Applied Statistical Methods - Solution 3

Peter von Rohr

2021-03-15

Problem 1: Fixed Linear Effects Model

We want to analyse a dataset with genetic information using a fixed linear effects model. The dataset is taken from the course notes and is shown in Table 1.

We assume that the SNP loci have a purely additive effect on the trait. That means for a SNP locus L the absolute value of the genotypic value of the homozygous genotypes (L_1L_1 and L_2L_2) is taken to be a_L and the genotypic value of the heterozygous genotype (L_1L_2) is taken to be 0. The fixed linear effects model contains the observation in Table 1 as the response variable, an intercept and the genotypic values of the the genotypes at the two SNP Loci G and H as predictor variables.

For the observation y_i of animal i, we can specify the model as

$$y_i = \beta_0 + W_i \cdot a + \epsilon_i$$

where β_0 is the intercept, a is the vector of additive SNP-effects, W_i is a row vector denoting the SNP-Genotypes and ϵ_i is the random error term.

The data can be read from https://charlotte-ngs.github.io/gelasmss2021/data/ex03p01_data.csv. The address from where the data can be downloaded is assigned to a variable.

```
### # specify path to data file depending on online status
s_data_file <- "https://charlotte-ngs.github.io/gelasmss2021/data/ex03p01_data.csv"
```

It can be read using the following statement

```
### # read the data into a tibble
tbl_geno_data <- readr::read_csv(file = s_data_file)
```

Table 1: Genotypic Data Used for Fitting a Fixed Linear Effect Model

Animal	SNP G	SNP H	Observation
1	1	0	510
2	0	1	528
3	0	1	505
4	1	-1	539
5	1	1	530
6	0	0	489
7	0	-1	486
8	-1	1	485
9	0	-1	478
10	-1	0	479
11	1	0	520

12	1	1	521
13	-1	0	473
14	-1	0	457
15	0	1	497
16	0	0	516
17	1	0	524
18	1	0	502
19	1	-1	508
20	0	0	506

Your Tasks

- Specify the fixed linear effects model in matrix-vector notation by putting the information from the dataset into the model. Use the same parametrization as shown in the course notes where the intercept β_0 and the vector a are combined into a single parameter vector b. The design matrix that links elements in b to observations y is then called X.
- Use the function `Matrix::rankMatrix()` from the `Matrix` package on the matrix X to find out the rank of the design matrix.
- Depending on the rank of X compute an estimate for b, if the rank of the matrix is equal to the number of columns of matrix X, then the same forumla as was used in the regression model can be used
- Verify your results using the `lm()` function

Hints

- Read the data using the function `readr::read_csv()`

Solution

Model Specification: The fixed linear effect model in matrix-vector notation is given by

$$y = X \cdot b + e$$

where the vector b contains all unknown parameters which means, $b = \begin{bmatrix} \beta_0 \\ a \end{bmatrix}$ and X is the design matrix consisting of a column of all ones and with one column for each SNP locus. For our example we have

$$b = \begin{bmatrix} \beta_0 \\ a_G \\ a_H \end{bmatrix}$$

The matrix X comes from the data. The first columns of X is all ones and the second and the third column are codes corresponding to -1, 0 or 1, depending on the genotypes of the animals at the SNP loci G and H.

```
n_nr_animal <-  nrow(tbl_geno_data)
mat_X <- matrix(c(rep(1, n_nr_animal), tbl_geno_data$`SNP G`, tbl_geno_data$`SNP H`), ncol = 3)
```

The matrix X corresponds then to

$$X = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & -1 \\ 1 & 1 & 1 \\ 1 & 0 & 0 \\ 1 & 0 & -1 \\ 1 & -1 & 1 \\ 1 & 0 & -1 \\ 1 & -1 & 0 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \\ 1 & -1 & 0 \\ 1 & -1 & 0 \\ 1 & 0 & 1 \\ 1 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 1 & 0 \\ 1 & 1 & -1 \\ 1 & 0 & 0 \end{bmatrix}$$

The vector y corresponds to

$$y = \begin{bmatrix} 510 \\ 528 \\ 505 \\ 539 \\ 530 \\ 489 \\ 486 \\ 485 \\ 478 \\ 479 \\ 520 \\ 521 \\ 473 \\ 457 \\ 497 \\ 516 \\ 524 \\ 502 \\ 508 \\ 506 \end{bmatrix}$$

Rank of Matrix X: The rank of the matrix X is obtained by

```
Matrix::rankMatrix(mat_X)
```

```
## [1] 3
## attr(,"method")
## [1] "tolNorm2"
## attr(,"useGrad")
## [1] FALSE
## attr(,"tol")
```

```
## [1] 4.440892e-15
```

Solution for $\hat{b}$: Because matrix X has full column rank which means the rank of the matrix is the same as the number of columns, the solution for $\hat{b}$ can be computed the same way as for the regression model. Hence

$$\hat{b} = (X^T X)^{-1} X^T y$$

In R this corresponds to

```
(vec_hatb <- crossprod(solve(crossprod(mat_X)), crossprod(mat_X, vec_y)))
```

```
##            [,1]
## [1,] 497.146104
## [2,]  23.318182
## [3,]   8.402597
```

