Numerical linear algebra

Matrix multiplication

$$\begin{pmatrix} \\ C \\ {\scriptstyle (m\times n)} \\ \\ \end{pmatrix} = \begin{pmatrix} \\ A \\ {\scriptstyle (m\times p)} \\ \\ \end{pmatrix} \begin{pmatrix} \\ \\ B \\ {\scriptstyle (p\times n)} \\ \\ \\ \end{pmatrix}$$

$$c_{ij} = \sum_k a_{ik} b_{kj}$$

```
# C <- 0
for(i in 1:m)
  for(j in 1:n)
    for(k in 1:p)
      C[i,j] <- C[i,j] + A[i,k] * B[k,j]
```

The loops may be placed in any order. Using different loop orders may have a considerable effect on computation time, according to how the matrices are stored in memory.

Take advantage of matrix multiplication

Example

Consider an $n \times p$ matrix X containing 1's and 0's. We wish to calculate the $n \times n$ symmetric matrix D where

$$
\begin{aligned}
d_{ij} &= \text{prop mismatches betw } i\text{th and } j\text{th rows of } X \\
&= \text{ave}_k \; 1\{x_{ik} \neq x_{jk}\} \\
&= \sum_k [x_{ik}(1 - x_{jk}) + (1 - x_{ik})x_{jk}]/p
\end{aligned}
$$

Let $Y = 1 - X$ (*i.e.*, $y_{ij} = 1 - x_{ij}$).

Then

$$D = (XY' + YX')/p$$

Note: *Symmetric storage* can save space and time

Books

1. GH Golub and CF Van Loan (1996) Matrix computations, 3rd ed. Johns Hopkins University Press. [*Focus on computing; rather technical*]
2. RA Thisted (1988) Elements of statistical computing: Numerical computation. Chapman & Hall. [*Focus on computing for statistics*]
3. GAF Seber (1977) Linear regression analysis. Wiley. [*Good stuff on random vectors; appendix on matrix algebra*]
4. CR Rao (1973) Linear statistical inference and its applications. Wiley. [*Like Seber, but bigger and even more expensive*]
5. DA Harville (1997) Matrix algebra from a statistician's perspective. Springer-Verlag. [*Theory and results, very comprehensive*]
6. RA Horn and CR Johnson (1985) Matrix analysis. Cambridge University Press. [*Very deep matrix theory*]

Motivation: linear regression

Consider the model $y = X\beta + \varepsilon$, where $\varepsilon \sim \mathsf{N}(0, \sigma^2 I)$.
(y is $n \times 1$, X is $n \times p$, and β is $p \times 1$.)

Normal equations: $(X'X)\hat{\beta} = X'y$.

Also, var$(\hat{\beta}) = \sigma^2 (X'X)^{-1}$.

Calculating $\hat{\beta} = (X'X)^{-1}X'y$ is not very good computationally.

Problems:

1. We have to form $X'X$, with $p(p+1)/2$ elements
2. We need the inverse of a symmetric matrix with no special pattern

QR decomposition

Let X be an $n \times p$ matrix.

There exist
an *orthonormal* matrix Q [$Q'Q = I$] and
an *upper triangular* matrix R [$\forall\ i > j,\ R_{ij} = 0$]
such that $X = QR$

Note that Q is $n \times p$ and R is $p \times p$.

Normal equations:

$$
\begin{aligned}
& X'X\hat{\beta} = X'y \\
\iff\quad & (QR)'(QR)\hat{\beta} = (QR)'y \\
\iff\quad & R'Q'QR\hat{\beta} = R'Q'y \\
\iff\quad & R'R\hat{\beta} = R'Q'y \\
\Rightarrow\quad & R\hat{\beta} = Q'y \qquad \text{(if } R \text{ is full rank)}
\end{aligned}
$$

The last line is easy to solve since R is upper triangular. Also, note that $(X'X)^{-1} = (R'R)^{-1} = R^{-1}(R^{-1})'$, and R^{-1} is easy to obtain.

Gram-Schmidt algorithm

The following replaces X with the matrix Q and forms the matrix R.

```
for(j in 1:p) {
    r[j,j] <- sqrt( sum(x[,j]^2) )
    x[,j] <- x[,j] / r[j,j]

    if(j < p) for(k in (j+1):p) {

        r[j,k] <- sum(x[,j] * x[,k])
        x[,k] <- x[,k] - x[,j]*r[j,k]

    }
}
```

Note that if X is not of full rank, one of the columns will be made (very close to) 0. Thus $r_{jj} \approx 0$ and so there will be a divide-by-zero error.

If you apply G-S to the first p columns of the augmented matrix $(X : y)$, the last column will become the residuals $\hat{\varepsilon} = y - \hat{y}$.

