Sparse Matrices in package Matrix and applications

Martin Maechler and Douglas Bates

Seminar für Statistik
ETH Zurich Switzerland

Department of Statistics
University of Madison, Wisconsin U.S.A.
(maechler|bates)@R-project.org (R-Core)

useR! 2009, Rennes
July 10, 2009
Outline

1. Introduction to Matrix and Sparse Matrices
   - Sparse Matrices in package Matrix
   - Matrix: Goals
   - 3D space of Matrix classes

2. Applications in Spatial Statistics
   - Regression with Spatially Dependent Errors: SAR(1)

3. Application - Mixed Modelling (RE)ML in R

4. Who's the best liked prof at ETH?
Introduction

- **Matrix**: the movie
- **Matrix**: the R package
- Package Matrix: a recommended R package since R 2.9.0
- Infrastructure for other packages for several years, notably lme4
- CRAN nowadays lists direct "reverse dependencies"

---

Matrix: Generalized- and Non-Linear Mixed Effect Modelling, using S4 re-implemented from scratch the 4th time.

Martin Maechler, Doug Bates (R Core) 
Sparse Matrices in Matrix pkg 
useR!, Rennes 2009
Introduction

- **Matrix: the movie**
- **Matrix: the R package**:

- Package Matrix: a **recommended** R package since R 2.9.0
- Infrastructure for other packages for several years, notably lme4\(^1\)
- CRAN nowadays lists direct “reverse dependencies”:

\(^{1}\) lme4 := (Generalized–) (Non–) Linear Mixed Effect Modelling, (using S4 | re-implemented from scratch the 4\(^{th}\) time)
Introduction

- Matrix: the movie
- **Matrix: the R package:**
  - Package Matrix: a *recommended* R package since R 2.9.0
  - Infrastructure for other packages for several years, notably lme4\(^1\)
  - CRAN nowadays lists direct “reverse dependencies”:

\(^1\)lme4 := (Generalized–) (Non–) Linear Mixed Effect Modelling, (using S4 | re-implemented from scratch the 4th time)
Introduction

- **Matrix: the movie**
- **Matrix: the R package:**
  - Package **Matrix**: a **recommended** R package since R 2.9.0
  - Infrastructure for other packages for several years, notably **lme4**
  - CRAN nowadays lists direct “reverse dependencies”:

\[\text{lme4} := \text{(Generalized–) (Non–) Linear Mixed Effect Modelling,}
\text{(using S4 | re-implemented from scratch the } 4^{th} \text{ time)}\]
(reverse) Dependencies on Matrix

On June 26, 2008 (> one year ago), Matrix was not yet recommended, and had the following CRAN dependency graph:

31 nodes with 34 edges

i.e., 14 + 1 directly dependent packages.
Dependencies on Matrix – 2009-07

Today, quite a few more packages depend on Matrix explicitly:

CRAN → Packages → Matrix displays the following

http://cran.r-project.org/web/packages/Matrix/

Matrix: Sparse and Dense Matrix Classes and Methods

Classes and methods for dense and sparse matrices and operations on them using Lapack and SuiteSparse.

Version: 0.999375-29
Priority: recommended
Depends: R (≥ 2.9.0), stats, methods, utils, lattice
Imports: graphics, lattice, grid, stats
Enhances: graph, SparseM
Author: Douglas Bates and Martin Maechler

Reverse dependencies:

Reverse depends: FAiR, FTICRMS, GOSim, MCMCglmm, Metabonomic, arm, arules, glmnet, klin, languageR, lme4, mlmRev, pedigreemm, qgen, ramps, spdep, surveyNG, svcm, systemfit, tpr, tsDyn
Matrix: Sparse and Dense Matrix Classes and Methods

Classes and methods for dense and sparse matrices and operations on them using Lapack and SuiteSparse.

