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.

```
> include("../julia/retime.jl")
> retime("../JSON/Alfalfa.json","/tmp/Alfalfa.json")
> retime("../JSON/Alfalfa.json","/tmp/Alfalfa.json")
```

INCLUDE SUMMARY TABLE USING res.rda

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

For the `Alfalfa` data there is not much of a burden in refitting the model with all the **Julia** optimizers just to get the table shown above. But other examples can take an hour or more to converge and we don't really need to refit them every time. The `tabulate.jl` file contains a function `optdir` to create a `DataFrame` from the results of all the model fits.

```
> include("../julia/tabulate.jl")
> res = optdir("../JSON");
>
```

Row	opt	dsname	n	np	excess	time	reltime
1	"LD_CCSAQ"	"Alfalfa"	72	1	0.0	0.0017	27.5171
2	"LD_CCSAQ"	"AvgDailyGain"	32	1	0.0	0.0014	13.9236
3	"LD_CCSAQ"	"AvgDailyGain"	32	1	0.0	0.0014	10.8009

4	"LD_CCSAQ"	"BIB"	24	1	0.0	0.0013	11.7005
5	"LD_CCSAQ"	"Bond"	21	1	0.0	0.0009	13.6969
6	"LD_CCSAQ"	"bs10"	1104	20	0.0	1.0958	12.608
7	"LD_CCSAQ"	"bs10"	1104	8	39.9948	0.0375	23.2699
8	"LD_CCSAQ"	"cake"	270	1	0.0	0.0033	18.8753
9	"LD_CCSAQ"	"Cultivation"	24	1	0.0	0.0009	15.6477
10	"LD_CCSAQ"	"Demand"	77	2	3.21928	0.0055	22.0827
11	"LD_CCSAQ"	"dialectNL"	225866	6	0.0	6.9896	3.942
12	"LD_CCSAQ"	"Dyestuff2"	30	1	0.0	0.0005	0.682
13	"LD_CCSAQ"	"Dyestuff"	30	1	0.0	0.0008	1.0184
14	"LD_CCSAQ"	"ergoStool"	36	1	0.0	0.0012	1.1018
15	"LD_CCSAQ"	"Exam"	4059	1	0.0	0.0105	1.1851
16	"LD_CCSAQ"	"Exam"	4059	1	0.0	0.0105	1.318
17	"LD_CCSAQ"	"Gasoline"	32	1	0.0	0.0011	0.8193
18	"LD_CCSAQ"	"gb12"	225866	6	0.0	3.5221	17.4791
19	"LD_CCSAQ"	"gb12"	512	8	103.176	0.0218	13.1358
20	"LD_CCSAQ"	"HR"	120	3	0.0	0.0089	8.6481
21	"LD_CCSAQ"	"Hsb82"	7185	1	192.73	0.0102	3.7438
22	"LD_CCSAQ"	"IncBlk"	24	1	0.55726	0.001	5.0422
23	"LD_CCSAQ"	"kb07"	1790	72	8.20739	17.4698	9.2732
24	"LD_CCSAQ"	"Mississippi"	37	1	0.93471	0.0006	15.9582
25	"LD_CCSAQ"	"mm0"	69588	6	0.0	4.8286	1.1034
26	"LD_CCSAQ"	"Oxboys"	234	3	136.788	0.0169	5.3039
27	"LD_CCSAQ"	"PBIB"	60	1	0.0	0.0014	8.6177
28	"LD_CCSAQ"	"Penicillin"	144	2	0.0	0.0131	15.3763
29	"LD_CCSAQ"	"Semiconductor"	48	1	0.0	0.0012	1.133
30	"LD_CCSAQ"	"SIMS"	3691	3	3.60856	0.134	2.3154

The `time` column is the time in seconds to converge. The `reltime` column is the time relative to the `LN_BOBYQA` optimizer in the **MixedModels** package for Julia.

For Julia the time column is the time in seconds to converge. The reltime column is the time relative to the LN$_BOBYQAoptimizerintheMixedModelspackagefor$Julia.