QR decomposition: Other points

1. Other orthogonalization methods:
   (a) Householder transformations
   (b) Given's rotations

2. QR updates for stepwise regression
   (see Golub and Van Loan 1996)

3. QR decomposition in **R** : `qr()` returns something other than Q and R

```
qr2 <-
function(x)
{
  qq <- qr(x)
  p <- ncol(x); n <- nrow(x)

  r0 <- matrix(0,p,p)
  r0[row(r0) <= col(r0)] <-
    qq$qr[row(qq$qr) <= col(qq$qr)]
  r0 <- sweep(r0,1,(-1)^(1:p),"*")

  q0 <- qr.qy(qq,diag(1,n)[,1:p])
  q0 <- sweep(q0,2,(-1)^(1:p),"*")

  list(q=q0,r=r0)
}
```

Cholesky decomposition

Let A be a $p \times p$ symmetric matrix.

A is **positive semi-definite** if $\forall$ p-vector v, $v'Av \geq 0$.

A is **positive definite** if

- it's positive semi-definite and
- $v'Av = 0$ iff $v = 0$.

If A is positive semi-definite, $\exists$ an upper triangular matrix D such that $D'D = A$.

We have $a_{ij} = \sum_{k=1}^{p} d_{ki} d_{kj}$. But $d_{rc} = 0$ for $r > c$, so

$$a_{ij} = \sum_{k=1}^{i} d_{ki} d_{kj} = \sum_{k=1}^{i-1} d_{ki} d_{kj} + d_{ii} d_{ij}$$

Thus

$$d_{ii} = \left(a_{ii} - \sum_{k=1}^{i-1} d_{ki} d_{kj} \right)^{1/2}$$

$$d_{ij} = \left(a_{ij} - \sum_{k=1}^{i-1} d_{ki} d_{kj} \right) / d_{ii}$$

Cholesky: the algorithm

```
a[1,] <- a[1,] / sqrt(a[1,1])

for(i in 2:p) {

  a[i,i] <- sqrt( a[i,i] -
               sum(a[1:(i-1),i]^2) )

  if(i < p) for(j in (i+1):p)
      a[i,j] <- ( a[i,j] - sum(a[1:(i-1),i] *
                                a[1:(i-1),j])
                  ) / a[i,i]

}

a[lower.tri(a)] <- 0
```

The algorithm goes bad if A is not positive definite ($a[i,i]$ becomes effectively 0).

Uses of the Cholesky decomposition

1. Simulation of correlated random variables
   (See last Monday.)

2. Regression:

   (a) Let $D'D$ be the Cholesky decomposition of $X'X$.
   (b) The normal equations become $D'D\hat{\beta} = X'y$
   (c) Solve $D'\theta = X'y$ for $\theta = D\hat{\beta}$
   (d) Backsolve $D\hat{\beta} = \theta$ for $\hat{\beta}$

3. Determinant of a symmetric matrix, $A = D'D$

$$\det(A) = \det(D'D) = (\det\ D)^2 = \prod_i d_{ii}^2$$

Sweep operator

Consider the "sums of squares and cross-products" (SSCP) matrix

$$A = \begin{pmatrix} X'X & X'y \\ y'X & y'y \end{pmatrix}$$

where X is $n \times p$ and y is $p \times 1$.

Application of the *Sweep operator* to columns 1 through p of A results in the matrix

$$\tilde{A} = \begin{pmatrix} -(X'X)^{-1} & \hat{\beta} \\ \hat{\beta}' & \text{RSS} \end{pmatrix}$$

If you apply the Sweep operator to columns $i_1, \ldots, i_k$, you'll receive the results from regressing y on $X_{i_1}, \ldots, X_{i_k}$ —the corresponding elements in the last column will be the estimated regression coefficients, the $(p+1, p+1)$ element will contain RSS, and so forth.

The Sweep operator has a simple inverse; the two together make it very easy to do stepwise regression. The SSCP matrix is symmetric, and any application of Sweep or its inverse result in a symmetric matrix, so one may take advantage of *symmetric storage*.

Sweep algorithm

```
sweep <-
function(A, k)
{
  n <- nrow(A)

  for(i in (1:n)[-k])
    for(j in (1:n)[-k])
      A[i,j] <- A[i,j] - A[i,k]*A[k,j]/A[k,k]

  # sweep if A[k,k] > 0
  # inverse if A[k,k] < 0
  A[-k,k] <- A[-k,k] / abs(A[k,k])
  A[k,-k] <- A[k,-k] / abs(A[k,k])

  A[k,k] <- -1/A[k,k]

  A
}
```

Note: Be careful about `A[k,k] < tol`.

