Journal of Statistical Software

MMMMMM YYYY, Volume VV, Issue II. *http://www.jstatsoft.org/*

Comparing Optimization Algorithms in the Fitting of Linear Mixed Models: Evaluating Speed and Accuracy using lme4 in R and lmm in Julia

Douglas Bates
University of Wisconsin-Madison

Colin Longhurst
University of Wisconsin-Madison

Abstract

The **Timings** package allows for the comparison of several optimizers, in both R and Julia, used in the fitting of various linear mixed models. In R the optimizers are called by lmer from the **lme4** package (version 1.1-8). In Julia the optimizers are called by lmm from **MixedModels** package. From the **Timings** package, conclusions regarding an optimizers relative speed, accuracy and general effectiveness of different optimizers paired with different types of models (ranging from simple to complex) can easily be drawn and interpretted.

There are differences in the model formulations in **lme4** and in **MixedModels**. The numerical representation of the model in **lme4** and the method of evaluating the optimizers, described in this paper, is the same for all models. In **MixedModels** there are specialized representations for some model forms, such as models with a single grouping factor for the random effects. Some of the specialized representations allow for evaluation of the gradient of the objects, which can enhance convergence (but, interestingly, sometimes can impede convergence).

Keywords: optimizers, mixed models, linear mixed models, lme4, lmm, R, Julia.

1. Introduction

Mauris ac lectus sagittis, mollis diam porttitor, efficitur purus. Duis ut eros vitae dolor vulputate sagittis. Aenean viverra, dui in interdum consectetur, lectus elit placerat urna, dictum ultricies ex justo eget est. In at leo quam. Suspendisse eros nulla, gravida nec suscipit vel, egestas eget neque. Etiam pharetra, nisl nec viverra ultricies, mauris lectus facilisis velit, vel aliquam ipsum lorem ut sem. Nulla ornare, justo at tempus blandit, nulla urna porta ligula, id dictum mi est ac massa. Quisque at lacus neque. Pellentesque est nulla, dignissim eget lobortis ut, vehicula at erat. Aenean ornare lacus mattis, elementum elit vel, tempor risus. In in purus tempor lacus imperdiet rhoncus nec nec tortor. Duis sagittis nisl ante, id egestas neque tristique fermentum. Fusce aliquet, odio non auctor aliquet, purus orci venenatis purus, sit amet pulvinar nisi est at dolor. Pellentesque lobortis dui eros, et ultricies tellus ultrices at. Ut sit amet interdum justo. Integer placerat vehicula interdum.

2. Methods

To provide consistency we have copied all the data sets used in the timings to the **Timings** package itself. We have done all timings on the same computer. This computer has a relatively recent Intel processor and we used the Intel Math Kernel Library (MKL) with Julia. We attempted to use Revolution R Open (RRO) as the R implementation as it can be configured with **MKL**. However, we ran into version problems with this so we used the standard Ubuntu version of R linked against OpenBLAS, which is also multi-threaded.

Variables were renamed in the pattern:

- **Y** the response
- **A**, **B**, ... categorical covariates
- **G**,**H**, **I**, ... grouping factors for random effects
- **U**, **V**, ... (skipping **Y**) continuous covariates

The timing results are saved in JSON (JavaScript Object Notation) files in the directory accessible as

```
> system.file("JSON",package="Timings")
```

within **R**. The directory name will end with `./Timings/inst/JSON/` in the package source directory, for example the result of cloning the github repository. There is one `.json` file for each data set. Each such file contains results on timings of one or more models.

The **Timings** package for R provides a `retime` function that takes the name of one of these JSON files and, optionally, the name of a file with the updated timings. Similarly there are some source files for Julia retimings.

```
dsname => "Alfalfa"


        `parse` has no method matching parse(::Int64)
    while loading In[1], in expression starting on line 2



         in retime at /home/bates/git/Timings/inst/julia/retime.jl:17
```

The timing was repeated so that compilation time is not included in the results. This repetition is only needed once per session.

A careful examination of these results shows that the main differences in the Julia timings (the R timings are merely reported, not evaluated) are that the `LN_BOBYQA` and `LD_MMA` optimizers are much faster in the second run. This is because much of the code needs to be compiled the first time that a derivative-free optimizer and a derivative-based optimizer are used.