Row	opt	dsname	n	np	excess	time	reltime
1	"NLOPT_LN_BOBYQA"	"Alfalfa"	72	1	0.0	0.042	27.5171
2	"NLOPT_LN_BOBYQA"	"Animal"	20	2	0.0	0.023	13.9236
3	"NLOPT_LN_BOBYQA"	"Assay"	60	2	1.0e-5	0.032	10.8009
4	"NLOPT_LN_BOBYQA"	"AvgDailyGain"	32	1	0.0	0.02	11.7005
5	"NLOPT_LN_BOBYQA"	"AvgDailyGain"	32	1	0.0	0.02	13.6969
6	"NLOPT_LN_BOBYQA"	"BIB"	24	1	0.0	0.02	12.608
7	"NLOPT_LN_BOBYQA"	"Bond"	21	1	0.0	0.02	23.2699
8	"NLOPT_LN_BOBYQA"	"bs10"	1104	20	1.0e-5	4.661	18.8753
9	"NLOPT_LN_BOBYQA"	"bs10"	1104	8	0.0	1.057	15.6477
10	"NLOPT_LN_BOBYQA"	"cake"	270	1	0.0	0.053	22.0827
11	"NLOPT_LN_BOBYQA"	"Chem97"	31022	2	0.0	0.632	3.942

```
.
.
.
| 38|"NLOPT\_LN\_BOBYQA" | "PBIB"          | 60     | 1  | 0.0     | 0.018  |13.1358|
| 39|"NLOPT\_LN\_BOBYQA" | "Penicillin"    | 144    | 2  | 0.0     | 0.023  |8.6481|
| 40|"NLOPT\_LN\_BOBYQA" | "Poems"         | 275996 | 3  | 0.0     | 21.309 |3.7438|
| 41|"NLOPT\_LN\_BOBYQA" | "ScotsSec"      | 3435   | 2  | 0.0     | 0.076  |5.0422|
| 42|"NLOPT\_LN\_BOBYQA" | "Semi2"         | 72     | 3  | 0.0     | 0.03   |9.2732|
| 43|"NLOPT\_LN\_BOBYQA" | "Semiconductor"| 48      | 1  | 0.0     | 0.019  |15.9582|
| 44|"NLOPT\_LN\_BOBYQA" | "SIMS"          | 3691   | 3  | 0.0     | 0.15   |1.1034|
| 45|"NLOPT\_LN\_BOBYQA" | "sleepstudy"    | 180    | 3  | 0.0     | 0.037  |5.3039|
| 46|"NLOPT\_LN\_BOBYQA" | "sleepstudy"    | 180    | 2  | 0.0     | 0.024  |15.3763|
| 47|"NLOPT\_LN\_BOBYQA" | "TeachingII"    | 96     | 1  | 0.0     | 0.021  |1.133|
| 48|"NLOPT\_LN\_BOBYQA" | "Weights"       | 399    | 3  | 1.0e-5  | 0.039  |5.3039|
| 49|"NLOPT\_LN\_BOBYQA" | "WWheat"        | 60     | 3  | 0.0     | 0.025  |2.3154|
```

3.1. Proportion Converged

The most important question regarding the optimizers is whether or not they have converged to the global optimum. We cannot test this directly. Instead we use a ”crowd-sourced” criterion based on the minimum objective achieved by any of the algorithms. The difference between the objective achieved by a particular algorithm and this minimum is called the excess. In the summaries `excess` is rounded to 5 digits after the decimal so the minimum non-zero `excess` is 10^{-5}.

```
Out[4]: 49x3 DataFrame
        | Row | opt          | dsname           | excess |
        |-----|--------------|------------------|--------|
        | 1   | "LN_BOBYQA" | "Alfalfa"        | 0.0    |
        | 2   | "LN_BOBYQA" | "Animal"         | 0.0    |
        | 3   | "LN_BOBYQA" | "Assay"          | 0.0    |
        | 4   | "LN_BOBYQA" | "AvgDailyGain"   | 0.0    |
        | 5   | "LN_BOBYQA" | "AvgDailyGain"   | 0.0    |
        | 6   | "LN_BOBYQA" | "BIB"            | 0.0    |
        | 7   | "LN_BOBYQA" | "Bond"           | 0.0    |
        | 8   | "LN_BOBYQA" | "bs10"           | 1.0e-5 |
        | 9   | "LN_BOBYQA" | "bs10"           | 0.0    |
        | 10  | "LN_BOBYQA" | "cake"           | 0.0    |
        | 11  | "LN_BOBYQA" | "Chem97"         | 0.0    |
```

If we wish to declare ``converged'' or ``not converged'' according to the excess objective value we must establish a threshold. An absolute threshold seems reasonable because the objective,negative twice the log-likelihood,is on a scale where differences in this objective are compared to a χ^2 random variable. Thus an excess of 10^{-9} or even 10^{-5} is negligible.