Verify Results in R:

```
lm_snpflem <- lm(Observation ~ `SNP G` + `SNP H`, data = tbl_geno_data)
summary(lm_snpflem)
```

```
##
## Call:
## lm(formula = Observation ~ `SNP G` + `SNP H`, data = tbl_geno_data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -18.4643  -8.2468  -0.6883   3.9448  26.9383
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  497.146      3.008 165.257  < 2e-16 ***
## `SNP G`       23.318      3.861   6.040 1.33e-05 ***
## `SNP H`        8.403      4.127   2.036   0.0577 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 12.8 on 17 degrees of freedom
## Multiple R-squared:  0.691,  Adjusted R-squared:  0.6546
## F-statistic: 19.01 on 2 and 17 DF,  p-value: 4.621e-05
```

Problem 2: Genomic Relationship Matrix

From the given dataset that can be obtained from

https://charlotte-ngs.github.io/gelasmss2021/data/ex03p02_data.csv,

compute the genomic relationship matrix G. The dataset is organised such that animals are in rows and SNPs are in columns.

Hints

- Read the data using the function `readr::read_csv()`
- Convert the input data with the function `as.matrix()` to a matrix
- Use the function `apply(mat, 2, mean)` to compute the columnwise mean of matrix `mat`
- Use the `matrix` function to construct the matrix P from the vector of SNP allele frequencies

Solution

Start by assigning the address of the input file to a variable.

```
s_data_ex03p02 <- "https://charlotte-ngs.github.io/gelasmss2021/data/ex03p02_data.csv"
```

Now the data can be read from the file.

```
tbl_grm_data <- readr::read_csv(file = s_data_ex03p02)
n_nr_animal <- nrow(tbl_grm_data)
n_nr_snp <- ncol(tbl_grm_data)
```

The frequency of the positive alleles for all SNP positions is computed as

```
mat_W <- as.matrix(tbl_grm_data)
vec_allele_freq <- apply(mat_W+1, 2, mean)/2
```

The sum of $p_j q_j$ is computed as

```
sumpq <- sum(vec_allele_freq * (1-vec_allele_freq))
```

The matrix U is computed from the matrix W and the matrix P

```
mat_P <- matrix(2*vec_allele_freq-1, nrow = n_nr_animal, ncol = n_nr_snp, byrow = TRUE)
mat_U <- mat_W - mat_P
```

The genomic relationship matrix is obtained by

```
mat_grm <- tcrossprod(mat_U) / (2*sumpq)
```

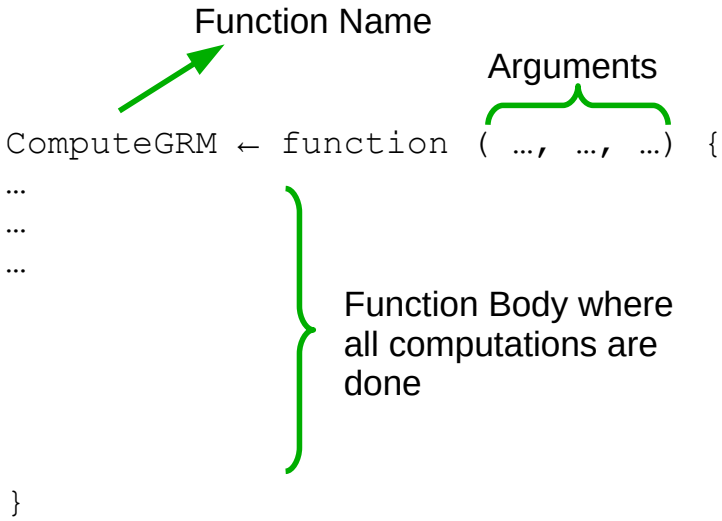
Additional Problem

Write a function in R that accepts a matrix of genotypes and that computes the genomic relationship matrix. Verify your results from Problem 2.

Hints

- A function in R can be declared using the keyword `function`
- A function in R consists of three parts
  1. the name of the function
  2. the function arguments and
  3. the body of the function.

The following figure shows the general structure of an R-function



Solution

The following function computes the genomic relationship matrix

```
#' Compute genomic relationship matrix based on data matrix
computeMatGrm <- function(pmatData) {
  matData <- pmatData
  # check the coding, if matData is -1, 0, 1 coded, then add 1 to get to 0, 1, 2 coding
  if (min(matData) < 0) matData <- matData + 1
  # Allele frequencies, column vector of P and sum of frequency products
  freq <- apply(matData, 2, mean) / 2
  P <- 2 * (freq - 0.5)
  sumpq <- sum(freq*(1-freq))
  # Changing the coding from (0,1,2) to (-1,0,1) and subtract matrix P
  Z <- matData - 1 - matrix(P, nrow = nrow(matData),
                            ncol = ncol(matData),
                            byrow = TRUE)
  # Z%*%Zt is replaced by tcrossprod(Z)
  return(tcrossprod(Z)/(2*sumpq))
}
```

The function is tested with

```
mat_grm_func <- computeMatGrm(pmatData = mat_W)
all.equal(mat_grm, mat_grm_func)
```

```
## [1] TRUE
```