiencies in NOT fully solving the problem.  $(Q' * y)[n+1:m]$  gives the residual sum of squares from `qr()`
- Be cautious about problems of this type.
  - How to interpret the linear parameters?

# Shifted Exponential - Lievens

Model:  $y \sim y_0 + \alpha * E^t$

Data:

$y = c(2018.34, 2012.54, 2018.85, 2023.52, 2054.58, 2132.61, 2247.17, 2468.32, 2778.47)$

$t = c(17, 18, 19, 20, 21, 22, 23, 24, 25)$

Note `expontest.Rnw` (Sweave file)

- Naive attempts fail “singular gradient”
- Can see approx  $y_0$  from graph and use linear model to get starting values.
- DEoptim for finding starting values
- qpcR package – big learning curve (>90 pages)



# Distribution fitting

- Tools:
  - MASS::fitdistr, stats4::mle, bbmle::mle2
  - Marie Laure Delignette-Muller: fitdistrplus to extend fitdistr with some extra tools
  - <http://openmx.psyc.virginia.edu/> OpenMX
    - Meant for Structural Equation modeling
  - Can sometimes use glm()
- Biggest issue is learning cost & getting the setup correct

# Examples

- See `fitdistr`, `mle`, and `fitdistrplus`
- Jens Oehlschlägel problem (Poisson glm)
  - Shows Powell's `bobyqa` quite useful here
    - But other examples give trouble e.g. `bvlstest.R`
  - `PoissLikJO.Rnw` vignette

# Large n problems

- Statistical problems tend to be complicated, with difficult code
- Math Programming problems have rather different structure and focus on constraints
- Here will use the eigenvalue problem and some artificial test problems
  - Large as we want
  - Illustrative of the issues
  - Easier to explain the problems and provide tests

# Rayleigh Quotient Minimization

- Given matrix  $A$ , find eigensolution with most positive or most negative eigenvalue by optimizing Rayleigh quotient.
  - $RQ(u) \leftarrow \frac{t(u)^* A u}{t(u)^* u}$
- Do not need  $A$ , though in our tests it will be “around”
  - Should have routine that forms  $v \leftarrow A u$  implicitly

# Rayleigh Quotient minimization

- For symmetric matrix  $A$  of dimension  $n$ , find the vector  $\mathbf{x}$  that minimizes

$$Q = (\mathbf{x}' A \mathbf{x}) / (\mathbf{x}' \mathbf{x})$$

Subject to some constraint on the size of  $\mathbf{x}$ ,

- Typically constrain  $\mathbf{x}' \mathbf{x} = 1$
- Vignette – eigprob.Rnw
  - Need to specialize the optimization to get good results

# Artificial tests

- Almost always sums of squares
- Sometimes hard to find “real” version (broydt.R and genrose.R) – several variants
- But relatively easy to set up and use, including gradients and other derivatives

artificial.Rnw (incomplete)

# Non-smooth / imprecise

- Functions can be non-smooth, i.e., the function or gradient is non-continuous
- Imprecise functions cannot be evaluated exactly e.g., Schumacher time to lap as function of racing car settings
- We tend to use similar – and largely stochastic and heuristic – methods for both classes of problems, but should really differentiate between approaches.

# H Joe problems

Maximum likelihood type problems where the objective function can be thought of as a multi-dimensional integral approximately computed by Monte Carlo techniques

- JN is NOT familiar with the real-world problems
- Harry and I spent > 10 years developing RSMIN which worked rather well, but “nasty” to use

Joe & Nash, *Statistics & Computing*, 13, 277-286, 2005

- Hope: optimizing imprecise function faster than traditional method on accurate function
- NOT in R; need interested users



# Handling constraints

- Bounds – tools available that are relatively easy to use
  - Masks – fixed parameters – Rcgmin & Rvmmin
- Equality Constraints – can be tricky, as really want to solve for parameters if possible
- Inequality constraints
  - linear inequalities – ConstrOptim
  - Projection method -- spg from BB
  - Penalty and Barrier functions – user coded?

