Sparse Matrices for Large Data Modeling

Martin Maechler

ETH Zurich
Switzerland
maechler@R-project.org

useR!2007, Ames, Iowa, USA
Aug 9 2007

Overview

- Large Data – not just sparse matrices
- (Sparse) Matrices for Large Data : Applications
  - LMER – talk by Doug Bates
  - * Quantile Smoothing with Constraints
  - Regression Splines for "Total Variation Penalized" Densities; notably in 2D (triograms)
  - * Sparse Least Squares Regression (and Generalized, Robust,...)
  - Sparse Matrix ⟷ Graph incidence (in "Network")
  - Sparse Covariance or Correlation Matrices and Conditional Variance via sparse arithmetic
  - * Teaser of a case study: ETH professor evaluation by students: Who's the best — in teaching?
- Overview of Sparse Matrices
  - sparse matrix storage
  - Sparse Matrices in R's `Matrix`: arithmetic, indexing,
  - `solve` methods, possibly even for sparse RHS.
  - Sparse Matrices factorizations: chol(), qr, Schur, LU, Bunch-Kaufmann

Acknowledgements

Doug Bates has introduced me to sparse matrices and the `Matrix` package, and the last two years have been a lot of fun in collaboration with him.

Large Data Analysis

This session "Large Data" will focus on one important kind of large data analysis, namely: Sparse Matrix modelling.
Further considerations a useR should know:

1. Think first, then "read" the data
   - *Read the docs!* – The "**R Data Import/Export**" manual (part of the R manuals that come with R and are online in PDF and HTML).
   - Note section 2.1 *Variations on '`read.table`'*; and read `help(read.table)`, notably about the `colClasses` and `as.is` arguments.
   - How large is "large"?
   - Do I only need some variables?
   - Should I use a database system from R (SQLite, MySQL)?
   - Rather work with simple random samples of rows ?!
2. Think again:
   "First plot, then model!" (T-shirt); or slightly generalized:
   "First explore, then model!",
   i.e., first use exploratory data analysis (EDA).
3. ......

Constrained Quantile Smoothing Splines

Pin Ng (1996) defined quantile smoothing splines, as solution of, e.g.,

$$\min_g \sum_{i=1}^{n} \rho_\tau(y_i - g(x_i)) + \lambda \cdot \max_x \left|g''(x)\right|. \tag{1}$$

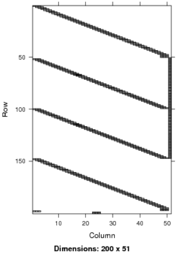
as a nonparametric estimator for $g_\tau(x)$, $\tau \in (0,1)$, where for $\tau = \frac{1}{2}$, $\sum_i \rho_\tau(r_i) = \sum_i |r_i|$ is least absolute values (L_1) regression.

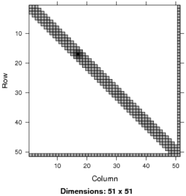
Solving (1) means linear optimization with linear constraints, and hence can easily be extended by further (linear) constraints such *monotonicity*, *convexity*, upper bounds, exact fit constraints, etc. The matrix $\boldsymbol{X}$ corresponding to the linear optimization problem for the constrained smoothing splines is of dimension $(f \cdot n) \times n$ but has only $f_2 \cdot n$ ($f_2 \approx 3$) non-zero entries.

Example: constraints on $g(.)$ are: $g()$ increasing, i.e., $g'(x) > 0$, and

$$0 < g(-3) \le g(0) = 0.5 \le g(3) < 1,$$

and $n = 50$ observations, the matrices $\boldsymbol{X}$ and $\boldsymbol{X}^\intercal \boldsymbol{X}$ are



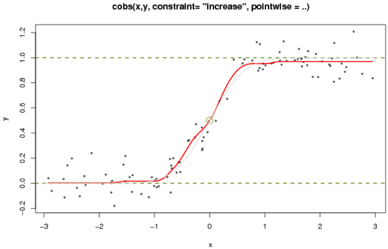


constrained B-spline fit

Fit a constrained (B-) smoothing spline to $n = 100$ data points constrained to be monotone increasing, and fulfull the 3 pointwise constraints (above):

```
> library(cobs)
> Phi.cnstr <- rbind(c( 1, -3, 0),    ## g(-3) >= 0
+                    c(-1,  3, 1),    ## g(+3) <= 1
+                    c( 0,  0, 0.5)) ## g( 0) == 0.5
> msp <- cobs(x, y, nknots = length(x) - 1,
+             constraint = "increase", pointwise = Phi.cnstr,
+             lambda = 0.1)
```

```
> ## msp <- cobs(.........)
> plot(msp, main = "cobs(x,y, constraint= \"increase\", pointwis
> abline(h = c(0,1), lty=2, col = "olivedrab", lwd = 2)
> points(0, 0.5, cex=2,  col = "olivedrab")
> lines(xx, pnorm(2 * xx), col="light gray")# true function
```

