Matrices and Statistic with Applications
Computer Exercise: Sparse matrices. Solution of sparse LS problems

NGSSC, LU, SLU, UU

September, 2020

The tasks within this Lab are related to (i) sparse matrices and the effect of utilizing that, and (ii) solving LS problems by iterative methods. You are welcome to extend some of the exercises with additional tasks, relevant to the theme of this lab.
At the end, the results of the exercise have to be sketched and shown to the lab-consultant.
The tasks require programs and data files, which can be downloaded from
`http://user.it.uu.se/~maya/Courses/NGSSC/index_Stat_2020.html`.
Make a copy of these files in some of your directories.

Exercise 1 (For $\mathbb{R}$ users: $\mathbb{R}$ packages for sparse matrix handling)
Check the description of the packages 'SparseM' and 'spam' (on the web or from the provided files 'Spam_package.pdf' and 'SparseM_package.pdf'.
Install the packages and load them.

Exercise 2 (IBD matrices and how do we handle them (pen and paper))
"Identity by descent (IBD) matrix estimation is a central component in mapping of Quantitative Trait Loci (QTL) using variance component models. A large number of algorithms have been developed for estimation of IBD between individuals in populations at discrete locations in the genome for use in genome scans to detect QTL affecting various traits of interest in experimental animal, human and agricultural pedigrees."

Background: The task originates in variance component analysis in animal breeding, using the so-called Average Information Restricted Maximum Likelihood (AI-REML) method. The estimation of the variance components σ by REML for genetic mapping of quantitative traits can be formulated as an optimization problem, which reads as follows

$$\text{find} \quad \min L \tag{1}$$

$$\text{s.t. } \sigma_1 \geq 0, \sigma_2 > 0, \tag{2}$$

where the log-likelihood function, L, defined by

$$L \equiv -2ln(l) = C + ln(det(V)) + ln(det(X^T V^{-1} X)) + y^T P y \tag{3}$$

is derived from the linear mixed model determined by the basic relation

$$y = Xb + Qu + e. \tag{4}$$

Here y is a vector of n observations, X is the $n \times r$ design matrix for fixed effects, Q is the $n \times q$ design matrix for q random effects, b is the vector of r unknown fixed effects, u is the vector of q unknown random effects, and e is a vector of n residuals. We assume that the entries of e are identically and independently distributed and there is a single observation for each individual in the pedigree. This assumption implies that the variance-covariance matrix for the vector y determined by equation (4) is of the form

$$V = \sigma_a^2 A + \sigma_e^2 I \equiv \sigma_1 A + \sigma_2 I. \tag{5}$$

Here, $\sigma_1 \geq 0$ denotes the variance of the random effects and $\sigma_2 > 0$ denotes the residual variance. The matrix $X(n, r)$ is assumed to be full column rank.
Let A be an IBD matrix, assumed to be positive semi-definite (thus, can be singular) and not necessarily sparse.
We will consider the scenario when A is singular. A general technique to circumvent the singularity is to add a diagonal matrix D to A, where $D = \sigma * I$ and consider $V = A + D$ instead. Here I is the identity matrix of the corresponding size and σ is a properly chosen scalar parameter.
In some of the broadly used QTL analysis methods, one creates and performs operations with the following matrices:

$$P = V^{-1} - V^{-1}X(X^T V^{-1} X)^{-1} X^T V^{-1}$$

referred to as "the projection matrix", and

$$HI = \begin{pmatrix} y^T PAPAPy & y^T PAPPy \\ y^T PAPPy & y^T PPPy \end{pmatrix} \quad GL = -\begin{pmatrix} tr(AP) - y^T PAPy \\ tr(P) - y^T PPy \end{pmatrix}$$

which arise from a nonlinear solution procedure.
Suggest a numerical algorithm to compute the entries of HI and LG as cheap as possible. Remember that V is a sparse matrix. What is the computational complexity of your algorithm in terms of the size of the matrix A? Make some simplified derivations.
Hints: Observe that $PVP = P$. Would a spectral decomposition of A be of help? Or some other decomposition?

Exercise 3 (Operations with sparse matrices)
For this exercise, you could use both `Matlab` and $\mathbb{R}$.
Load the matrix `X20` from the file `20.dat`. The file contains only the lower triangular part of a sparse (symmetric) IBD matrix.

1. Recover the full-sized matrix from its lower-triangular part (let us name it A).
2. Visualize the structure of the matrix (`spy`)
3. Compute the eigenvalues and the singular values of A. Plot both, compute the rank of A. What kind of matrix do you have to work with?
4. Convert A into `Matlab` sparse format. How much memory is saved (in terms of matrix entries)?
5. Use sparse SVD (`svds`) to compute the largest and the smallest singular values of A. Try `svds(A,q)` for $q = 0.1, 0.01, 0.00001, 0$ and compare with what you have obtained when computing all singular values of A. Try to explain the effect which you see for $q = 0$?

6. Check the possible permutation methods available in `Matlab`, apply them to V and recompute the Cholesky factor for each of them (`>> lookfor permutation`) Which is the best reordering strategy for the matrix V in terms of least fill-in?

7. Consider the matrices A, V (defined as above with $\sigma = 0.01$), Make sure that the resulting matrix is still sparse.

   Use the algorithm you suggested to compute the entries of HI and GL. Think about the computational complexity, keep in mind that those computations have to be embedded in a nonlinear solver, which requires the matrices to be recomputed at each nonlinear iteration. (Note, one particular X and y are to be downloaded from the Lab directory.)

8. It is required to try to do at least parts of the exercise in $\mathbb{R}$:

   (a) Load the package `spam`.

   (b) Browse the user manual for the package (available at the web-page).

   (c) Import the matrix X20 from `20.dat` and create the matrix V as above.

   (d) Compute the Cholesky factor and plot it with the tools available in `spam`. Attach the plot to the report.

   Keywords: `as.spam, chol.spam, diag.spam, display`

```
set.seed(13)
nz=128
ln=nz^2
smat=spam(0,ln,ln)
is = sample(ln,nz)
js = sample(ln,nz)
system.time(for (i in 1:nz) smat[is[i], js[i]] <- i)
system.time(smat[cbind(is,js)] <- 1:nz)
par(mfcol=c(1,1),pty='s')
display(smat,cex=100)
```

Exercise 4 (One more IBD matrix, thanks to Xia Chen)
Load `80.dat` Is it a sparse matrix? What kind of matrix is that (symmetric, full-rank, positive definite, ...) Can you apply the algorithm you suggested in Exercise 2 to handle this matrix?

Exercise 5 (Experience with CGLS)
The exercise is in Matlab.
Download the files `cgls.m`, `hb_file_read.m`, `well1850.dat` and `well1850_rhs.mtx`. The two `.mtx` files contain a matrix A and a right hand side vector b from the 'Matrix-market' collection. We aim at solving $Ax = b$ using the CGLS method. The function `hb_file_read.m` will load the data in Matlab.

Tasks:

- Solve the LS problem using SVD or QR (or by `y=A`). Keep track of the computing time and check the size of the relative residual.

- Solve the LS problem using CGLS and try several stopping criteria for the method. Observe the number of iterations as a function of the stopping criterion.
- Compare the QR/SVD solution and the iterative solution. What is your conclusion regarding accuracy and computing time?

Download a larger sparse LS matrix as

```
load EternityII_A.mat
AA=Problem.A';
bb=rand(size(AA,1),1);
```

Does the iterative method win? In what sense?