The names of the optimizers used with `lmm` are those from the NLopt package for **Julia**. Names that begin with `LD_` are gradient-based methods. Names that begin with `LN_` are derivative-free methods. There is one other derivative-free method, `LN_PRAXIS`, available in the `NLopt` package but, for some reason, it can hang on very simple problems like this. Frequently we omit it.

The optimizers used with `lmer` include the `Nelder_Mead` optimizer built into the `lme4` package, the `bobyqa` optimizer from the minqa package, the derivative-free optimizers from the nloptr package and several optimizers from the optimx pacakge.

The `optimx:bobyqa` optimizer is just a wrapper around `bobyqa` (bounded optimization by quadratic approximation) from the `minqa` package and should provide results similar to those from the `bobyqa` optimizer. For some reason the number of function evaluations is not reported for the version in `optimx`.

The optimizers from `nloptr` (i.e. those whose names begin with `NLOPT_LN_`) use the same underlying code as do the similarly named optimizers in the `NLopt` package for **Julia**. The number of iterations to convergence should be similar for the same underlying code, although not nessarily exactly the same because the evaluation of the objective in **R** and in **Julia** may produce slightly different answers. Also the convergence criteria in the **Julia** version are more strict than those in the **R** version

Also shown are the value of the criterion (negative twice the log-likelihood, lower is better) achieved, the elapsed time and the number of function and gradient evaluations. The `nopt` value is the number of parameters in the optimization problem. `mtype` is the model type in the **Julia** code. There are special methods for solving the penalized least squares (PLS) problem, and for evaluating the objective and its gradient when there is only one grouping factor for the random effects. The model type is called `PLSOne`.

The `Alfalfa` example is a particularly easy one and all of the optimizerws converge to an objective value close to -10.81023 in less than 0.6 seconds.

3. Results

Nullam eget gravida justo. Proin lectus mauris, dictum vitae ultricies ut, accumsan sed nisi. Pellentesque quis nunc vestibulum, aliquam arcu eu, dapibus risus. Pellentesque lobortis nulla at posuere molestie. Etiam risus nisi, pellentesque a tortor at, porta mollis est. Curabitur nec ante eget felis pellentesque sagittis id in nulla. Vivamus sit amet accumsan lorem. Nam egestas ultrices nisi, volutpat rutrum tortor tincidunt ac. Ut ex nunc, ornare eget nibh quis, mollis aliquam dui. Sed tempor, sapien sed vestibulum malesuada, dolor est ullamcorper leo, at mollis erat nulla eget urna. Aenean risus leo, pellentesque sit amet pulvinar a, ultrices ut lacus. Sed id nunc tempus, commodo augue a, hendrerit nunc. Mauris et purus lorem. Praesent ornare, ipsum id bibendum placerat, arcu enim iaculis tellus, tincidunt rhoncus augue dui sed ex. Proin nunc nisl, lacinia quis tristique eget, tempor eu velit.

3.1. Speed

Etiam placerat commodo dapibus. Vivamus a porta dui. Phasellus accumsan sollicitudin cursus. In nec lorem sit amet purus vulputate suscipit quis ac lectus. Aliquam sollicitudin malesuada blandit. Aliquam sed nisi accumsan, pulvinar diam eget, facilisis urna. Fusce a dignissim ipsum. Integer maximus, felis nec lobortis vulputate, nunc leo blandit sem, id suscipit velit leo in purus. Pellentesque vel arcu sit amet nulla tincidunt ornare. Mauris sodales lectus eget est laoreet lobortis.

3.2. Reliability

Sed iaculis sodales elit quis vehicula. In et tristique neque, sodales aliquet metus. In posuere dictum nisl, quis laoreet augue congue a. Aenean in commodo neque, sit amet hendrerit ex. Aliquam id faucibus ante. Vivamus in fermentum nunc. Nam condimentum eros id orci pretium, quis aliquam magna eleifend.

4. Conclusions

Affiliation:

Douglas Bates
Emeritus Professor
Department of Statistics
University of Wisconsin-Madison
1300 University Ave Madison, WI 53706-1685 U.S.A.
E-mail: dmbates@stat.wisc.edu
Github URL: https://github.com/dmbates

Journal of Statistical Software http://www.jstatsoft.org/
published by the American Statistical Association http://www.amstat.org/
Volume VV, Issue II *Submitted:* yyyy-mm-dd
MMMMMM YYYY *Accepted:* yyyy-mm-dd