Factors and Sparse Model Matrices

Martin Maechler
R Core Development Team
maechler@R-project.org

July 2007, 2008 (typeset on July 25, 2008)

Model matrices in our (generalized) linear models (`lm`) are often practically sparse — whenever categorical predictors are used.

Let's start with an artifical small example:

```
>  (ff <- factor(substring("statistics", 1:10, 1:10), levels=letters))

 [1] s t a t i s t i c s
Levels: a b c d e f g h i j k l m n o p q r s t u v w x y z

>  factor(ff)      # drops the levels that do not occur

 [1] s t a t i s t i c s
Levels: a c i s t

>  (f. <- ff[, drop=TRUE]) # the same, more transparently

 [1] s t a t i s t i c s
Levels: a c i s t

>  library(Matrix)
>  Matrix(contrasts(f.)) # "treatment" contrasts by default -- level "a" = baseline

5 x 4 sparse Matrix of class "dgCMatrix"
  c i s t
a . . . .
c 1 . . .
i . 1 . .
s . . 1 .
t . . . 1

>  Matrix(contrasts(C(f., sum)))

5 x 4 sparse Matrix of class "dgCMatrix"

a  1  .  .  .
c  .  1  .  .
i  .  .  1  .
s  .  .  .  1
t -1 -1 -1 -1

>  Matrix(contrasts(C(f., helmert)), sparse=TRUE) # S-plus default; much less sparse
```

```
5 x 4 sparse Matrix of class "dgCMatrix"

a -1 -1 -1 -1
c  1 -1 -1 -1
i  .  2 -1 -1
s  .  .  3 -1
t  .  .  .  4
```

where contrasts is (conceptually) just one major ingredient in the well-known `model.matrix()` function. Since 2007, the **Matrix** package has been providing coercion from a `factor` object to a `sparseMatrix` one to produce the transpose of the model matrix corresponding to a model with that factor as predictor (and no intercept):

```
> as(f., "sparseMatrix")
```

```
5 x 10 sparse Matrix of class "dgCMatrix"

a . . 1 . . . . . . .
c . . . . . . . . 1 .
i . . . . 1 . . 1 . .
s 1 . . . . 1 . . . 1
t . 1 . 1 . . 1 . . .
```

which is really almost the transpose of using the above sparsification of `contrasts()`,

```
> t( Matrix(contrasts(f.))[as.character(f.),] )
```

```
4 x 10 sparse Matrix of class "dgCMatrix"

c . . . . . . . . 1 .
i . . . . 1 . . 1 . .
s 1 . . . . 1 . . . 1
t . 1 . 1 . . 1 . . .
```

and that is the same as the "sparsification" of `model.matrix()`, apart from the column names (here transposed),

```
> t( Matrix(model.matrix(~ 0 + f.)) )
```

```
5 x 10 sparse Matrix of class "dgCMatrix"

f.a . . 1 . . . . . . .
f.c . . . . . . . . 1 .
f.i . . . . 1 . . 1 . .
f.s 1 . . . . 1 . . . 1
f.t . 1 . 1 . . 1 . . .
```

