Livestock Breeding and Genomics - Solution 2

Peter von Rohr

2018-10-05

Problem 1: Matrices in R

In R, matrices are constructed using the function `matrix()`. This function accepts different options. We want to see, how these options work.

Your Task: Construct matrices using the different options to better understand the meaning of the different options.

Parameter `data`

- `data`: Specify the different matrix elements

```
(matA <- matrix(data = c(1:9), nrow = 3, ncol = 3))
```

```
##      [,1] [,2] [,3]
## [1,]    1    4    7
## [2,]    2    5    8
## [3,]    3    6    9
```

- `data`: without specifying the matrix elements

```
(matB <- matrix(nrow = 3, ncol = 3))
```

```
##      [,1] [,2] [,3]
## [1,]   NA   NA   NA
## [2,]   NA   NA   NA
## [3,]   NA   NA   NA
```

- `data`: specifying not all matrix elements

```
(matC <- matrix(data = c(1,2,3), nrow = 3, ncol = 3))
```

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

```
(matC2 <- matrix(data = c(1,2,3,4), nrow = 3, ncol = 3))
```

```
## Warning in matrix(data = c(1, 2, 3, 4), nrow = 3, ncol = 3): data length
## [4] is not a sub-multiple or multiple of the number of rows [3]
```

```
##      [,1] [,2] [,3]
## [1,]    1    4    3
## [2,]    2    1    4
## [3,]    3    2    1
```

Parameters `nrow` and `ncol`

- Leaving out one of both parameters

```
(matD <- matrix(data = c(1:9), nrow = 3))
```

```
##      [,1] [,2] [,3]
## [1,]    1    4    7
## [2,]    2    5    8
## [3,]    3    6    9
```

```
(matE <- matrix(data = c(1:9), ncol = 3))
```

```
##      [,1] [,2] [,3]
## [1,]    1    4    7
## [2,]    2    5    8
## [3,]    3    6    9
```

Parameter `byrow`

```
(matF <- matrix(data = c(1:9), nrow = 3, ncol = 3, byrow = TRUE))
```

```
##      [,1] [,2] [,3]
## [1,]    1    2    3
## [2,]    4    5    6
## [3,]    7    8    9
```

```
(matG <- matrix(data = c(1:9), nrow = 3, ncol = 3, byrow = FALSE))
```

```
##      [,1] [,2] [,3]
## [1,]    1    4    7
## [2,]    2    5    8
## [3,]    3    6    9
```

Problem 2: Matrix multiplication in R

In R, matrices can be multiplied using the operator `%*%` or with the functions `crossprod()` or `tcrossprod()`. With `crossprod()` and `tcrossprod()` vectors and matrices can be multiplied directly. The conversion of vectors to matrices is done automatically inside of these functions. The result will always be a matrix. When doing matrix-vector multiplications with `%*%` the vector has to be converted first into a matrix using the function `as.matrix()`.

In a first part of this problem, compare the results of the functions `crossprod()`, `tcrossprod()` and `%*%`.

a) Given are the following matrices

```
matA <- matrix(data = c(1:9), ncol = 3)
matB <- matrix(data = c(2:10), ncol = 3)
```

Find out which matrix multiplication with `%*%` corresponds to the following statement?

```
crossprod(matA,matB)
```

```
##      [,1] [,2] [,3]
## [1,]   20   38   56
## [2,]   47   92  137
## [3,]   74  146  218
```

Solution

The statement `crossprod(matA,matB)` corresponds to

```
t(matA) %*% matB
```

```
##      [,1] [,2] [,3]
## [1,]   20   38   56
## [2,]   47   92  137
## [3,]   74  146  218
```

Alternatively there is the function `tcrossprod()`. Find out which matrix multiplication is executed by

```
tcrossprod(matA, matB)
```

```
##      [,1] [,2] [,3]
## [1,]   78   90  102
## [2,]   93  108  123
## [3,]  108  126  144
```

Solution

```
matA %*% t(matB)
```

```
##      [,1] [,2] [,3]
## [1,]   78   90  102
## [2,]   93  108  123
## [3,]  108  126  144
```

b) Given is the vector `vecB`

```
vecB <- c(-3,16,1)
```

Multiply the matrix `matA` with the vector `vecB` once using `%*%` and then with the function `crossprod()`.

Hint: a vector can be converted to a matrix using the function `as.matrix()`.

Solution

```
matA %*% as.matrix(vecB)
```

```
##      [,1]
## [1,]   68
## [2,]   82
## [3,]   96
```

```
crossprod(t(matA), vecB)
```

```
##      [,1]
## [1,]   68
## [2,]   82
## [3,]   96
```

Problem 3: Quantitative Genetics

In a population the following numbers of genotypes were counted for a given genetic locus called A.

Genotypes	Numbers
A_1A_1	24
A_1A_2	53
A_2A_2	23

a) Compute the genotype frequencies

Solution

```
nTotNrInd <- sum(dfGenotypeFreq$Numbers)
vGenoTypeFreq <- dfGenotypeFreq$Numbers / nTotNrInd
cat(paste("genotype-frequency", dfGenotypeFreq$Genotypes[1]), ": ", vGenoTypeFreq[1])
```

```
## genotype-frequency $A_1A_1$ :  0.24
```

```
cat(paste("genotype-frequency", dfGenotypeFreq$Genotypes[2]), ": ", vGenoTypeFreq[2])
```

```
## genotype-frequency $A_1A_2$ :  0.53
```

```
cat(paste("genotype-frequency", dfGenotypeFreq$Genotypes[3]), ": ", vGenoTypeFreq[3])
```

```
## genotype-frequency $A_2A_2$ :  0.23
```

b) Compute the allele frequencies

Solution

```
vGenFreqP <- vGenoTypeFreq[1] + 0.5*vGenoTypeFreq[2]
vGenFreqQ <-  vGenoTypeFreq[3] + 0.5*vGenoTypeFreq[2]
cat("allele frequency for A1: ", vGenFreqP)
```

```
## allele frequency for A1:  0.505
```

```
cat("allele frequency for A2: ", vGenFreqQ)
```

```
## allele frequency for A2:  0.495
```

c) Compute the population mean μ under the following assumptions

- the difference between the genotypic values of the homozygous genotypes is 20 and
- the genotypic value of the heterozygous genotype is 2.

Solution

```
nDeltaHom <- 20
### # additive value A
nAddValue <- nDeltaHom / 2
nDom <- 2
### # population mean
```

```r
nMu <- (vGenFreqP-vGenFreqQ) * nAddValue + 2 * vGenFreqP * vGenFreqQ * nDom
cat("Population mean: ", nMu, "\n")
```

```
## Population mean:  1.0999
```