Using and extending the optimr package

John C. Nash

2016-01-11

optimr is a package intended to provide improved and extended function minimization tools for R. Such facilities are commonly referred to as "optimization", but the original `optim()` function and its replacement in this package, which has the same name as the package, namely `optimr()`, only allow for the minimization or maximization of nonlinear functions of multiple parameters subject to at most bounds constraints. Some methods also permit fixed (masked) parameters, which could be considered as equal upper and lower bounds, though that is a VERY bad way to implement masks. In general, we wish to find the vector of parameters `bestpar` that minimize an objective function specified by an R function `fn(par, ... )` where `par` is the general vector of parameters, initially provided as the vector start, and the dot arguments are additional information needed to compute the function. Function minimization methods may require information on the gradient or Hessian of the function, which we will identify as `gr(par, ...)` and `hess(par, ....)`.

How the optimr() function works

`optimr()` is an aggregation of wrappers for a number of individual function minimization ("optimization") tools available for R. The individual wrappers are selected by a sequence of `if()` statements using the argument `method` in the call to `optimr()`.

To add a new optimizer, we need in general terms to carry out the following:

- Ensure the new function is available, that is, the package containing it is installed;
- Add an appropriate `if()` statement to select the new "method";
- Translate the `control` list elements of `optimr()` into the corresponding control arguments (possibly not in a list of that name but in one or more other structures, or even arguments or environment variables) for the new "method";
- If necessary, redefine the R function or functions to compute the value of the function, gradient and possibly Hessian of the objective function so that the output is suited to the method at hand.
- When derivative information is required by a method, we may also need to incorporate the possibility of numerical approximations to the derivative information.
- Add code to check for situations where the new method cannot be applied, and in such cases return a result with appropriate diagnostic information so that the user can either adjust the inputs or else choose a different method. \end{itemize}

Adjusting the objective function

The method `nlm()` provides a good example of a situation where the default `fn()` and `gr()` are inappropriate to the method to be added to `optimr()`. Don't forget the dot arguments!

```
nlmfn <- function(spar, ...){
   f <- efn(spar, ...)
   g <- egr(spar, ...)
   attr(f,"gradient") <- g
   attr(f,"hessian") <- NULL # ?? maybe change later
   f
}
```

In the present `optimr()`, the definition of `nlmfn` is put near the top of `optimr()` and it is always loaded. It is the author's understanding that such functions will always be loaded/interpreted no matter where they are in the code of a function. For ease of finding the code, I have put it near the top, as the structure can be then shared across several similar optimizers. There are other methods that compute the objective function and gradient at the same set of parameters. Though `nlm()` can make use of Hessian information, we have chosen here to omit the computation of the Hessian.