Version: 0.999375-29
Priority: recommended
Depends: R (≥ 2.9.0), stats, methods, utils, lattice
Imports: graphics, lattice, grid, stats
Enhances: graph, SparseM
Author: Douglas Bates and Martin Maechler

Reverse dependencies:

Reverse depends: FAiR, FTICRMS, GOSim, MCMCglmm, Metabonomic, arm, arules, glmnet, klin, languageR, lme4, mlmRev, pedigreeem, qgen, ramps, spdep, surveyNG, svcm, systemfit, tpr, tsDyn
Reverse imports: arules, cba
Reverse suggests: R.matlab, RSiena, Rcsdp, blockmodeling, classGraph, e1071, gmodels, igraph, rattle, spam, survey
Reverse enhances: Rcplex, Rcsdp
After one year, we have 22 (up from 15) packages depending on Matrix explicitly, plus another 12 “suggest” or “enhance” it.

Notably glmnet, Trevor Hastie’s favorite in yesterday’s keynote.

Most important one: lme4 and its dependencies
After one year, we have 22 (up from 15) packages depending on Matrix explicitly, plus another 12 “suggest” or “enhance” it.

Notably glmnet, Trevor Hastie’s favorite in yesterday’s keynote.

Most important one: lme4 and its dependencies.
The R Package Matrix contains dozens of matrix classes and hundreds of method definitions.

Has sub-hierarchies of \texttt{denseMatrix} and \texttt{sparseMatrix}.

Very basic intro in \textit{some} of sparse matrices:
simple example — Triplet form

The most obvious way to store a sparse matrix is the so called “Triplet” form; (virtual class TsparseMatrix in Matrix):

```
> A <- spMatrix(10, 20, i = c(1,3:8),
+       j = c(2,9,6:10),
+       x = 7 * (1:7))
> A # a "dgTMatrix"
```

10 x 20 sparse Matrix of class "dgTMatrix"

```
[1,] . 7 . . . . . . . . . . . . . . . . . .
[2,] . . . . . . . . . . . . . . . . . . . .
[3,] . . . . . . . . 14 . . . . . . . . . .
[4,] . . . . . 21 . . . . . . . . . . . . .
[5,] . . . . . 28 . . . . . . . . . . . . .
[6,] . . . . . . 35 . . . . . . . . . . . .
[7,] . . . . . . 42 . . . . . . . . . . . .
[8,] . . . . . . . 49 . . . . . . . . . . .
[9,] . . . . . . . . . . . . . . . . . . . .
[10,] . . . . . . . . . . . . . . . . . . . .
```

Less didactical, slightly more recommended: A1 <- sparseMatrix(.....)
simple example – 2 –

```r
> str(A) # note that *internally* 0-based indices (i,j) are used

Formal class 'dgTMatrix' [package "Matrix"] with 6 slots
  ..@ i    : int [1:7] 0 2 3 4 5 6 7
  ..@ j    : int [1:7] 1 8 5 6 7 8 9
  ..@ Dim  : int [1:2] 10 20
  ..@ Dimnames:List of 2
    .. ..$: NULL
    .. ..$ : NULL
  ..@ x    : num [1:7] 7 14 21 28 35 42 49
  ..@ factors : list()

> A[2:7, 12:20] <- rep(c(0,0,0,(3:1)*30,0), length = 6*9)
> A >= 20  # what result do you expect ?
```
simple example – 3 –

> A >= 20 # -> logical sparse; nice show() method

10 x 20 sparse Matrix of class "lgTMatrix"

```
[1,] . . . . . . . . . . . . . . . . . . . .
[2,] . . . . . . . . . . . . . | | | . . . .
[3,] . . . . . . . . . . . . . | | | . . . .
[4,] . . . . | . . . . . . . . . | | | . .
[5,] . . . . | . . . . | . . . . | | | .
[6,] . . . . | . . . | | . . . . | | | .
[7,] . . . . | . . | | | . . . . | | | .
[8,] . . . . | . . . . . . . . . . | | | .
[9,] . . . . . . . . . . . . . . . . . . . .
[10,] . . . . . . . . . . . . . . . . . . . .
```
sparse *compressed* form

Triplet representation: easy for us humans, but can be both made smaller *and* more efficient for (column-access heavy) operations:

The “column compressed” sparse representation:

```r
> Ac <- as(t(A), "CsparseMatrix")
> str(Ac)

Formal class ’dgCMatrix’ [package "Matrix"] with 6 slots
  ..@ i    : int [1:30] 1 13 14 15 8 14 15 16 5 15 ... 
  ..@ p    : int [1:11] 0 1 4 8 12 17 23 29 30 30 ... 
  ..@ Dim  : int [1:2] 20 10 
  ..@ Dimnames:List of 2 
     ..$ : NULL 
     ..$ : NULL 
  ..@ x    : num [1:30] 7 30 60 90 14 30 60 90 21 30 ... 
  ..@ factors : list()

Column *index* slot `j` replaced by a column *pointer* slot `p`.  

Classes for Matrices: well-defined inheritance hierarchies:

1. Content kind: Classes `dMatrix`, `lMatrix`, `nMatrix`, (`iMatrix`, `zMatrix`) for contents of `double`, `logical`, `pattern` (and not yet `integer` and `complex`) Matrices, where `nMatrix` only stores the location of non-zero matrix entries (where as logical Matrices can also have `NA` entries)

2. Sparsity: `denseMatrix`, `sparseMatrix`

3. Structure: general, triangular, symmetric, diagonal Matrices

Inheritance: Visualisation via graphs

1. Multiple Inheritance (of classes)

4. Multiple Dispatch (of methods)
Multiple Dispatch in S4 .... for Matrix operations

Methods for ”Matrix”-matrices: Often 2 matrices involved..

1. \( x \%*\% y \)
2. `crossprod(x,y) \rightarrow x^T y`
3. `tcrossprod(x,y) \rightarrow xy^T`
4. \( x + y \) — "Arith" group methods
5. \( x \leq y \) — "Compare" group methods

and many many more.

S4 >> S3

- S4 - multiple dispatch: Find method according to classes of both (or more) arguments.
- S3 - single dispatch: e.g., ”ops.Matrix”: only first argument counts.
Goals of Matrix package

1. Interface to LAPACK = state-of-the-art numerical linear algebra for dense matrices
   - making use of special structure for symmetric or triangular matrices (e.g. when solving linear systems)
   - setting and keep such properties allows more optimized code in these cases.

2. Sparse matrices for large designs: regression, mixed models, etc

3. ....... [omitted in this talk]

Hence, quite a few different classes for matrices.
many Matrix classes ...

> library(Matrix)
> length(allCl <- getClasses("package:Matrix"))

[1] 98

> ## Those called "...Matrix" :
> length(M.Cl <- grep("Matrix$", allCl, value = TRUE))

[1] 70

i.e., many ..., each inheriting from root class "Matrix"

> str(subs <- showExtends(getClassDef("Matrix")@subclasses,
+ printTo=FALSE))

List of 2
$ what: chr [1:76] "compMatrix" "triangularMatrix" "dMatrix" "iMatrix" ...
$ how : chr [1:76] "directly" "directly" "directly" "directly" ...

> ## even more... : All those above and these in addition:
> subs$what[ ! (subs$what %in% M.Cl)]

[1] "Cholesky"    "pCholesky"    "BunchKaufman" "pBunchKaufman"

...... a bit messy ......
3-way Partitioning of “Matrix space”

Logical organization of our Matrices: Three (3) main “class classifications” for our Matrices, i.e., three “orthogonal” partitions of “Matrix space”, and every Matrix object’s class corresponds to an intersection of these three partitions. i.e., in R’s S4 class system: We have three independent inheritance schemes for every Matrix, and each such Matrix class is simply defined to contain three virtual classes (one from each partitioning scheme), e.g.,