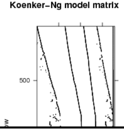


Sparse Least Squares

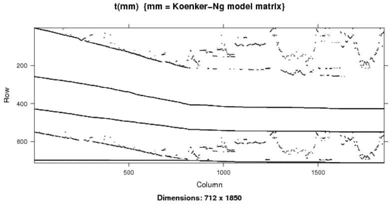
Koenker and Ng (2003) were the first to provide a sparse matrix package for R, including sparse least squares, via `slm.fit(x,y, ...)`.

They provide the following nice example of a model matrix (probably from a quantile smoothing context):

```
> library(Matrix)
> data(KNex)     # Koenker-Ng ex{ample}
> dim (KNex$mm)
[1] 1850  712
> print(image(KNex$mm, aspect = "iso", colorkey = FALSE,
+              main = "Koenker-Ng model matrix"))
```



or rather transposed, for screen display:



Cholesky for Sparse L.S.

For *sparse* matrices, the Cholesky decomposition has been the most researched factorization and hence used for least squares regression modelling. Estimating $\boldsymbol\beta$ in the model $\boldsymbol y = \boldsymbol X\boldsymbol\beta + \boldsymbol\epsilon$ by solving the *normal equations*

$$\boldsymbol X^{\mathsf T}\boldsymbol X\boldsymbol\beta = \boldsymbol X^{\mathsf T}\boldsymbol y \quad (2)$$

the Cholesky decomposition of the (symmetric positive semi-definite) $\boldsymbol X^{\mathsf T}\boldsymbol X$ is $\boldsymbol L\boldsymbol L^{\mathsf T}$ or $\boldsymbol R^{\mathsf T}\boldsymbol R$ with lower-**L**eft upper-**R**ight *triangular* matrix $\boldsymbol L$ or $\boldsymbol R \equiv \boldsymbol L^{\mathsf T}$, respectively.

System solved via two triangular (back- and forward-) "solves":

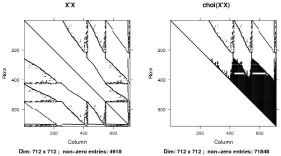
$$\hat{\boldsymbol\beta} = (\boldsymbol X^{\mathsf T}\boldsymbol X)^{-1}\boldsymbol X^{\mathsf T}\boldsymbol y = (\boldsymbol L\boldsymbol L^{\mathsf T})^{-1}\boldsymbol X^{\mathsf T}\boldsymbol y = \boldsymbol L^{-\mathsf T}\boldsymbol L^{-1}\boldsymbol X^{\mathsf T}\boldsymbol y \quad (3)$$

CHOLMOD (Tim Davis, 2006): Efficient sparse algorithms.

Cholesky – "Fill-in"

The usual Cholesky decomposition works, ...

```
> X.X <- crossprod(KNex$mm)
> c1 <- chol(X.X)
> image(X.X, main= "X'X", aspect="iso", colorkey = FALSE)
> image(c1,  main= "chol(X'X)", ......)
```



but the resulting cholesky factor has suffered from so-called fill-in, i.e., its sparsity is quite reduced compared to $\boldsymbol X^{\mathsf T}\boldsymbol X$.

Fill-reducing Permutation

So, `chol(X'X)` suffered from *fill-in* (sparsity decreased considerably).

Solution: *Fill-reducing* techniques
which permute rows and columns of $X^\intercal X$, i.e., use $PX^\intercal XP'$ for a *permutation* matrix P or in R syntax, `X.X[pvec,pvec]` where `pvec` is a permutation of `1:n`.

