Sparse Matrix Computation in R with an Application to GEEs

Lee S. McDaniel

LSUHSC - School of Public Health

Motivation - ODS

- Goal: Identify risk and prognostic factors for ADHD in early childhood
- Sampling: 255 subjects, about half cases/controls

  Cases: Referred by parent or teacher

  Controls: Matched demographically

- Followed up for 15 years (we have 8)
- **Analyze** time course of hyperactivity symptom count

Estimation

- Estimate $E(Y|X)$
- Use GEE to improve efficiency
- In the population, use canonical link:

$$\mu_P = g^{-1}(X'\beta)$$

- In the sample, relationship is:

$$\mu_S = \int y \times dF_S(y|X)$$

Need to fit GEEs with user-defined link functions.

What We Want

- An R package
- Solve GEEs w/ user-defined link and variance functions
- Do it quickly
- Easy to modify
- Purely in R without C++ elements

Created `geeM` package satisfying these criteria

Existing GEE Software

- SAS: GENMOD
- Stata: xtgee
- R: gee, geepack

GEE Computation

Estimate Coefficients by solving

$$\sum_{i=1}^{K} D_i^T V_i^{-1} S_i = 0$$

$$\hat{\beta}_{j+1} = \hat{\beta}_j - \left\{ \sum_{i=1}^{K} D_i^T V_i^{-1} D_i \right\}^{-1} \left\{ \sum_{i=1}^{K} D_i^T V_i^{-1} S_i \right\}$$

- D_i is related to the design matrix $(n_i \times p)$
- S_i is a vector $(n_i \times 1)$
- $V_i = A_i^{1/2} R_i(\alpha) A_i^{1/2}$
- A_i is diagonal $(n_i \times n_i)$
- $R_i(\alpha)$ is a working correlation matrix $(n_i \times n_i)$

GEE Computation

Can instead write

$$\hat{\beta}_{j+1} = \hat{\beta}_j - \left\{D^T V^{-1} D\right\}^{-1} \left\{D^T V^{-1} S\right\}$$

- D is related to stacked design matrices $\left(\sum_i n_i \times p\right)$
- S is a vector $\left(\sum_i n_i \times 1\right)$
- V is block diagonal $\left(\sum_i n_i \times \sum_i n_i\right)$

Sparse Matrix Storage

```
Formal class 'dsTMatrix' with 7 slots
  ..@ i        : int [1:5370] 0 0 1 0 1 2  ...
  ..@ j        : int [1:5370] 0 1 1 2 2 2  ...
  ..@ Dim      : int [1:2] 2148 2148
  ..@ Dimnames:List of 2
  .. ..$ : NULL
  .. ..$ : NULL
  ..@ x        : num [1:5370] 1.364 -0.303 ...
  ..@ uplo     : chr "U"
  ..@ factors : list()
```

Computing 0×0 takes just as long as $x \times y$

A Crazy Example

Add $I_{10,000}$ to itself:

```
m1 <- diag(10000)
system.time(for(i in 1:100) m1+m1)
object_size(m1)
```

Result: a leisurely 34.08s and bloated 800MB

```
library(Matrix)
M1 <- Diagonal(10000)
system.time(for(i in 1:100) M1+M1)
object_size(M1)
```