```r
setClass("dgCMatrix",
    contains= c("CsparseMatrix", "dsparseMatrix", "generalMatrix"),
    validity= function(..) .....)
```

Martin Maechler, Doug Bates (R Core)
The three partitioning schemes are

1. **Content type:** Classes `dMatrix`, `lMatrix`, `nMatrix`, (`iMatrix`, `zMatrix`) for entries of type `double`, `logical`, `pattern` (and not yet `integer` and `complex`) Matrices.
   `nMatrix` only stores the *location* of non-zero matrix entries (where as `logical` Matrices can also have `NA` entries!)

2. **Structure:** general, triangular, symmetric, diagonal Matrices

3. **Sparsity:** `denseMatrix`, `sparseMatrix`

First two schemes: a slight generalization from `LAPACK` for dense matrices.
3D space of Matrix classes

three-way partitioning of Matrix classes visualized in 3D space, dropping the final Matrix, e.g., "d" instead of "dMatrix":

```r
> d1 <- c("d", "l", "n")
> d2 <- c("general", "symmetric", "triangular", "diagonal")
> d3 <- c("dense", c("Csparse", "Tsparse", "Rsparse"))
> clGrid <- expand.grid(dim1 = d1, dim2 = d2, dim3 = d3, KEEP.OUT.ATTRS = FALSE)
> clGr <- data.matrix(clGrid)
> library(scatterplot3d)
used for visualization:
```

![3D space of Matrix classes diagram](image-url)
3-fold classification — Matrix naming scheme

1. “Actual” classes: Matrix objects are of those; the above “points in 3D space”

2. Virtual classes: e.g. the above coordinate axes categories. Superclasses of actual ones cannot have objects of, but —importantly— many methods for these virtual classes.

Actual classes follow a “simple” terse naming convention:

```r
> str(M3cl <- grep("^...Matrix\$", M.Cl, value = TRUE))
```

```
chr [1:47] "corMatrix" "ddiMatrix" "dgCMatrix" "dgeMatrix" ...
```

```r
> substring(M3cl,1,3)
```

```
[1] "cor" "ddi" "dgC" "dge" "dgR" "dgT" "dpo" "dpp" "dsC" "dsp" "dsR" "dsT"
[13] "dsy" "dtC" "dtp" "dtr" "dtR" "dtT" "ldi" "lgC" "lge" "lgR" "lgT" "lgT"
[25] "lsp" "lSR" "lST" "lsy" "lTC" "ltp" "ltr" "lTR" "lT" "ngC" "nge" "ngR"
[37] "ngT" "nsC" "nsp" "nsR" "nsT" "nsy" "ntC" "ntp" "ntR" "ntT"
```

```r
> M3cl <- M3cl[M3cl != "corMatrix"] # corMatrix not desired in following
```

Martin Maechler, Doug Bates (R Core)  Sparse Matrices in Matrix pkg  useR!, Rennes 2009  20 / 34
3D space of Matrix classes

- Rsparse
- Tsparse
- Csparse
- dense

- general
- symmetric
- triangular
- diagonal

dim1, dim2, dim3
Matrix 3d space: filled (2)
Matrix 3d space: filled (3)
Matrix 3d space: filled (4)
Spatially Dependent Errors — SAR(1)

Regression with spatially dependent errors; observations at locations $i, \ i = 1, \ldots, n$, $n$ in the thousands, possibly 100,000s.

Simultaneous Autoregression

$$y = X\beta + u \quad \text{where} \quad u = \lambda W u + \epsilon. \quad (1)$$

- $W$: matrix $(W_{ij})$ of “distance-based contiguities” of locations $i$ and $j$ ($W_{ii} \equiv 0$).
- $\lambda$: SAR(1) parameter; estimate via MLE, ($\beta$ profiled out).
- $u \sim \mathcal{N}(0, \sigma^2 (I - \lambda W)^{-1} (I - \lambda W^T)^{-1})$
- For log likelihood, need to compute determinant $|I - \lambda W| = (-\lambda)^n |-W + \frac{1}{\lambda} I|$ for many $\lambda$.

Compute Cholesky / Determinant of $A + \rho I$ for large sparse symmetric $A$: $\implies$ Fast Cholesky Update
Data provided by Roger Bivand, as a relevant test case:

```r
> data(USCounties, package="Matrix")
> dim(USCounties)