Consider regression with an intercept.

$$\text{SSCP} = \begin{pmatrix} n & \sum x_1 \; \sum x_2 \; \ldots \; \sum x_p & \sum y \\ \begin{matrix} \sum x_1 \\ \vdots \\ \sum x_p \end{matrix} & X'X & X'y \\ \sum y & y'X & y'y \end{pmatrix}$$

After sweeping column 1, we get the following (the "corrected" SSCP matrix).

$$\text{CSSCP} = \begin{pmatrix} -1/n & \bar{x}_1 \; \bar{x}_2 \; \ldots \; \bar{x}_p & \bar{y} \\ \begin{matrix} \bar{x}_1 \\ \vdots \\ \bar{x}_p \end{matrix} & (X^\star)'X^\star & (X^\star)'y^\star \\ \bar{y} & (y^\star)'X^\star & (y^\star)'y^\star \end{pmatrix}$$

where $X^\star$ and $y^\star$ are X and y with their columns centered about their means (*eg*, $x^\star_{ij} = x_{ij} - \bar{x}_i$).

Thus the elements of the CSSCP matrix are like

$$\sum_j x_{j1}x_{j2} \quad \longrightarrow \quad \sum_j (x_{j1} - \bar{x}_1)(x_{j2} - \bar{x}_2)$$

Note: You should from the CSSCP matrix directly rather than forming the SSCP matrix and then sweeping the first column.

General least squares

Consider the model $y = X\beta + \varepsilon$
where $\varepsilon \sim \mathsf{N}(0, \sigma^2 V)$ with V known.

Consider the Cholesky decomposition of V, $V = D'D$, and let $S' = (D')^{-1} = (D^{-1})'$.

Let $y^\star = S'y$, $X^\star = S'X$, and $\varepsilon^\star = S'\varepsilon$.

Then we have $y^\star = X^\star\beta + \varepsilon^\star$, with $\varepsilon \sim \mathsf{N}(0, \sigma^2 I)$, and we may proceed as before.

Things are particularly simple in the case
$V = \text{diag}\{v_1, \dots, v_n\}$.

Singular value decomposition

Let X be an $n \times p$ matrix with rank$(X) = k \leq p$.

We can write $X = U\Lambda V'$ where $U'U = I_n$, $V'V = I_p$, and

$$\Lambda = \begin{pmatrix} D \\ 0 \end{pmatrix}$$

with $D = \text{diag}\{\lambda_1, \ldots, \lambda_p\}$ and $\lambda_1 \geq \ldots \geq \lambda_p \geq 0$.

Then $X'X = V\Lambda^2 V'$, and the λ_i are the eigenvalues of $X'X$.

Also, $(X'X)^{-1}X'y = V\Lambda^{-1}U'y$, where

$$\Lambda^{-1} = \begin{pmatrix} D^{-1} & 0 \end{pmatrix}$$

When X is not full rank

Suppose rank$(X) = k < p$. Then $\lambda_{k+1} = \ldots = \lambda_p = 0$, and so Λ (and $X'X$) is not invertible.

Write $\hat{\beta}^\star = V\Lambda^+ U'y$ where

$$\Lambda^+ = \begin{pmatrix} \lambda_1^{-1} & & & \\ & \ddots & & 0 \\ & & \lambda_k^{-1} & \\ & & & \\ & 0 & & 0 \end{pmatrix}$$

We should note here than the normal equations $X'X\hat{\beta} = X'y$ do not have a unique solution.

$\hat{\beta}^\star$ (above) is the solution of the normal equations for which $||\hat{\beta}||^2 = \sum_j \hat{\beta}_j^2$ is minimized.

In the case we're discussing, in which X is not of full rank, β is not estimable. The only estimable contrasts (linear functions of β) are of the form $c'\beta$ such that $||Xc|| = c'X'Xc > 0$.

We say $c'\beta$ is *estimable*, if it has a linear unbiased estimate, say $b'y$.

Principal components

Let x be a random p-vector with $\mathsf{E}\, x = \mu$ and $\mathsf{var}\, x = \Sigma$.

Consider the SVD of Σ, $\Sigma = \Gamma'\Lambda\Gamma$ where $\Gamma'\Gamma = I_p$ and $\Lambda = \mathsf{diag}\{\lambda_1, \ldots \lambda_p\}$.

$z = \Lambda x$ contains the "(population) principal components" of x (linear combinations of x that are uncorrelated):

$$\mathsf{var}\, z = \Gamma\Gamma'\Lambda\Gamma\Gamma' = \Lambda$$

Note that the ith row of Γ (*ie*, the ith column of Γ'; call it γ_i) is the eigenvector of Σ corresponding to the eigenvalue λ_i: $\Sigma\gamma_i = \lambda_i\gamma_i$.

Let X be an $n \times p$ matrix, where the rows are iid draws from the population above. (Assume, for convenience, that $\mu = 0$.)

The estimated covariance matrix is $\hat{\Sigma} = X'X/n = \hat{\Gamma}'\hat{\Lambda}\hat{\Gamma}$.

$Z = X\hat{\Gamma}$ contains the "(sample) principal components" of X.

Note that if $X = UDV'$ is the SVD of X, $\hat{\Gamma} = V$ and $\hat{\Lambda} = D'D/n$.