# Practical 1: Loading genetic data into R, simulating genetic data, allelic frequencies

Jerome Goudet and Bruce Weir

2021-07-14

## Importing data into R

1. download the first 20 megabytes of chromosome 22 (this is a subset of chromosome 22, data from all chromosomes can be downloaded from the 1000 genome download site. From a unix command prompt, enter `wget url` ; from a browser, just download the file.
2. Import this file in R using package `hierfstat` (check the packages vignettes, import ) and its function `read.VCF` using the following commands. :

```
library(hierfstat)
ch22<-read.VCF("chr22_Mb0_20.recode.vcf.gz")
```

```
## ped stats and snps stats have been set.
## 'p' has been set.
## 'mu' and 'sigma' have been set.
```

How many individuals in the data set ? How many SNPs ? Explore the structure of the `ch22` object with `str(ch22)`. Describe the different components

for answers, see section 2.3 of gaston vignette

```
#ch22 matrix with nrows inds and ncol snps
dim(ch22)
```

```
## [1]   2504 103240
```

```
str(ch22)
```

```
## Formal class 'bed.matrix' [package "gaston"] with 8 slots
##   ..@ ped                 :'data.frame': 2504 obs. of  30 variables:
##   .. ..$ famid      : chr [1:2504] "HG00096" "HG00097" "HG00099" "HG00100" ...
##   .. ..$ id         : chr [1:2504] "HG00096" "HG00097" "HG00099" "HG00100" ...
##   .. ..$ father     : num [1:2504] 0 0 0 0 0 0 0 0 0 0 ...
##   .. ..$ mother     : num [1:2504] 0 0 0 0 0 0 0 0 0 0 ...
##   .. ..$ sex        : num [1:2504] 0 0 0 0 0 0 0 0 0 0 ...
##   .. ..$ pheno      : logi [1:2504] NA NA NA NA NA NA ...
##   .. ..$ N0         : int [1:2504] 98074 97918 98286 98078 98612 98437 98030 98512 98528 98113 ..
##   .. ..$ N1         : int [1:2504] 3254 3459 3021 3608 2891 3266 3614 2751 3039 3484 ...
##   .. ..$ N2         : int [1:2504] 1912 1863 1933 1554 1737 1537 1596 1977 1673 1643 ...
##   .. ..$ NAs        : int [1:2504] 0 0 0 0 0 0 0 0 0 0 ...
##   .. ..$ N0.x       : int [1:2504] 0 0 0 0 0 0 0 0 0 0 ...
##   .. ..$ N1.x       : int [1:2504] 0 0 0 0 0 0 0 0 0 0 ...
##   .. ..$ N2.x       : int [1:2504] 0 0 0 0 0 0 0 0 0 0 ...
```

```
##   .. ..$ NAs.x      : int [1:2504] 0 0 0 0 0 0 0 0 0 0 ...
##   .. ..$ N0.y       : int [1:2504] 0 0 0 0 0 0 0 0 0 0 ...
##   .. ..$ N1.y       : int [1:2504] 0 0 0 0 0 0 0 0 0 0 ...
##   .. ..$ N2.y       : int [1:2504] 0 0 0 0 0 0 0 0 0 0 ...
##   .. ..$ NAs.y      : int [1:2504] 0 0 0 0 0 0 0 0 0 0 ...
##   .. ..$ N0.mt      : int [1:2504] 0 0 0 0 0 0 0 0 0 0 ...
##   .. ..$ N1.mt      : int [1:2504] 0 0 0 0 0 0 0 0 0 0 ...
##   .. ..$ N2.mt      : int [1:2504] 0 0 0 0 0 0 0 0 0 0 ...
##   .. ..$ NAs.mt     : int [1:2504] 0 0 0 0 0 0 0 0 0 0 ...
##   .. ..$ callrate   : num [1:2504] 1 1 1 1 1 1 1 1 1 1 ...
##   .. ..$ hz         : num [1:2504] 0.0315 0.0335 0.0293 0.0349 0.028 ...
##   .. ..$ callrate.x : num [1:2504] NaN NaN NaN NaN NaN NaN NaN NaN NaN NaN ...
##   .. ..$ hz.x       : num [1:2504] NaN NaN NaN NaN NaN NaN NaN NaN NaN NaN ...
##   .. ..