[1] 3111 3111

> (n <- ncol(USCounties))

[1] 3111

> IM <- .symDiagonal(n)
> nWC <- -USCounties
> set.seed(1)
> rho <- sort(runif(50, 0, 1)) ## rho = 1 / lambda

and now compute \( \text{determinant}(A) =: |A| \)

\[
|I - \lambda W| \propto \left| -W + \frac{1}{\lambda} I \right| \quad \text{for many} \ \lambda \text{'s.}
\]
SAR(1) – Cholesky Update – 2 –

> ## Determinant : Direct Computation
> system.time(MJ <- sapply(rho, function(x)
+    determinant(IM - x * USCounties, logarithm = TRUE)$modulus))

    user  system elapsed
       3.620    0.124    4.006

> ## Determinant : "high-level" Update of the Cholesky {Simplicial / Supernodal}
> C1 <- Cholesky(nWC, Imult = 2)
> system.time(MJ1 <- n * log(rho) +
+    sapply(rho, function(x) c(determinant(update(C1, nWC, 1/x))$mod)))

    user  system elapsed
       0.700    0.012    0.722

> stopifnot(all.equal(MJ, MJ1))
> C2 <- Cholesky(nWC, super = TRUE, Imult = 2) ## <-- "Supernodal"
> system.time(MJ2 <- n * log(rho) +
+    sapply(rho, function(x) c(determinant(update(C2, nWC, 1/x))$mod)))

    user  system elapsed
       0.804    0.020    0.859
> stopifnot(all.equal(MJ, MJ2))
> ## Determinant : "low-level" Update of the Cholesky {Simplicial / Supernodal}
> system.time(MJ3 <- n*log(rho) + Matrix:::ldetL2up(C1, nWC, 1/rho))

    user  system elapsed
 0.404   0.012   0.454

> stopifnot(all.equal(MJ, MJ3))
> system.time(MJ4 <- n*log(rho) + Matrix:::ldetL2up(C2, nWC, 1/rho))

    user  system elapsed
 0.384   0.008   0.405

> stopifnot(all.equal(MJ, MJ4))

Findings:

1. Using Cholesky update: order of magnitude faster
2. Simplicial (super= FALSE) ↔ Supernodal (super= TRUE) : no big difference here
3. An even faster method for Det(Chol(.)) yields another 50% speed.
In (linear) mixed effects, the evaluation of the (RE) likelihood or equivalently deviance, needs repeated Cholesky decompositions of

$$U_\theta U_\theta^\top + I,$$ (3)

for many $\theta$ values (= the relative variance components) and (often very large), very sparse matrix $U_\theta$ where only the non-zeros of $U$ depend on $\theta$, i.e., the sparsity pattern is given (by the observational design).

Sophisticated (fill-reducing) Cholesky done in two phases:

1. “symbolic” decomposition: Determine the non-zero entries of $L$ ($LL^\top = UU^\top + I$),
2. numeric phase: compute these entries.

Phase 1: typically takes much longer; only needs to happen once.
Phase 2: “update the Cholesky Factorization”
Who’s the best liked prof at ETH?

- Private donation for encouraging excellent teaching at ETH
- Student union of ETH Zurich organizes survey to award prizes: Best lecturer — of ETH, and of each of the 14 departments.
- Smart Web-interface for survey: Each student sees the names of his/her professors from the last 4 semesters and all the lectures that applied.
- Ratings in \{1, 2, 3, 4, 5\}.
- High response rate
Who's the best prof — data

```r
> md <- within(read.csv("~/R/MM/Pkg-ex/lme4/puma-lmertest.csv"), {
+   s <- factor(s) # Student_ID
+   d <- factor(d) # Lecturer_ID ("d"ozentIn)
+   dept <- factor(dept)
+   service <- factor(service)
+   studage <- ordered(studage)## *ordered* factors
+   lectage <- ordered(lectage) })
> str(md)

'data.frame': 73421 obs. of 7 variables:
$ s : Factor w/ 2972 levels "1","2","3","4",...: 1 1 1 1 2 2 3 3 3 3 ...
$ d : Factor w/ 1128 levels "1","6","7","8",...: 525 560 832 1068 62 406 3 6 19 75 ...
$ studage: Ord.factor w/ 4 levels "2"<"4"<"6"<"8": 1 1 1 1 1 1 1 1 1 1 ...
$ lectage: Ord.factor w/ 6 levels "1"<"2"<"3"<"4"<...: 2 1 2 2 1 1 1 1 1 1 ...
$ service: Factor w/ 2 levels "0","1": 1 2 1 2 1 1 2 1 1 1 ...
$ dept : Factor w/ 15 levels "1","2","3","4",...: 15 5 15 12 2 2 14 3 3 3 ...
$ y : int 5 2 5 3 2 4 4 5 5 4 ...
```
Modelling the ETH teacher ratings