The *permutation* P is chosen such that the Cholesky factor of $PX^\intercal XP'$ is as sparse as possible

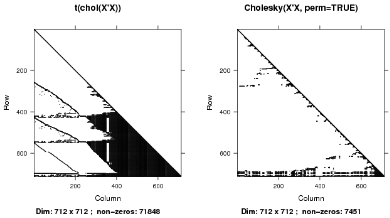
```
>  image(t(c1), main= "t( chol(X'X) )", .........)
>  c2 <- Cholesky(X.X, perm = TRUE)
>  image(c2,    main= "Cholesky(X'X, perm = TRUE)", ........)
```

((Note that such permutations are done for dense `chol()` when `pivot=TRUE`, but there the goal is dealing with rank-deficiency.))

Timing – Least Squares Solving

```
> y <- KNex$y
> m. <- as(KNex$mm, "matrix") # traditional (dense) Matrix
> system.time(cpod.sol <- solve(crossprod(m.), crossprod(m., y)))

   user  system elapsed
  2.111   0.001   2.119

> ## Using sparse matrices is so fast, we have to bump the time
> system.time(for(i in 1:10) ##  sparse solution withOUT permuta
+             sp1.sol <- solve(c1,solve(t(c1), crossprod(KNex$mm

   user  system elapsed
  0.049   0.000   0.048

> system.time(for(i in 1:10) ## sparse Cholesky  WITH  fill-redu
+             sp2.sol <- solve(c2, crossprod(KNex$mm, y)))

   user  system elapsed
  0.009   0.000   0.010

> stopifnot(all.equal(sp1.sol, sp2.sol),
+           all.equal(as.vector(sp2.sol), c(cpod.sol)))
```

Fill-reducing Permutation - Result



Teaser case study: Who's the best prof?

- 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

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`$\in \{2, 4, 6, 8\}$).
- how many semesters back was the lecture (`lectage`).

Main question: $\boxed{\text{Who's the best prof?}}$
Hence, for "political" reasons, want `d` as a **fixed** effect.

Who's the best prof — data

```
> ## read the data; several factor assignments, such as
> md$d  <- factor(md$d) # Lecturer_ID ("d"ozentIn)
> 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
 $ d      : Factor w/ 1128 levels "1","6","7","8",..: 525 560 832 1068
 $ 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
 $ 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
 $ y      : int  5 2 5 3 2 4 4 5 5 4 ...
```

Model for ETH teacher ratings

Want `d` ("teacher_ID", ≈ 1000 levels) as **fixed** effect.
Consequently, in

$$y = \boldsymbol{X\beta} + \boldsymbol{Zb} + \boldsymbol{\epsilon}$$

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

```
> fm0 <- lmer(y ~ d*dept + dept*service + studage + lectage + (1
+              data = md)
Error in model.matrix.default(mt, mf, contrasts) :
cannot allocate vector of length 1243972003
> 1243972003 / 2^20 ## number of Mega bytes
[1] 1186.344
```

$\longrightarrow$ Want **sparse** matrices for $\boldsymbol{X}$ *and* $\boldsymbol{Z}$ and crossprods, etc.

Intro to Sparse Matrices in R package Matrix

- The R Package `Matrix` contains dozens of matrix classes and hundreds of method definitions.
- Has sub-hierarchies of `denseMatrix` and `sparseMatrix`.
- Very basic intro in *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,] . . . . .  .  .  .  .  . . . . . . . . . . .
```

simple example – 2 –

```
> 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)
```

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 humbly humans, but can be both made smaller *and* more efficient for (column-access heavy) operations:
The "column compressed" sparse representation, (virtual class `CsparseMatrix` in `Matrix`):

```
> 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`.

Conclusions

- Sparse Matrices: crucial for several important data modelling situations
- There's the R package `Matrix`
- ...

Many ? more conclusions at the end of Doug Bates' talk :-)

Other R packages for large "matrices"

- `biglm` – updating QR decompositions; storing $O(p^2)$ instead of $O(n \times p)$.
- `R.Huge`: Using class `FileMatrix` to store matrices on disk instead of RAM memory.
- `SQLiteDF` by Miguel Manese ("our" Google Summer of Code project 2006).
  `Description`: Transparently stores data frames & matrices into SQLite tables.
- ...
- `sqldf` by Gabor Grothendieck: learn SQL $\longleftrightarrow$ R
- ...
- `ff` package (memory mapping arrays) by Adler, Nendic, Zucchini and Glaser: poster of today and .... winner of the useR!2007 programming competition.