Livestock Breeding and Genomics - Solution 9

Peter von Rohr

2023-11-10

Problem 1: Numerator Relationship Matrix

The following pedigree is given

Calf	Sire	Dam
4	1	2
5	3	2
6	4	5

The pedigree can be read from the file

```
## https://charlotte-ngs.github.io/lbgfs2023/data/ped_num_rel_mat.csv
```

Compute the numerator relationship matrix A for the given pedigree. Recall from the course notes that elements of matrix A are computed differently for elements on the diagonal and for off-diagonal elements. In summary, we compute

- diagonal element $(A)_{ii}$ as $(A)_{ii} = 1 + F_i$ where $F_i = 0.5 * (A)_{sd}$ where s and d are parents of i.
- off-diagonal element $(A)_{ki}$ as $(A)_{ki} = 0.5 * [(A)_{ks} + (A)_{kd}]$ where s and d are parents of i

Task

Use two nested loops over the rows and the columns of matrix A to compute all the elements of matrix A using the formulas given above.

Solution

Pedigree is read from the given file

```
tbl_ped <- readr::read_delim(file = s_ped_data,
                             delim = ";",
                             col_types = readr::cols(
                               Calf = readr::col_integer(),
                               Sire = readr::col_integer(),
                               Dam  = readr::col_integer()
                             ))
tbl_ped
```

```
## # A tibble: 3 x 3
##    Calf  Sire   Dam
##   <int> <int> <int>
## 1     4     1     2
## 2     5     3     2
## 3     6     4     5
```

Find animals that appear only as parents, start with sires

```
vec_founder_sire_idx <- sapply(tbl_ped$Sire[!is.na(tbl_ped$Sire)],
                               function(x) !is.element(x, tbl_ped$Calf),
                               USE.NAMES = FALSE)
vec_founder_sire <- unique(tbl_ped$Sire[vec_founder_sire_idx])
vec_founder_sire
```

```
## [1] 1 3
```

The same for the dams

```
vec_founder_dam_idx <- sapply(tbl_ped$Dam[!is.na(tbl_ped$Dam)],
                              function(x) !is.element(x, tbl_ped$Calf),
                              USE.NAMES = FALSE)
vec_founder_dam <- unique(tbl_ped$Dam[vec_founder_dam_idx])
vec_founder_dam
```

```
## [1] 2
```

Combining them into one vector and sorting them

```
vec_founder <- c(vec_founder_sire, vec_founder_dam)
vec_founder <- vec_founder[order(vec_founder)]
vec_founder
```

```
## [1] 1 2 3
```

For each of the founders, a new row is added to the top of the pedigree

```
tbl_ped_numrelmat <- NULL
for (n_founder_idx in vec_founder){
  tbl_ped_numrelmat <- dplyr::bind_rows(tbl_ped_numrelmat,
                                        tibble::tibble(Calf = n_founder_idx,
                                                       Sire = NA,
                                                       Dam = NA))
}
tbl_ped_numrelmat <- dplyr::bind_rows(tbl_ped_numrelmat, tbl_ped)
n_ani_ped <- nrow(tbl_ped_numrelmat)
tbl_ped_numrelmat
```

```
## # A tibble: 6 x 3
##    Calf  Sire   Dam
##   <int> <int> <int>
```

```
## 1     1    NA    NA
## 2     2    NA    NA
## 3     3    NA    NA
## 4     4     1     2
## 5     5     3     2
## 6     6     4     5
```

An empty matrix A is initialized

```
mat_A <- matrix(nrow = n_ani_ped, ncol = n_ani_ped)
mat_A
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6]
## [1,]   NA   NA   NA   NA   NA   NA
## [2,]   NA   NA   NA   NA   NA   NA
## [3,]   NA   NA   NA   NA   NA   NA
## [4,]   NA   NA   NA   NA   NA   NA
## [5,]   NA   NA   NA   NA   NA   NA
## [6,]   NA   NA   NA   NA   NA   NA
```