Model: The rating depends on

- students (s) (rating subjectively)
- teacher (d) – main interest
- department (dept)
- “service” lecture or “own department student”, (service: 0/1).
- semester of student at time of rating (studage∈ {2, 4, 6, 8}).
- how many semesters back was the lecture (lectage).

Main question: **Who’s the best prof?**

Hence, for “political” reasons, want d as a **fixed** effect.
Model for ETH teacher ratings

Want d ("teacher_ID", \(\approx\) 1000 levels) as \textbf{fixed} effect. Consequently, in

\[ y = X\beta + Zb + \epsilon \]

have \(X\) as \(n \times 1000\) (roughly), \(Z\) as \(n \times 5000\), \(n \approx 70'000\).

```r
> fm0 <- lmer2(y ~ d + dept*service + studage + lectage + (1|s),
+             data = md, sparseX = TRUE)
```

\texttt{sparseX = TRUE}: \textit{sparse} \(X\) (fixed effects) in addition to the indispensably sparse \(Z\) (random effects).

Unfortunately: Here, the above "sparseX - lmer" ends in

\texttt{Error ... Cholmod error 'not positive definite' at file:../Cholesky/......}

\textbf{Good News: Newly in Matrix:}

\begin{verbatim}
sparse.model.matrix()
\end{verbatim}

- which \texttt{lmer()} can use,
- or you can use for "truly sparse" least squares (i.e. no intermediately dense design matrix)
- something we plan to provide in Matrix 1.0-0.
Summary

- **Recommended R package ”Matrix”**
- *Sparse* Matrices: in increasing number of applications
- S4 classes and methods are the natural implementation tools
- lme4 is going to contain an alternative “pure R” version of ML and REML, you can pass to `nlminb()` (or `optim()` if you must :-). UseRs can easily extend these R functions to more flexible models or algorithms.
- Matrix 1.0-0
  - will happen
  - will contain `sparse.model.matrix()
  - will contain truly sparse `lm(*, sparse=TRUE)`

That’s all folks — with thanks for your attention!
Summary

- **Recommended R package ”Matrix”**
- **Sparse Matrices:** in increasing number of applications
  - S4 classes and methods are the natural implementation tools
  - lme4 is going to contain an alternative “pure R” version of ML and REML, you can pass to `nlminb()` (or `optim()` if you must :-). UseRs can easily extend these R functions to more flexible models or algorithms.
  - **Matrix 1.0-0**
    - @ will happen
    - @ will contain `sparse.model.matrix`
    - @ will contain truly sparse `lm(*, sparse=TRUE)`

That’s all folks — with thanks for your attention!