Result: a swift 0.05s and dainty 1.14kB

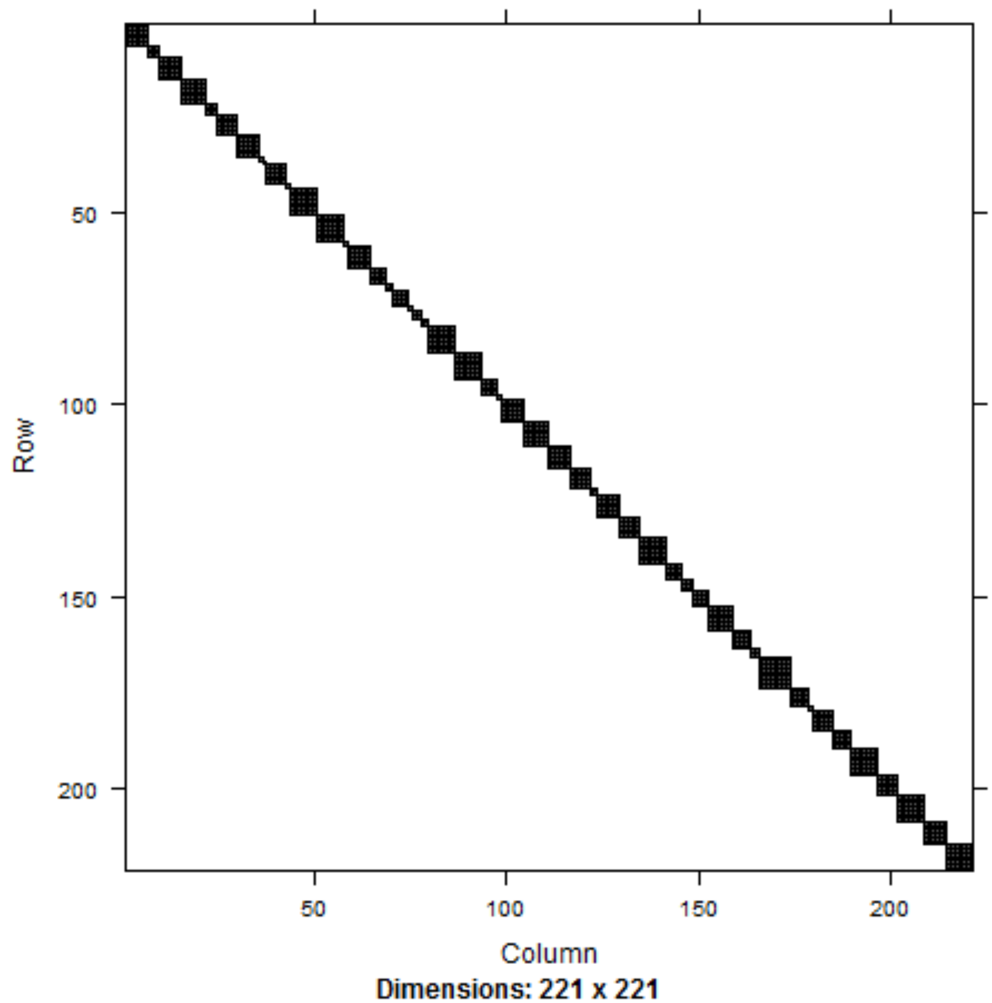
Creating Sparse Matrices

```
sparseMatrix(i, j, x, symmetric=FALSE)
```

- `i` is vector of row indices
- `j` is vector of column indices
- `x` is vector of contents
- `symmetric=TRUE` if the matrix is symmetric

And that's it

Anatomy of a Correlation Matrix



Anatomy of a Correlation Matrix

- Typically Sparse
- Block Diagonal
- Similarly-sized blocks are identical

Don't even need to invert entire matrix!

Inverting a Correlation Matrix

- Assume b different-sized blocks
- Build b matrices
- Invert each of them
- Build the block diagonal matrix

Special Case: AR-1

Correlation Matrix is defined by

$$R(\alpha) = \left(\alpha^{|t-t'|}\right)$$

Then

$$R(\alpha)^{-1} = \frac{1}{1-\alpha^2}\begin{bmatrix} 1 & -\alpha & 0 & 0 & \cdots 0 & 0 & \\ -\alpha & 1+\alpha^2 & -\alpha & 0 & \cdots & 0 & 0 \\ 0 & -\alpha & 1+\alpha^2 & -\alpha & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & 1+\alpha^2 & -\alpha \\ 0 & 0 & 0 & 0 & \cdots & -\alpha & 1 \end{bmatrix}$$

Or

$$R(\alpha)^{-1} = \frac{1}{1-\alpha^2}\left\{L - \alpha M + (1+\alpha^2)N\right\}$$

For Example: ohio Data Set in geepack

- Part of the Six City Study
- 537 children in Steubenville, Ohio
- Aged 7-10 years
- Followed for 4 years each
- Wheezing status is response (binary)
- Age, maternal smoking are predictors
- Full correlation matrix is 2148x2148

Three Methods of Inversion

First: invert sparse block matrix

```
user  system elapsed
0.046  0.000 0.046
```

Second: loop through and invert

```
user  system elapsed
0.028  0.000  0.028
```

Third: invert 4x4 matrix and build

```
user  system elapsed
0.016  0.000  0.016
```

The Matrix Multiplication

$$\sum_{i=1}^{K} D_i^T V_i^{-1} S_i$$

Using the block diagonal:

```
user  system elapsed
0.084  0.000  0.084
```

Looping through subjects:

```
user  system elapsed
0.772  0.000  0.777
```

This multiplication happens about 10 times

A Larger Example: Birth Data

- Mothers with 2 or 3 children
- 141,929 clusters
- Response is gestational age at birth
- Predictor is mother's age
- 296,218 observations

Matrix Inversion

First: invert sparse block matrix

```
user  system elapsed
A really long time
```

Second: loop through subjects and invert

```
user  system elapsed
7.99   0.00   7.99
```

Third: invert 2x2 and 3x3 matrices and build

```
user  system elapsed
1.82   0.04   1.86
```

The Matrix Multiplication

Using the block diagonal:

```
user  system elapsed
0.198  0.008  0.206
```

Looping through subjects:

```
user    system elapsed
302.45   0.02  303.24
```

Other R GEE Solvers

R package `gee`

- A basic solver
- Based on C
- Link and variance functions limited

R package `geepack`

- Regression models for scale and correlation parameters
- `glm`-like interface
- Based on C
- Link and variance functions limited