Start the computation with the diagonal element for animal 1

```
i <- 1
s <- tbl_ped_numrelmat$Sire[i]
d <- tbl_ped_numrelmat$Dam[i]
F1 <- ifelse(is.na(s) || is.na(d), 0, 0.5*mat_A[s,d])
mat_A[1,1] <- 1 + F1
mat_A
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6]
## [1,]    1   NA   NA   NA   NA   NA
## [2,]   NA   NA   NA   NA   NA   NA
## [3,]   NA   NA   NA   NA   NA   NA
## [4,]   NA   NA   NA   NA   NA   NA
## [5,]   NA   NA   NA   NA   NA   NA
## [6,]   NA   NA   NA   NA   NA   NA
```

The off-diagonal elements of the first row are computed in using the following loop

```
i <- 1
for (col_idx in 2:n_ani_ped){
  u <- tbl_ped_numrelmat$Sire[col_idx]
  mat_elem_u <- ifelse(is.na(u), 0, mat_A[i,u])
  v <- tbl_ped_numrelmat$Dam[col_idx]
  mat_elem_v <- ifelse(is.na(v), 0, mat_A[i,v])
  mat_A[i,col_idx] <- 0.5 * (mat_elem_u + mat_elem_v)
  mat_A[col_idx,i] <- mat_A[i,col_idx]
}
mat_A
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6]
## [1,] 1.00    0    0  0.5    0 0.25
```

```
## [2,] 0.00   NA   NA   NA   NA   NA
## [3,] 0.00   NA   NA   NA   NA   NA
## [4,] 0.50   NA   NA   NA   NA   NA
## [5,] 0.00   NA   NA   NA   NA   NA
## [6,] 0.25   NA   NA   NA   NA   NA
```

This should now be done for all rows. Hence, we create a second loop that runs form the first row until the last row.

```
for (row_idx in 1:n_ani_ped){
  # diagonal element
  s <- tbl_ped_numrelmat$Sire[row_idx]
  d <- tbl_ped_numrelmat$Dam[row_idx]
  Fi <- ifelse(is.na(s) || is.na(d), 0, 0.5*mat_A[s,d])
  mat_A[row_idx,row_idx] <- 1 + Fi

  # off-diagonal
  if (row_idx < n_ani_ped){
    for (col_idx in (row_idx+1):n_ani_ped){
      u <- tbl_ped_numrelmat$Sire[col_idx]
      mat_elem_u <- ifelse(is.na(u), 0, mat_A[row_idx,u])
      v <- tbl_ped_numrelmat$Dam[col_idx]
      mat_elem_v <- ifelse(is.na(v), 0, mat_A[row_idx,v])
      mat_A[row_idx,col_idx] <- 0.5 * (mat_elem_u + mat_elem_v)
      mat_A[col_idx,row_idx] <- mat_A[row_idx,col_idx]
    }
  }
}
mat_A
```

```
##      [,1] [,2] [,3]  [,4]  [,5]  [,6]
## [1,] 1.00  0.0 0.00 0.500 0.000 0.250
## [2,] 0.00  1.0 0.00 0.500 0.500 0.500
## [3,] 0.00  0.0 1.00 0.000 0.500 0.250
## [4,] 0.50  0.5 0.00 1.000 0.250 0.625
## [5,] 0.00  0.5 0.50 0.250 1.000 0.625
## [6,] 0.25  0.5 0.25 0.625 0.625 1.125
```

Problem 2: Verification

Use the function `pedigreemm::getA()` from package `pedigreemm` to verify your result from problem 1.

Solution

The pedigree is defined by

```
ped <- pedigreemm::pedigree(sire = tbl_ped_numrelmat$Sire,
                            dam = tbl_ped_numrelmat$Dam,
                            label = as.character(tbl_ped_numrelmat$Calf))
```

The numerator relationship matrix is computed by

```
mat_num_relmat <- pedigreemm::getA(ped = ped)
mat_num_relmat
```

```
## 6 x 6 sparse Matrix of class "dsCMatrix"
##      1   2    3     4     5     6
## 1 1.00 .   .    0.500 .     0.250
## 2 .    1.0 .    0.500 0.500 0.500
## 3 .    .   1.00 .     0.500 0.250
## 4 0.50 0.5 .    1.000 0.250 0.625
## 5 .    0.5 0.50 0.250 1.000 0.625
## 6 0.25 0.5 0.25 0.625 0.625 1.125
```