Martin Maechler, Doug Bates (R Core)
Summary

- Recommended R package "Matrix"
- *Sparse* Matrices: in increasing number of applications
- S4 classes and methods are the natural implementation tools
- lme4 is going to contain an alternative “pure R” version of ML and REML, you can pass to `nlminb()` (or `optim()` if you must :-). UseRs can easily extend these R functions to more flexible models or algorithms.
- Matrix 1.0-0
  - will happen
  - will contain `sparse.model.matrix()`
  - will contain truly sparse `lm(*, sparse=TRUE)`

That’s all folks — with thanks for your attention!
Summary

- Recommended R package "Matrix"
- *Sparse* Matrices: in increasing number of applications
- S4 classes and methods are the natural implementation tools
- lme4 is going to contain an alternative “pure R” version of ML and REML, you can pass to `nlminb()` (or `optim()` if you must :-). UseRs can easily extend these R functions to more flexible models or algorithms.
- Matrix 1.0-0
  - will happen
  - will contain `sparse.model.matrix()`
  - will contain truly sparse `lm(*, sparse=TRUE)`

That’s all folks — with thanks for your attention!
Summary

- Recommended R package "Matrix"
- *Sparse* Matrices: in increasing number of applications
- S4 classes and methods are the natural implementation tools
- lme4 is going to contain an alternative "pure R" version of ML and REML, you can pass to `nlminb()` (or `optim()` if you must :-) . UseRs can easily extend these R functions to more flexible models or algorithms.
- Matrix 1.0-0
  - will happen
  - will contain `sparse.model.matrix()`
  - will contain truly sparse `lm(*, sparse=TRUE)`

That's all folks — with thanks for your attention!
Summary

- **Recommended R package “Matrix”**
- *Sparse* Matrices: in increasing number of applications
- S4 classes and methods are the natural implementation tools
- lme4 is going to contain an alternative “pure R” version of ML and REML, you can pass to `nlminb()` (or `optim()` if you must :-). UseRs can easily extend these R functions to more flexible models or algorithms.
- **Matrix 1.0-0**
  - 1. will happen
  - 2. will contain `sparse.model.matrix()`
  - 3. will contain truly sparse `lm(*, sparse=TRUE)`

That’s all folks — with thanks for your attention!
Summary

- Recommended R package "Matrix"
- *Sparse* Matrices: in increasing number of applications
- S4 classes and methods are the natural implementation tools
- lme4 is going to contain an alternative "pure R" version of ML and REML, you can pass to nlminb() (or optim() if you must :-). UseRs can easily extend these R functions to more flexible models or algorithms.
- Matrix 1.0-0
  1. will happen
  2. will contain sparse.model.matrix()
  3. will contain truly sparse lm(*, sparse=TRUE)

That’s all folks — with thanks for your attention!

Martin Maechler, Doug Bates (R Core)
Summary

- Recommended R package "Matrix"
- Sparse Matrices: in increasing number of applications
- S4 classes and methods are the natural implementation tools
- lme4 is going to contain an alternative “pure R” version of ML and REML, you can pass to `nlminb()` (or `optim()` if you must :-). UseR can easily extend these R functions to more flexible models or algorithms.
- Matrix 1.0-0
  1. will happen
  2. will contain `sparse.model.matrix()`
  3. will contain truly sparse `lm(*, sparse=TRUE)`

That’s all folks — with thanks for your attention!
Summary

- **Recommended R package** ”Matrix”
- **Sparse Matrices**: in increasing number of applications
- **S4 classes and methods** are the natural implementation tools
- **lme4** is going to contain an alternative “pure R” version of ML and REML, you can pass to `nlminb()` (or `optim()` if you must :-). UseRs can easily extend these R functions to more flexible models or algorithms.

- **Matrix 1.0-0**
  1. will happen
  2. will contain `sparse.model.matrix()`
  3. will contain truly sparse `lm(*, sparse=TRUE)`

That’s all folks — with thanks for your attention!

Martin Maechler, Doug Bates (R Core)
Summary

- Recommended R package ”Matrix"
- *Sparse* Matrices: in increasing number of applications
- S4 classes and methods are the natural implementation tools
- lme4 is going to contain an alternative “pure R” version of ML and REML, you can pass to \texttt{nlminb()} (or \texttt{optim()} if you must :-). UseRs can easily extend these R functions to more flexible models or algorithms.
- Matrix 1.0-0
  1. will happen
  2. will contain \texttt{sparse.model.matrix()}
  3. will contain truly sparse \texttt{lm(*, sparse=TRUE)}

That's all folks — with thanks for your attention!