Speed Comparison: Ohio Data

	Independence		AR-1		Exchangeable		Unstructured	
	Avg. (s)	Rel.	Avg. (s)	Rel.	Avg. (s)	Rel.	Avg. (s)	Rel.
geeM	0.11	1.83	0.11	1.00	0.11	1.33	0.16	1.11
gee	0.07	1.17	0.12	1.07	0.08	1.00	0.15	1.00
geepack	0.06	1.00	0.18	1.53	0.12	1.48	0.20	1.34

Speed Comparison: Birth Data

	Independence		AR-1		Exchangeable		Unstructured	
	Avg. (s)	Rel.	Avg. (s)	Rel.	Avg. (s)	Rel.	Avg. (s)	Rel.
geeM	5.28	1.00	9.87	1.00	7.33	1.00	8.18	1.00
gee	10.02	1.90	29.39	2.98	20.09	2.74	21.49	2.63
geepack	9.66	1.83	23.67	2.40	22.59	3.08	25.67	3.14

Changing the Link Function

```
linkfun <- qcauchy
linkinv <- pcauchy
mu.eta <- dcauchy
variance <- function(p){p*(1-p)}
FunList <- list(linkfun, variance,
                linkinv, mu.eta)

geem(resp~age*smoke, id=id, data=ohio,
     family=FunList, corstr="exch")
```

Flexibility!

Changing the Link Function

Logit Link

```
 Inter    age   smoke age:smoke
 -1.9   -0.14   0.31   0.071
(0.12) (0.058) (0.19) (0.088)
```

Probit Link

```
 Inter    age   smoke age:smoke
 -1.1   -0.077   0.17   0.037
(0.063) (0.031) (0.10) (0.049)
```

Cauchit Link

```
 Inter    age   smoke age:smoke
 -2.3   -0.28   0.63   0.18
(0.27)  (0.13) (0.37) (0.17)
```

Back to the ADHD Example

A Further Complication

Inverse link is defined by:

$$\mu_S = \int y \times dF_S(y|X)$$

Q: What's the link function?

A: It doesn't matter.

Instead of using the mean, use the linear predictor

The Inverse Link

```
linkinv <- function(eta){
  ymat <- matrix(rep(y,nobs),nrow=nobs,byrow=T)
  etaymat <- apply(ymat, 2, "*", eta)
  reference <- -lfactorial(matrix(rep(y, nobs),
                                  nrow=nobs, byrow=T))

  exp(etaymat+reference+log(rho.y))%*%support
}
```

And, if you're curious

```
linkfun <- identity
```

ODS Simulations

- Response is marginally poisson with link

$$\log \mu_P = -1.4 + 0.4X_1 - 0.1t + 0.1X_1 t$$

- X_1 is binary
- $t = 0, \ldots, 7$
- Responses have an AR-1 correlation with $\alpha = 0.9$.
- If $Y_{i1} = 0$, probability of being sampled is 0.041
- If $Y_{i1} > 0$, probability of being sampled is 0.959

Simulated Results of ODS

Design	Estimation	$\beta_0 = -1.4$	$\beta_{x_1} = 0.4$	$\beta_t = -0.1$	$\beta_{tx_1} = 0.1$
SRS	GEE	1 (95)	0 (94)	2 (95)	2 (95)
ODS	Naive GEE	-42 (0)	-21 (88)	24 (84)	-11 (92)
	Corrected GEE	1 (95)	2 (95)	2 (94)	0 (95)
		[1.40]	[1.37]	[1.26]	[1.14]

Percent Bias (Coverage Probability) [Efficiency Relative to SRS]

Results from ADHD Study

	Naive		Corrected	
Intercept	4.01	(3.32, 4.90)	2.75	(2.23, 3.39)
t	0.98	(0.90, 1.05)	1.06	(0.97, 1.15)
$(t-2)_+$	0.95	(0.88, 1.03)	0.89	(0.80, 0.97)
age	0.87	(0.75, 1.00)	0.87	(0.75, 1.01)
sex	0.80	(0.53, 1.22)	0.62	(0.41, 0.93)
afr	1.58	(1.25, 2.03)	1.67	(1.30, 2.12)
other	1.11	(0.63, 1.95)	1.03	(0.58, 1.84)
sex*t	1.04	(0.84, 1.28)	1.07	(0.85, 1.35)
sex*$(t-2)_+$	0.91	(0.71, 1.16)	0.89	(0.68, 1.15)

Exponentiated Estimate (Confidence Interval). CI containing 1 indicates not significant.

References

L. S. McDaniel, N. C. Henderson, and P. J. Rathouz. Fast Pure R Implementation of GEE: Application of the Matrix Package. The R Journal, 5(1):181-188, June 2013.

Schildcrout, J. S. and Rathouz, P. J. (2010). Longitudinal Studies of Binary Response Data Following Case-Control and Stratified Case-Control Sampling: Design and Analysis. Biometrics, 66(2):365-373.