Fitting linear, generalized linear and mixed models to large data sets

Douglas M. Bates

University of Wisconsin – Madison
and
R Development Core Team
<bates@R-project.org>

useR!2007, Ames, Iowa, U.S.A., August 9, 2007

Outline

Outline

Outline

Outline

Outline

Outline

Are my data a "large" data set?

- Computing hardware and software evolves rapidly (my DVD player runs *R*). We must apply an evolving standard for what constitutes a "large" data set.
- Your data are not large if using the naive (i.e. the "blunt instrument" or `fortune(122)`) approach takes less time than finding and reading the documentation that describes how to do what you want to do efficiently.
- If you find that the naive approach runs out of memory or takes a long time you can
  - Send an email message to R-help with a subject like "R is SOOOOOO SLOOOOOOW" and vaguely describe what you are trying to do and why this software is so bad. (Do keep an asbestos suit handy for the `fortune(9)` response.)
  - Try the `fortune(138)` approach.

What is the effect of large data?

- In short, you need to think about the computation and how it is to be done.
- Early texts on statistical computing (Chambers, 1975 or Kennedy and Gentle, 1980) emphasized algorithms for numerical linear algebra, simulation, optimization, etc. paying careful attention to the time and space requirements.
- In modern environments we don't bother with "counting flops" because we realize that setting up the numerical computation often dominates the time spent actually doing the computation (`fortune(98)`).
- Large data sets can change that balance. All that hard earned knowledge of how the cost of various operations changes with the size of the matrices involved

Outline

Models and model matrices

- One of the underappreciated aspects of the *S* language is the ability to combine a formula and a data set to produce model frame and a model matrix. It is not as trivial as it may seem.
- Most operations with an $n \times p$ model matrix $\boldsymbol{X}$ and a response vector $\boldsymbol{y}$ are some form of least squares (e.g. iteratively reweighted least squares).
- Everything you learned in a linear algebra class doesn't apply to computational linear algebra. ("No, Virginia, least squares computations are not really done as $(\boldsymbol{X}'\boldsymbol{X})^{-1}\boldsymbol{X}'\boldsymbol{y}$.")
- Numerical analysts recognize two direct decomposition methods (QR and Cholesky) for linear least squares. The time complexity is np^2 for both. Space complexity is p^2 for Cholesky and np for QR. (The "sweep" operator used by SAS is not even considered a contender.)

Rank deficiencies

- The sweep-based method used by SAS is used because it can detect and remove rank deficiencies (in a way different from how numerical analysts would do it).
- The problem of rank deficiencies occurs less frequently in the *S* language implementation but still frequently enough that it must be taken into consideration. This is why the QR decomposition in *R* is still based on Linpack, not Lapack.
- Standard numerical linear algebra software doesn't quite do things the way that statisticians want to (and numerical analysts don't seem to want to make the necessary changes).

Outline

Cholesky versus QR

- The Cholesky method can be row-oriented (i.e. it can operate on horizontal sections of $\boldsymbol{X}$ and $\boldsymbol{y}$).
- The QR method is column-oriented.
- In theory *R* provides the option of using Cholesky or QR (there is a `method` argument to `lm.fit`). In practice it only allows QR. (See previous discussion of rank deficiency.) "In theory, theory and practice are the same. In practice, they're not."
- QR has somewhat better numerical properties than Cholesky. If this matters your $\boldsymbol{X}$ is close to rank deficient.

What can you do with existing dense matrix software?

- You could work on horizontal sections of $\boldsymbol{X}$ and $\boldsymbol{y}$ (biglm package) with a Cholesky decomposition.
- If you use iterative reweighting then you must store all of $\boldsymbol{X}$ somewhere or regenerate it from the model frame and the formula at each iteration.

Is n large, p moderate, or are n and p both large?

- If p is getting large as n gets large, what causes p to get large?
- Do you have "nuisance parameters" associated with experimental (observational) units? If so, they should probably be modeled as random effects.
- If the linear predictor involves nuisance parameters, that part of the model matrix will end up being sparse.
- If you model the effects of units as fixed effects you need to watch for rank deficiencies. When modeled as random effects the "shrinkage" behavior of the BLUPs provides the regularization.
- For this type of data and model the combination of mixed-effects models and sparse matrix techniques is highly effective.