In situations with more than one factor, particularly with interactions, the model matrix is currently not directly available via **Matrix** functions and currently needs to go via the dense `model.matrix()` result:

```
> data(npk, package="MASS")
> npk.mf <- model.frame(yield ~ block + N*P*K, data = npk)
> ## str(npk.mf) # the data frame + "terms" attribute
>
> m.npk <- model.matrix(attr(npk.mf, "terms"), data = npk)
> class(M.npk <- Matrix(m.npk))
```

```
[1] "dgCMatrix"
attr(,"package")
[1] "Matrix"
```

```
> dim(M.npk)# 24 x 13  sparse Matrix
```

```
[1] 24 13
```

```
> t(M.npk) # easier to display, column names readably displayed as row.names(t(.))
```

```
13 x 24 sparse Matrix of class "dgCMatrix"

(Intercept) 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
block2      . . . . 1 1 1 1 . . . . . . . . . . . . . . . .
block3      . . . . . . . . 1 1 1 1 . . . . . . . . . . . .
block4      . . . . . . . . . . . . 1 1 1 1 . . . . . . . .
block5      . . . . . . . . . . . . . . . . 1 1 1 1 . . . .
block6      . . . . . . . . . . . . . . . . . . . . 1 1 1 1
N1          . 1 . 1 1 1 . . . 1 1 . 1 1 . . 1 . 1 . 1 1 . .
P1          1 1 . . . 1 . 1 1 1 . . . 1 . 1 1 . . 1 . 1 1 .
K1          1 . . 1 . 1 1 . . 1 . 1 . 1 1 . . . 1 1 1 . 1 .
N1:P1       . 1 . . . 1 . . . 1 . . . 1 . . 1 . . . . 1 . .
N1:K1       . . . 1 . 1 . . . 1 . . . 1 . . . . 1 . 1 . . .
P1:K1       1 . . . . 1 . . . 1 . . . 1 . . . . . 1 . . 1 .
N1:P1:K1    . . . . . 1 . . . 1 . . . 1 . . . . . . . . . .
```

An other example is the it seems realistic situation of a user who enquired on R-help (July 15, 2008, `https://stat.ethz.ch/pipermail/r-help/2008-July/167772.html`) about an “aov error with large data set”:

’m looking to analyze a large data set: a within-Ss 2*2*1500 design with 20 Ss. However, aov() gives me an error, reproducible as follows:

and gave the following code example (slightly edited):

```
> id <- factor(1:20)
> a <- factor(1:2)
> b <- factor(1:2)
> d <- factor(1:1500)
> aDat <- expand.grid(id=id, a=a, b=b, d=d)
> aDat$y <- rnorm(length(aDat[, 1])) # generate some random DV data
> dim(aDat) # 120'000 x 5  (120'000 = 2*2*1500 * 20 = 6000 * 20)
```

```
[1] 120000      5
```

and then continued with

```
m.aov <- aov(y ~ a*b*d + Error(id/(a*b*d)), data=aDat)
```

which yields the following error:
”

```
Error in model.matrix.default(mt, mf, contrasts) :
allocMatrix:  too many elements specified
```

`" Any suggestions?` to which he got the explanation by Peter Dalgaard that the formal model matrix involved was much too large in this case, and that PD assumed, **lme4** would be able to solve the problem. However, currently there would still be a big problem with using **lme4**, because of the many levels of *fixed* effects:

Specifically[1],

[1]the following is not run in R on purpose, rather just displayed here

```
dim(model.matrix( ~ a*b*d, data = aDat)) # 120'000 x 6000
```

where we note that $120'000 \times 6000 = 720$mio, which is $720'000'000 * 8/2^{20} \approx 5500$ Megabytes.

Unfortunately **lme4** does *not* use a sparse X-matrix for the fixed effects (yet), it just uses sparse matrices for the Z-matrix of random effects and sparse matrix operations for computations related to Z.

Let us use a smaller factor `d` in order to investigate how sparse the X matrix would be:

```
> d2 <- factor(1:150) # 10 times smaller
> tmp2 <- expand.grid(id=id, a=a, b=b, d=d2)
> dim(tmp2)

[1] 12000     4

> dim(mm <- model.matrix( ~ a*b*d, data=tmp2))

[1] 12000   600

> ## is 100 times smaller than original example
>
> class(smm <- Matrix(mm)) # automatically coerced to sparse

[1] "dgCMatrix"
attr(,"package")
[1] "Matrix"

> object.size(mm) / object.size(smm)

[1] 42.98469
```

shows that even for the small `d` here, the memory reduction would be more than an order of magnitude.

```
> image(smm, aspect=3, lwd=0, col.r = "red")
```

and working with the sparse instead of the dense model matrix is considerably faster as well,

```
> x <- 1:600
> system.time(y <- smm %*% x) ## sparse is much faster

   user  system elapsed
  0.003   0.000   0.004

> system.time(y. <- mm %*% x) ## than dense

   user  system elapsed
  0.072   0.000   0.073

> identical(as.matrix(y), y.) ## TRUE

[1] TRUE
```