# Examples of constraints

- Nonlinear equality constraint – hassan182.R
  - Linear model with constraint on parameters
  - Cannot replicate my own work from ~ 1977 possibly due to typo in data table
  - nls.lm from minpack.lm seems best tool
    - Similar result in 1970s (Marquardt best)
    - Eliminate 1 parameter by solving in constraint
    - Penalty fn method “works” moderately well
    - Parameters ill-conditioned in problem

# Linear inequality constraints

- Could use math programming tools, especially if many constraints.
- Penalty or barrier methods when just a few  
    See Dixon72.R
- Other examples?

# Issues raised by constraints

- How should we interpret measures of dispersion
  - Beta > 0. Does this mean interval ends at 0?
  - How to define & compute dispersion measures
- Setup for constraints is generally non-trivial
- Infeasibility? How do we know?
- Introduced ill-conditioning from constraints
- Disjoint parameter regions
- Inconsistent handling of fn <- Inf or NA

# DANGER!

Advice about to be given.

# Objective function setup

- Keep it as simple as possible
- Scale if possible – poor scaling creates trouble
- Check, check, check
  - Build in checks 'debug<-TRUE' etc.
  - R debug tools?
- Graphs where they make sense

Can we eliminate many “extra” minima? Other “bad” situations?

# Gradients

- Important
  - Sometimes better solutions
  - Speedup, esp. large-n problems
- Using `deriv` or `D` is helpful but not trivial
- Check with `numDeriv()`
- Automatic Differentiation -- work in progress
  - ADMB approach fairly well-developed
  - General tools “under construction” -- `rdax`
- BUT ... lots of work

# Starting values

- Use of linearizing approximations
- Use of “last” values for repetitive estimations
- DEOptim(), optim/SANN
- Random starts
- Use of bounds (and midpoints; random in  $[a,b]$ )
  - Often don't want to be on the bounds
  - May need “local” knowledge of problem
  - Force user to think about problem



# Control settings

- Set iterations  $> 50$  for `nls` via

```
control=list(maxit = 500,trace=TRUE)
```

- A serious issue for different optimization tools is that the controls are different
  - One reason for `optimx()`
- Some methods have more controls than others
  - Often not well-documented; examine code (!?)
- Package defaults may not suit your problem

# Subject specific packages

Polymerase chain reaction models – qpcR

Analysis of dose-response curves – drc

Others ....

- Great if you are doing “same” work
- Not so good if your setup a bit different
- Bad if you don't know the subject
- In any event – lots to learn, so time cost

# Special Methods Packages

Nonlinear mixed effects models – nlme (+ gnls)

Ben Bolkers maxlik package – bbmle (+ mle2)

Bates et al – lme4

Others?

- May offer useful tools and examples
- BUT ... things we want may be missing
- Focus is always on the developer's needs

# Automatic starting values

- SelfStart ideas
- Useful if there is a model already worked out that you need
- Otherwise you have work to do
- Handling exceptions takes most of the work

# Reverse communication

- Attempt to avoid passing “large” structures to subroutines
- MESSY!
- But does simplify setup in some ways, while creating spaghetti in another
- Main routine gets “return” from optimizer with “instruction” -- usually an integer
  - Does work and calls optimizer again
  - Loop until “instruction” is to stop

# Finding Help

- CRAN Task View on Optimization (S Theussl)
- Rhelp – including archives (? how ?)
  - <http://finzi.psych.upenn.edu/search.html>
- Rseek – but does it work?
- Rwiki – Should use it more!
- Nash optimx wiki – for bleeding edge ideas of both users and developers
  - <http://macnash.telfer.uottawa.ca/optimx/>

# Future directions and needs

- **USERS!**
  - Trying things out & organizing tests
  - Helping with documentation
  - Complaining constructively
- **Developers**
  - Integration of methods and tools
  - Better interfaces
- **“Educators”**
  - To help organize our understandings

# Notes to address

- Rcgmin – failure to start in some optimx probs
- Nlmrt – no summary()
- Uli problem as rootfinding