Outline

An example

```
'data.frame': 1721024 obs. of  9 variables:
 $ instr   : Factor w/ 7964 levels "10000","10001",..: 1 1 1 1 1 1 1 1
 $ dept    : Factor w/ 106 levels "AERO","AFAM",..: 43 43 43 43 43 43 4
 $ id      : Factor w/ 54711 levels "900000001","900000002",..: 12152 1
 $ nclass  : num  40 29 33 13 47 49 37 14 21 20 ...
 $ vgpa    : num  NA NA NA NA NA NA NA NA NA NA ...
 $ rawai   : num   2.88 -1.15 -0.08 -1.94  3.00 ...
 $ gr.pt   : num  4 1.7 2 0 3.7 1.7 2 4 2 2.7 ...
 $ section : Factor w/ 70366 levels "19959 AERO011A001",..: 18417 18417
 $ semester: num  19989 19989 19989 19989 19972 ...
```

An initial model fit

```
> system.time(m1 <- lmer(gr.pt ~ (1|id) + (1|instr) + (1|dept),
+                        anon.grades.df, verbose = 1))

  0:     3661278.9: 0.294219 0.111907 0.0127038
  1:     3449897.0: 0.759218 0.360591 0.862370
  2:     3447850.7: 0.933685 0.418499 0.797016
  3:     3447432.7: 0.872045 0.462103 0.792406
  4:     3447415.1: 0.876286 0.448215 0.791340
  5:     3447413.5: 0.873344 0.445115 0.790446
  6:     3447413.1: 0.874789 0.443935 0.789482
  7:     3447412.9: 0.873407 0.443476 0.787969
  8:     3447412.6: 0.874738 0.443452 0.786346
  9:     3447399.6: 0.873381 0.435214 0.665693
 10:     3447382.9: 0.873987 0.446658 0.545296
 11:     3447380.0: 0.873637 0.440919 0.486841
 12:     3447379.6: 0.874317 0.444942 0.472003
 13:     3447379.5: 0.874401 0.443633 0.475912
 14:     3447379.5: 0.874153 0.443656 0.475909
 15:     3447379.5: 0.874185 0.443656 0.475662
 16:     3447379.5: 0.874191 0.443667 0.475346
    user   system  elapsed
3237.977   28.393 3266.937
```

An initial model fit

```
Linear mixed-effects model fit by REML
Formula: gr.pt ~ (1 | id) + (1 | instr) + (1 | dept)
   Data: anon.grades.df
     AIC     BIC   logLik MLdeviance REMLdeviance
 3447387 3447437 -1723690    3447374      3447379
Random effects:
 Groups   Name Variance Std.Dev.
 id            0.3085   0.555
 instr         0.0795   0.282
 dept          0.0909   0.301
 Residual      0.4037   0.635
Number of obs: 1685394, groups: id, 54711; instr, 7915; dept, 102

Fixed effects:
            Estimate Std. Error t value
(Intercept)   3.1996     0.0314     102
```

How can such a computation be done?

- If we write a linear mixed model as

$$\boldsymbol{y} = \boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{Z}\boldsymbol{b} + \boldsymbol{\epsilon}, \quad \boldsymbol{b} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{\Sigma}),\ \boldsymbol{\epsilon} \sim \mathcal{N}(\boldsymbol{0}, \sigma^2\boldsymbol{I}),\ \boldsymbol{b} \perp \boldsymbol{\epsilon} \tag{1}$$

  where $\boldsymbol{\Sigma}$ depends on a parameter $\boldsymbol{\theta}$ (the three relative variance components, in the example), then $\boldsymbol{Z}$ and $\boldsymbol{\Sigma}$ are very sparse.
- Let $\boldsymbol{\Sigma}(\boldsymbol{\theta}) = \boldsymbol{T}(\boldsymbol{\theta})\boldsymbol{S}(\boldsymbol{\theta})\boldsymbol{S}(\boldsymbol{\theta})\boldsymbol{T}'(\boldsymbol{\theta})$ be the "LDL" form of the Cholesky factorization of the relative variance matrix. That is, $\boldsymbol{T}$ is unit lower triangular and bmS is diagonal and on the scale of the relative standard deviation, not the variance. In our example $\boldsymbol{T} = \boldsymbol{I}$.
- Let $\boldsymbol{P}$ represent the fill-reducing permutation for $\boldsymbol{Z}'\boldsymbol{Z}$.
- For a given value of $\boldsymbol{\theta}$ set $\boldsymbol{V} = \boldsymbol{Z}\boldsymbol{T}(\boldsymbol{\theta})\boldsymbol{S}(\boldsymbol{\theta})\boldsymbol{P}'$.

