0.1 Magnitude of Non-diagonal Elements of D With Blended G

Function to determine a set of linearly independent rows of SNP covariates

Animals with these genotypes will be used as the core.

```
In [2]: function findCore(X)
    M = copy(float(X))
    n, p = size(M)
    rows = collect(1:n)
    cols = collect(1:p)
    for i=1:min(n,p)
        selr = i:n
        selc = i:p
        v, indx = findmax(abs(M[selr,selc]))
        row, col = ind2sub(M[selr,selc], indx)
        irow = i-1 + row
        icol = i-1 + col
        rows[i], rows[irow] = rows[irow], rows[i]
        cols[i], cols[icol] = cols[icol], cols[i]
        M[i, :], M[irow, :] = M[irow, :], M[i, :]
        M[:, i], M[:, icol] = M[:, icol], M[:, i]
    end
    if abs(M[i,i]) < 0.000000001
        println("breaking at i = ", i)
        return rows, cols, i-1, M
    end
    M[sel1, selc] -= (M[sel1, i]*M[i, selc])./M[i, i]
end
```

```
Out[2]: findCore (generic function with 1 method)
```

Input SNP covariates

```
In [12]: snpDat = readdlm("snpDat.txt")
```

```
p = size(snpDat,2);
```

Input pedigree, phenotype and breeding values

```
In [4]: data = readdlm("pedDat.txt")
```

```
Out[4]: 7x5 Array{Float64,2}:
    1.0  0.0  0.0  99.25  -0.25
    2.0  0.0  0.0  97.92  -0.94
```
In [5]: pedDat = data[:,1:3]  # pedigree
    y = data[:,4]  # phenotypes
    a = data[:,5]  # breeding values

Determine rank and set of linearly independent rows

In [6]: rows, cols, rank, RE = findCore(snpDat)

Out[6]: ([2,7,1,4,5,6,3], [1,2,3,4], 4, 7x4 Array{Float64,2}:
    -1.0  1.0  0.0  0.0
    0.0  2.0 -1.0  0.0
    0.0  0.0 -1.0  0.0
    0.0  0.0  0.0  1.0
    0.0  0.0  0.0  0.0
    0.0  0.0  0.0  0.0
    0.0  0.0  0.0  0.0)

The following rows are linearly independent:

In [7]: rows[1:rank]

Out[7]: 4-element Array{Int64,1}:
    2
    7
    1
    4

SNP covariates are now reordered such that the first 4 rows are linearly independent

In [8]: M = snpDat[rows,:]

Out[8]: 7x4 Array{Float64,2}:
    -1.0  1.0  0.0  0.0
    1.0  1.0 -1.0  0.0
    0.0  0.0 -1.0  0.0
    -1.0  0.0  0.0  1.0
    0.0  1.0  0.0  1.0
    0.0  1.0 -1.0  0.0
    1.0  0.0 -1.0  0.0

Ap'y Calculations

In [15]: using JWAS.PedModule
    ped = PedModule.mkPed("ped.txt")
    Ainv = PedModule.AIInverse(ped)
    AA = inv(full(Ainv))
    indx = [ped.idMap[dec(i)].seqID for i in 1:7]
    A = AA[indx,indx]
\[ G = 0.95 \times M \times M' / p + 0.05 \times A[\text{rows,rows}] \]
\[ k = 4 \]
\[ Sc = 1:k \]
\[ Sn = (k+1):7 \]
\[ Gcc = G[Sc,Sc] \]
\[ Gnc = G[Sn,Sc] \]
\[ iGcc = \text{inv}(Gcc) \]
\[ P = Gnc \times iGcc \]
\[ Gnn = G[Sn,Sn] \]
\[ P; \]

Finished!

The matrix \( D \) can be written as:

\[ D = G_{nn} - P \times G_{cc} \times P' \]

or as

\[ D = G_{nn} - G_{nc} \times G^{-1}_{nn} \times G_{cn} \]

In [16]: D = Gnn - P*Gcc*P'
round(D,15)

Out[16]: 3x3 Array{Float64,2}:

\[
\begin{array}{ccc}
0.0883634 & 0.000213913 & -0.0204569 \\
0.000213913 & 0.0481462 & 0.0119978 \\
-0.0204569 & 0.0119978 & 0.0679162
\end{array}
\]

Note that \( D \) is not diagonal and has values that are large relative to the diagonals. As the dimension of \( D \) gets larger, the consequence of these non-diagonals will be greater.