Cleaning the Polynesian Rat SNP raw data file

Grace Saville

11/02/2022

1. Loading the data

```
data <- read.delim("./data/Raw_data/Genotyping-007.010-01_SNP_Raw_data.tsv")
dim(data) #478 rows (specimens), 333 columns (SNP loci)
```

```
## [1] 478 333
```

```
kable(t(data[1,1:17])) # SNP data in columns 17 to 333
```

	1
island	Borneo_001
registration.number	NBC.LAB.1967
genus	Rattus
species	exulans
sex	female
country	Indonesia
state_province	Kalimantan Timur
island.1	Borneo
locality	Badang, Sungai Kajan
site	
geo_lat	-0.5102
geo_long	117.0912
collector	Victor von Plessen
collecting.date	1935
field.number	AMNH.103838
Populatie	1
X11_CHR1_101004452	?

```
class(data[5,17]) # character
```

```
## [1] "character"
```

```
kable(count(data$island.1)) #  how many samples from each island there are
```

x	freq
	2
Aotea (Great Barrier I)	10
Borneo	25
Doubtful Sound	1
Great Mercury Island	1
Halmahera	25
Hatutaa	21
Honuea	21
Kaikura Island	20
Kamaka	21
Kayangel	21
Late Island	21
Luzon	1
Mainland	3
Malenge	25
Mohotani	14
Motukawanui	21
New Britain	26
New Guinea	25
Normanby Island	25
Rakiura (Stewart Isl)	21
Reiono	21
Rimatuu (Tetiaroa)	21
Slipper Island	21
Sulawesi	25
Tahanea	20
Wake Island	20

```
# data[data$island.1 == "",1:17] # checking why 2 "island.1" cells are blank
kable(data[c(471,473),c(1,3,4,6,8:10)]) # the blanks are from Laos and Cambodia
```

	island	genus	species	country	island.1	locality	site
471	Laos____001	Rattus	exulans	Laos			pak-h0002
473	Cambodia_1	Rattus	exulans	Cambodia			pur-h0010

```
data[471,"island.1"] <- "Mainland" # replacing the blanks with "Mainland"
data[473,"island.1"] <- "Mainland"

x <- data # keeping "data" as backup original
```

2. Tidying SNP order

- I'm doing this to make R evaluation easier (e.g when checking for counts it does not count A:G and G:A separately)

```
dim(x) # 333 cols
```

```
## [1] 478 333
```

```
# count(unlist(x[,17:333]))
x[x == "T:A"] <- "A:T"
x[x == "C:A"] <- "A:C"
x[x == "G:A"] <- "A:G"
x[x == "T:C"] <- "C:T"
x[x == "G:C"] <- "C:G"
x[x == "G:T"] <- "T:G"
kable(count(unlist(x[,17:333]))) # checking success
```

x	freq
?	43641
A:A	17277
A:C	637
A:G	4068
A:T	483
C:C	28637
C:G	1141
C:T	3264
G:G	34313
T:G	488
T:T	17577

3. Removing SNP columns with no variation (invariant/monomorphic)

```
ncol(x) #333
```

```
## [1] 333
```

```
monocols <- integer() # empty vector for the for loop
for (i in 17:333) {
  z <- length(unique(x[, i])) # no. of unique values in the row
  if (z <= 3) {
    monocols <- append(monocols, i) # for TRUE z, add the column number to the vector
  }
  rm(z)
}
# tried with z <= 2 but no result, therefore tried z <= 3
# checked the results manually below

monocols
```

```
##  [1]  17  34  73  80  88  95  98 101 102 108 119 129 139 154 156 171 176 177 178
## [20] 179 194 203 207 208 209 227 237 239 243 251 252 253 265 271 276 324 331
```

```r
for (i in monocols) {
  print(unique(x[,i]))
}
```

```
## [1] "?"   "G:G" "A:A"
## [1] "T:T" "C:T" "C:C"
## [1] "?"   "G:G" "A:A"
## [1] "?"   "A:A" "T:T"
## [1] "A:A" "?"   "G:G"
## [1] "?"   "C:C" "C:T"
## [1] "?"   "G:G" "T:G"
## [1] "?"   "G:G" "A:G"
## [1] "?"   "T:T" "A:A"
## [1] "?"   "A:A" "G:G"
## [1] "?"   "G:G" "A:G"
## [1] "?"   "T:T" "C:T"
## [1] "A:A" "?"   "G:G"
## [1] "?"   "C:C" "C:T"
## [1] "?"   "G:G" "A:A"
## [1] "G:G" "A:G" "?"
## [1] "?"   "C:C" "T:T"
## [1] "?"   "C:C" "A:C"
## [1] "?"   "C:C" "T:T"
## [1] "?"   "C:C" "T:T"
## [1] "A:A" "A:G" "G:G"
## [1] "?"   "G:G" "A:G"
## [1] "T:T" "C:T" "C:C"
## [1] "C:C" "?"   "A:A"
## [1] "?"   "G:G" "A:A"
## [1] "A:A" "A:G" "G:G"
## [1] "?"   "C:C" "A:C"
## [1] "?"   "A:A" "A:C"
## [1] "?"   "T:T" "A:T"
## [1] "?"   "G:G" "A:G"
## [1] "?"   "G:G" "C:G"
## [1] "?"   "G:G" "A:G"
## [1] "?"   "C:C" "T:T"
## [1] "?"   "C:C" "C:T"
## [1] "?"   "G:G" "A:G"
## [1] "C:C" "C:G" "G:G"
## [1] "?"   "C:C" "C:T"
```

```r
# none with only 1 unique SNP in each column ...? It's possible since the SNP loci
# were selected for their differences, but double check this
```

```r
# x <- x[,-c(monocols)] # for removal of monomorphic columns if necessary

rm(i, monocols)
```