Check whether matrices are identical

```
mat_A - as.matrix(mat_num_relmat)
```

```
##   1 2 3 4 5 6
## 1 0 0 0 0 0 0
## 2 0 0 0 0 0 0
## 3 0 0 0 0 0 0
## 4 0 0 0 0 0 0
## 5 0 0 0 0 0 0
## 6 0 0 0 0 0 0
```

Problem 3: Functions in R

Computations such as the computation of the diagonal elements or such as the one of the off-diagonal elements can be isolated and factored out in a important programming construct which is called `function`. A function takes a set of input parameter and transforms them into a result which is returned. For our example of the numerator relationship matrix two functions can be constructed according to the following template

```
compute_square <- function(pn_number){
  square_result <- pn_number*pn_number
  return(square_result)
}
```

The function can be used by function calls which take a given input and return a result

```
compute_square(pn_number = 3)
```

```
## [1] 9
```

```
compute_square(1:10)
```

```
##  [1]   1   4   9  16  25  36  49  64  81 100
```

Task

Use the above template to construct a function and factor out the computations of the diagonal elements and of the off-diagonal elements into two separate functions.

Solution

The function to compute the diagonal elements can be defined as follows

```
compute_diag_elem <- function(pmat_numrel, ptbl_ped, pn_ani_idx){
   s <- ptbl_ped$Sire[pn_ani_idx]
   d <- ptbl_ped$Dam[pn_ani_idx]
   Fi <- ifelse(is.na(s) || is.na(d), 0, 0.5*pmat_numrel[s,d])
   result_diag_elem <- 1 + Fi
   return(result_diag_elem)
}
```

The function to compute an offdiagonal element is defined below

```
compute_off_diag_elem <- function(pmat_numrel, ptbl_ped, pn_row_idx, pn_col_idx){
  u <- ptbl_ped$Sire[pn_col_idx]
  mat_elem_u <- ifelse(is.na(u), 0, pmat_numrel[pn_row_idx,u])
  v <- ptbl_ped$Dam[pn_col_idx]
  mat_elem_v <- ifelse(is.na(v), 0, pmat_numrel[pn_row_idx,v])
  result_off_diag_elem <- 0.5 * (mat_elem_u + mat_elem_v)
  return(result_off_diag_elem)
}
```

The functions are used to compute all elements of the matrix as shown below.

```
mat_A <- matrix(nrow = n_ani_ped, ncol = n_ani_ped)
for (row_idx in 1:n_ani_ped){
  # compute diagonal element
  mat_A[row_idx, row_idx] <- compute_diag_elem(pmat_numrel = mat_A,
                                               ptbl_ped = tbl_ped_numrelmat,
                                               pn_ani_idx = row_idx)
  # compute off-diagonal elements
  if (row_idx < n_ani_ped){
    for (col_idx in (row_idx+1):n_ani_ped){
      mat_A[row_idx, col_idx] <- compute_off_diag_elem(pmat_numrel = mat_A,
                                                       ptbl_ped = tbl_ped_numrelmat,
                                                       pn_row_idx = row_idx,
                                                       pn_col_idx = col_idx)
      mat_A[col_idx,row_idx] <- mat_A[row_idx, col_idx]
    }
  }
}
mat_A
```

```
##      [,1] [,2] [,3]  [,4]  [,5]  [,6]
## [1,] 1.00  0.0 0.00 0.500 0.000 0.250
## [2,] 0.00  1.0 0.00 0.500 0.500 0.500
## [3,] 0.00  0.0 1.00 0.000 0.500 0.250
## [4,] 0.50  0.5 0.00 1.000 0.250 0.625
## [5,] 0.00  0.5 0.50 0.250 1.000 0.625
## [6,] 0.25  0.5 0.25 0.625 0.625 1.125
```

Checking the results

```r
mat_A <- as.matrix(mat_num_relmat)
```

```
##   1 2 3 4 5 6
## 1 0 0 0 0 0 0
## 2 0 0 0 0 0 0
## 3 0 0 0 0 0 0
## 4 0 0 0 0 0 0
## 5 0 0 0 0 0 0
## 6 0 0 0 0 0 0
```