Computing Inbreeding and Additive Relationships using the pedigreeTools R-package

gustavoc@msu.edu

09/29/2025

```
knitr::opts_chunk$set(echo = TRUE)
```

The Pedigree Tools R-package (available at Github) can be used to edit, sort and handle complete pedigrees.

Installing pedigreeTools from `GitHub`

```
 remotes:::install_github('https://github.com/Rpedigree/pedigreeTools/')
```

If the above does not work, you can try installing from CRAN

```
 install.packages(pkg='pedigreeTools',type='binary',repos='https://cran.r-project.org')
```

Creating a pedigree object

A pedigree consist of the individual ID as well as the IDs of the father/mother (or sire/dam in animal breeding).

```
library(pedigreeTools)

# pedigree from the midterm.
PED <- data.frame(
                  id=as.character(1:5),
                  sire=as.character(c(NA,1,2,2,2)),
                  dam= as.character(c(NA,NA,NA,3,NA)))

 ped <- with(PED, pedigree(label=id, sire=sire, dam=dam))

 print(ped)
```

```
##   sire  dam
## 1 <NA> <NA>
## 2    1 <NA>
## 3    2 <NA>
## 4    2    3
## 5    2 <NA>
```

All the functions that work on pedigrees require the pedigree to be complete. A complete pedigree is one where each sire and dam also appear as a label. Most often, the pedigrees that we have are not complete. To complete it, and to get the generation number you can use the `editPed()` function. This is illustrated in the following script.

```
PED=PED[-1,] # keeping in the pedigree only the individuals with at least one known parent (this is ty
print(PED)
```

```
##   id sire  dam
## 2  2    1 <NA>
## 3  3    2 <NA>
## 4  4    2    3
## 5  5    2 <NA>
```

```
# This fails because the pedigree is incomplete
# ped <- with(PED, pedigree(label=id, sire=sire, dam=dam))


PED=editPed(label=PED$id,sire=PED$sire,dam=PED$dam)
print(PED)
```

```
##   label sire  dam generation
## 1     1 <NA> <NA>          0
## 2     2    1 <NA>          1
## 3     3    2 <NA>          2
## 5     5    2 <NA>          2
## 4     4    2    3          3
```

```
class(PED)
```

```
## [1] "data.frame"
```

```
# Now we use the complete pedigree to createa a 'pedigree object'

PED=pedigree(label=PED$label, sire=PED$sire, dam=PED$dam)
class(PED)
```

```
## [1] "pedigree"
## attr(,"package")
## [1] "pedigreeTools"
```

Computing inbreeding and additive relationships

```
F=inbreeding(PED)
F
```

```
## [1] 0.00 0.00 0.00 0.00 0.25
```

```
A=as.matrix(getA(PED) ) # A has 1+F in the diagonal, the offdiagonals are 2*coancestry
```

```
## 'as(<dtTMatrix>, "dtCMatrix")' is deprecated.
## Use 'as(., "CsparseMatrix")' instead.
## See help("Deprecated") and help("Matrix-deprecated").
```

```
A # additive relationship matrix
```

```
##       1    2    3     5     4
## 1 1.000 0.50 0.25 0.250 0.375
## 2 0.500 1.00 0.50 0.500 0.750
## 3 0.250 0.50 1.00 0.250 0.750
## 5 0.250 0.50 0.25 1.000 0.375
## 4 0.375 0.75 0.75 0.375 1.250
```

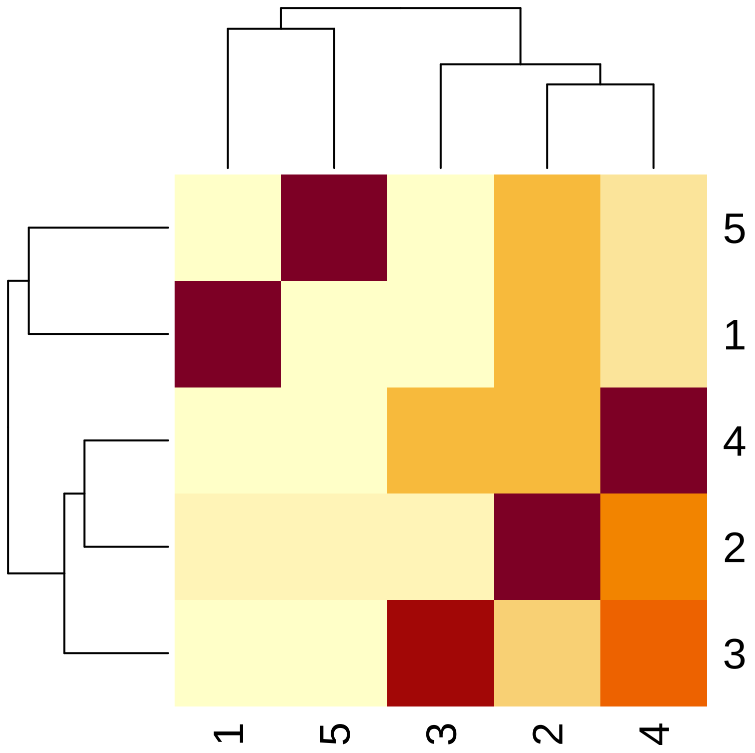
```
A/2 # co-ancestry
```

```
##        1     2     3      5      4
## 1 0.5000 0.250 0.125 0.1250 0.1875
## 2 0.2500 0.500 0.250 0.2500 0.3750
## 3 0.1250 0.250 0.500 0.1250 0.3750
## 5 0.1250 0.250 0.125 0.5000 0.1875
## 4 0.1875 0.375 0.375 0.1875 0.6250
```

```
cbind(round(diag(A)-1,4),F) # Inbreeding
```

```
##           F
## 1 0.00 0.00
## 2 0.00 0.00
## 3 0.00 0.00
## 5 0.00 0.00
## 4 0.25 0.25
```

```
heatmap(A,Rowv = FALSE,Colv=F)
```



5
1
4
2
3
1
5
3
2
4