4. Removing columns (SNPs) with few samples

```
ncol(x) #333
```

```
## [1] 333
```

```
percblank <- integer() # empty df for the for loop
for (i in 17:333) {
  y <- count(grepl("?", x[,i], fixed = TRUE)) # finds and counts freq of ?
  z <- signif((nrow(x)- y[1,2])/nrow(x)*100, 4) # percentage of ? in the column,
  # to 4 signif digits. I used the number of rows-false outcomes instead of the
  # true outcomes because some rows have no "?"s and result in errors.
    if (z > 60)
      {percblank <- append(percblank, i)
    }
  rm(z)
  rm(y)
}

percblank
```

```
##  [1]  17  18  19  25  48  65  69  73  80  88  89  96 102 108 131 133 146 147 156
## [20] 159 162 165 179 185 205 208 212 228 241 258 264 265 271 304 330
```

```
# checking:
# count(x[,212])
# 320/478
```

```
x <- x[,-c(percblank)] # removing columns listed above, with more than 60% missing data
rm(i, percblank)
```

5. Removing rows (specimens) with few samples

```
x2 <- data.table::transpose(x) # transposing the df temporarily since count()
# doesn't work well on rows

ncol(x2) #478 specimens
```

```
## [1] 478
```

```
percblank <- integer() # empty df for the for loop
for (i in 1:478) {
  y <- count(grepl("?", x2[,i], fixed = TRUE)) # finds and counts freq of "?"
  z <- signif((nrow(x2) - y[1,2]) / nrow(x2) * 100, 4) # percentage of ? in the
  # specimen, to 4 signif digits.
  # I used the no. rows-false outcomes instead of the true outcomes because
  # some rows have no "?" and result in errors.
  if (z > 56)
    # 56 percent missing allowed because it gives 90% completeness (see below)
  {
    percblank <- append(percblank, i)
  }
  rm(z)
  rm(y)
}

percblank
```

```
##  [1]   1   7   9  10  11  18  25  48  49  50  51  52  53  55  56  57  58  59  60
## [20]  62  63  66  71  79  80  87  88  89  90  91  92  93  95  96  97 101 102 103
## [39] 104 105 106 108 109 110 111 112 113 114 115 117 118 119 120 121 122 123 124
## [58] 125 128 129 133 134 135 136 137 139 141 142 143 144 145 146 150 151 152 153
## [77] 154 155 156 157 159 160 161 162 163 164 165 166 167 168 169 170 171 172 173
## [96] 174 175 176 335
```

```
# checking work:
# count(x2[17:298,171])
# 185/298
```

```
x <- x[-c(percblank),] # removing the rows that have too many "?" from the df
nrow(x) #379
```

```
## [1] 379
```

6. Saving

```
# checking the % of all "?"s in the df:
z <- count(grepl("?", unlist(x), fixed = TRUE))
signif(z[2,2]/(z[1,2]+z[2,2])*100, 4) # 9.723% "?"
```

```
## [1] 9.723
```

```
100 - 9.723 # 90.277% complete df, ideal point where there is more than 90%
```

```
## [1] 90.277
```

```
# completeness but not too many rows and columns removed (yet)

rm(i, percblank, x2, z)
# getwd()
# save(list=ls(all=TRUE), file=".RData") # save RDATA for later use if necessary
# write.csv(x, "./data/RStudio/ratsSNPs_halfclean.csv", row.names = FALSE)
```