For each optimizer we can examine which of the data set/model combinations resulted in an excess greater than a threshold.

```
| Row | opt                              | attempted | failed |
|-----|----------------------------------|-----------|--------|
| 1   | "LD\_CCSAQ"                      | 35         | 11      |
| 2   | "LD\_LBFGS"                      | 35         | 11      |
| 3   | "LD\_MMA"                        | 36         | 5       |
| 4   | "LD\_SLSQP"                      | 35         | 4       |
| 5   | "LD\_TNEWTON"                    | 34         | 8       |
| 6   | "LD\_TNEWTON\_PRECOND"           | 34         | 8       |
| 7   | "LD\_TNEWTON\_PRECOND\_RESTART"  | 33         | 12      |
| 8   | "LD\_TNEWTON\_RESTART"           | 34         | 11      |
| 9   | "LD\_VAR1"                       | 35         | 10      |
| 10  | "LD\_VAR2"                       | 35         | 10      |
| 11  | "LN\_BOBYQA"                     | 49         | 0       |
âŃ■
| 15  | "LN\_SBPLX"                      | 49         | 2       |
| 16  | "NLOPT\_LN\_BOBYQA"              | 49         | 0       |
| 17  | "NLOPT\_LN\_COBYLA"              | 48         | 2       |
| 18  | "NLOPT\_LN\_NELDERMEAD"          | 47         | 6       |
| 19  | "NLOPT\_LN\_PRAXIS"              | 18         | 4       |
| 20  | "NLOPT\_LN\_SBPLX"               | 48         | 2       |
| 21  | "Nelder\_Mead"                   | 49         | 8       |
| 22  | "bobyqa"                         | 49         | 2       |
| 23  | "optimx:L-BFGS-B"                | 49         | 0       |
| 24  | "optimx:bobyqa"                  | 49         | 2       |
| 25  | "optimx:nlminb"                  | 49         | 0       |
| 26  | "optimx:spg"                     | 49         | 4       |
\end{verbatim}
```

At this threshold the most reliable algorithm in Julia is \texttt{LN_BOBYQA}. In \proglang{R} the most reliable algorithms are \texttt{NLOPT_LN_BOBYQA}, \texttt{optimx:L-BFGS-B} and \texttt{optimx:nlminb}. It is interesting that nlminb is reliable as I felt that it wasn't converging well when it was the default optimizer in \pkg{lmer}.

Interestingly, the derviative-based algorithms in \texttt{NLopt} were not as reliable as the derivative-free algorithms. The most likely explanation is that I don't have the gradient coded properly.

The Nelder-Mead simplex algorithm did not perform well, failing on 8 out of 48 cases. For many of these the value at which convergence was declared was far from the optimum.

```
\begin{verbatim}
| Row | opt            | dsname         | excess   | time     | reltime | np | n|
```

```
|-----|---------------|---------------|---------|---------|---------|----|--|
| 1   | "Nelder\_Mead" | "bs10"        | 71.3859 | 145.368 | 588.684 | 20 |1104|
| 2   | "Nelder\_Mead" | "d3"          | 317.59  | 454.519 | 4.2502  | 9  |130418|
| 3   | "Nelder\_Mead" | "dialectNL"   | 181.632 | 54.541  | 31.1594 | 6  |225866|
| 4   | "Nelder\_Mead" | "gb12"        | 78.7119 | 38.206  | 189.605 | 20 |512|
| 5   | "Nelder\_Mead" | "kb07"        | 398.732 | 2825.46 | 667.015 | 72 |1790|
| 6   | "Nelder\_Mead" | "kb07"        | 403.478 | 269.436 | 383.087 | 16 |1790|
| 7   | "Nelder\_Mead" | "Mississippi" | 0.04272 | 0.018   | 20.3989 | 1  |37|
| 8   | "Nelder\_Mead" | "mm0"         | 181.632 | 76.87   | 70.0716 | 6  |69588|
\end{verbatim}
```

The \texttt{Nelder_Mead} algorithm, either in the native form in \pkg{lmer} or in the \texttt{NLopt} implementation performed poorly on those cases with many parameters to optimize. It was both unreliable and slow, taking over 45 minutes to reach a spurious optimum on the ``maximal'' model (in the sense of Barr et al., 2012) for the \texttt{kb07} data from Kronmueller and Barr (2007). This is not terribly surprising given that the model is horribly overparameterized, but still it shows that this algorithm is not a good choice in these cases.

We note in passing that all the models involving fitting 20 or more parameters are ``maximal'' models in the sense of Barr et al., 2012. Such models can present difficult optimization problems because they are severely overparameterized and inevitably converge on the boundary of the allowable parameter space. Whether or not it is sensible to compare results on such extreme cases is not clear.

The \texttt{SBPLX} (subplex) algorithm, which is an enhancement of \texttt{Nelder_Mead}, does better in these cases but is still rather slow.

```
 \begin{verbatim}
        | Row | opt               | dsname | excess  | time    | reltime | np | n    |
        |-----|-------------------|--------|---------|---------|---------|----|------|
        | 1   | "NLOPT\_LN\_SBPLX" | "gb12" | 0.82219 | 3.813   | 18.9228 | 20 | 512  |
        | 2   | "NLOPT\_LN\_SBPLX" | "kb07" | 4.96546 | 564.688 | 133.308 | 72 | 1790 |
\end{verbatim}
```

By comparison, the \texttt{LN_BOBYQA} algorithm converges quite rapidly on the \texttt{kb07} models.

```
\begin{Verbatim}
        | Row | dsname | excess  | objective | time   | np |
```

```
|-----|--------|---------|-----------|--------|----|
| 1   | "kb07" | 0.01695 | 28586.3   | 4.236  | 72 |
| 2   | "kb07" | 0.0     | 28670.9   | 0.7033 | 16 |
```

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

5. References

Note all optimization packages used,lmm,lme4,jsonlite,

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