The mixed model equations

- We work with the Choleksy decomposition of an extended system matrix

$$\begin{bmatrix} \boldsymbol{P}(\boldsymbol{V}'\boldsymbol{V}+\boldsymbol{I})\boldsymbol{P}' & \boldsymbol{P}\boldsymbol{V}'\boldsymbol{X} & \boldsymbol{P}\boldsymbol{V}'\boldsymbol{y} \\ \boldsymbol{X}'\boldsymbol{V}\boldsymbol{P}' & \boldsymbol{X}'\boldsymbol{X} & \boldsymbol{X}'\boldsymbol{y} \\ \boldsymbol{y}'\boldsymbol{V}\boldsymbol{P}' & \boldsymbol{y}'\boldsymbol{X} & \boldsymbol{y}'\boldsymbol{y} \end{bmatrix} = \boldsymbol{R}'\boldsymbol{R} \tag{2}$$

where

$$\boldsymbol{R} = \begin{bmatrix} \boldsymbol{L}' & \boldsymbol{R}_{VX} & \boldsymbol{r}_{Vy} \\ \boldsymbol{0} & \boldsymbol{R}_{\boldsymbol{X}} & \boldsymbol{r}_{Xy} \\ \boldsymbol{0} & \boldsymbol{0} & r \end{bmatrix}. \tag{3}$$

- The matrices $\boldsymbol{L}'$ and $\boldsymbol{R}_{\boldsymbol{X}}$ are upper triangular of dimension $q \times q$ and $p \times p$ respectively. The corresponding vectors, $\boldsymbol{r}_{Zy}$ and $\boldsymbol{r}_{Xy}$, are of dimension q and p, and r is a scalar.

Profiled deviance functions

- The *profiled deviance* function

$$
\begin{aligned}
-2\ell(\widehat{\boldsymbol{\beta}}(\boldsymbol{\theta}),\boldsymbol{\theta},\widehat{\sigma^2}(\boldsymbol{\theta})|\boldsymbol{y}) & \\
&= n\log\left(\frac{2\pi r^2(\boldsymbol{\theta})}{n}\right)+n+2\log|\boldsymbol{L}(\boldsymbol{\theta})| \\
&= n\left[1+\log\left(2\pi/n\right)\right]+n\log r^2(\boldsymbol{\theta})+2\log|\boldsymbol{L}(\boldsymbol{\theta})|. \quad (4)
\end{aligned}
$$

- The profiled REML deviance is

$$
\begin{aligned}
-2\ell_R(\boldsymbol{\theta},\widehat{\sigma}^2(\boldsymbol{\theta})|\boldsymbol{y}) & \\
&= (n-p)\left[1+\log\left(2\pi/(n-p)\right)\right]+(n-p)\log r^2(\boldsymbol{\theta}) \\
&\qquad +2\log|\boldsymbol{L}(\boldsymbol{\theta})|+2\log|\boldsymbol{R}_X(\boldsymbol{\theta})|. \quad (5)
\end{aligned}
$$

An initial model fit

```
> object.size(m1)/(2^20)
[1] 747.95
> slotsz(m1)/(2^20)
         L          Vt          Zt       frame       flist           X
5.2332e+02 6.4294e+01 6.4294e+01 4.4384e+01 2.3057e+01 1.2859e+01 1.285
      RVXy        ZtXy        uvec       ranef        call       terms
9.5734e-01 9.5734e-01 4.7861e-01 4.7861e-01 2.6398e-03 2.5024e-03 9.765
      dims    deviance      cnames       fixef         RXy       XytXy
8.0872e-04 6.2561e-04 3.6621e-04 2.5177e-04 2.2125e-04 2.2125e-04 5.340
    offset     weights
3.8147e-05 3.8147e-05
```

Outline

Conclusions

- There are several approaches to dealing with large data sets in *R*.
- The `fortune(122)` approach is generally not terribly successful.
- *R* is not a filter (in the sense that SAS and SPSS are).
- At present the `model.matrix` function always outputs a full dense matrix. This is usually the point at which those working with large data sets run into trouble.
- It is possible, but not easy, to work with horizontal sections of the data and model matrix in the current formulation.
- The `lmer` function fits various types of mixed models using a sparse matrices but does not go through `model.matrix` for the sparse parts.
- The case of mixed models is somewhat special because grouping factors can be represented as indicators and you don't need to watch for rank deficiencies.
- If both n and p are large (and if p grows with n) then