$ callrate.y : num [1:2504] NaN NaN NaN NaN NaN NaN NaN NaN NaN NaN ...
##   .. ..$ hz.y       : num [1:2504] NaN NaN NaN NaN NaN NaN NaN NaN NaN NaN ...
##   .. ..$ callrate.mt: num [1:2504] NaN NaN NaN NaN NaN NaN NaN NaN NaN NaN ...
##   .. ..$ hz.mt      : num [1:2504] NaN NaN NaN NaN NaN NaN NaN NaN NaN NaN ...
##   ..@ snps               :'data.frame': 103240 obs. of  19 variables:
##   .. ..$ chr     : int [1:103240] 22 22 22 22 22 22 22 22 22 22 ...
##   .. ..$ id      : chr [1:103240] "rs587697622" "rs587755077" "rs587654921" "rs587712275" ...
##   .. ..$ dist    : num [1:103240] 0 0 0 0 0 0 0 0 0 0 ...
##   .. ..$ pos     : int [1:103240] 16050075 16050115 16050213 16050319 16050527 16050568 16050607
##   .. ..$ A1      : chr [1:103240] "A" "G" "C" "C" ...
##   .. ..$ A2      : chr [1:103240] "G" "A" "T" "T" ...
##   .. ..$ quality : num [1:103240] 100 100 100 100 100 100 100 100 100 100 ...
##   .. ..$ filter  : Factor w/ 1 level "PASS": 1 1 1 1 1 1 1 1 1 1 ...
##   .. ..$ N0      : int [1:103240] 2503 2472 2467 2503 2503 2502 2499 2502 2503 2503 ...
##   .. ..$ N1      : int [1:103240] 1 32 36 1 1 2 5 2 1 1 ...
##   .. ..$ N2      : int [1:103240] 0 0 1 0 0 0 0 0 0 0 ...
##   .. ..$ NAs     : int [1:103240] 0 0 0 0 0 0 0 0 0 0 ...
##   .. ..$ N0.f    : int [1:103240] NA NA NA NA NA NA NA NA NA NA ...
##   .. ..$ N1.f    : int [1:103240] NA NA NA NA NA NA NA NA NA NA ...
##   .. ..$ N2.f    : int [1:103240] NA NA NA NA NA NA NA NA NA NA ...
##   .. ..$ NAs.f   : int [1:103240] NA NA NA NA NA NA NA NA NA NA ...
##   .. ..$ callrate: num [1:103240] 1 1 1 1 1 1 1 1 1 1 ...
##   .. ..$ maf     : num [1:103240] 0.0002 0.00639 0.00759 0.0002 0.0002 ...
##   .. ..$ hz      : num [1:103240] 0.000399 0.01278 0.014377 0.000399 0.000399 ...
##   ..@ bed                :<externalptr>
##   ..@ p                  : num [1:103240] 0.0002 0.00639 0.00759 0.0002 0.0002 ...
##   ..@ mu                 : num [1:103240] 0.000399 0.01278 0.015176 0.000399 0.000399 ...
##   ..@ sigma              : num [1:103240] 0.02 0.112 0.126 0.02 0.02 ...
##   ..@ standardize_p      : logi FALSE
##   ..@ standardize_mu_sigma: logi FALSE
```

```
The key thing to note is that ch22@ped is individual centered
while ch22@snps is snps centered
```

3. [optional] Other ways of importing VCF files into R

— If you want to discover the PLINK program : using this software, create a `bed` file containing all biallelic SNPs on this chromosome, excluding alleles other than `A,T,G,C,a,t,g,c`. For this, assuming your file is names `chr22.vcf.gz` type the following commands :

```
%%plink2 is essential for the --max-alleles option to work
```

```
plink2 --vcf chr22.vcf.gz --make-bed  --snps-only just-acgt
--max-alleles 2 --out chr22.1kg
```

You can explore PLINK website to discover some of its capabilities.

— [optional] import this using `R` and `gaston`, using the following commands from the `R` prompt

```
library(gaston)
chr22<-read.bed.matrix("chr22.1kg")
```

— [optional] using `SNPRelate`, convert the chromosome 22 VCF file into a GDS file, and open this file.

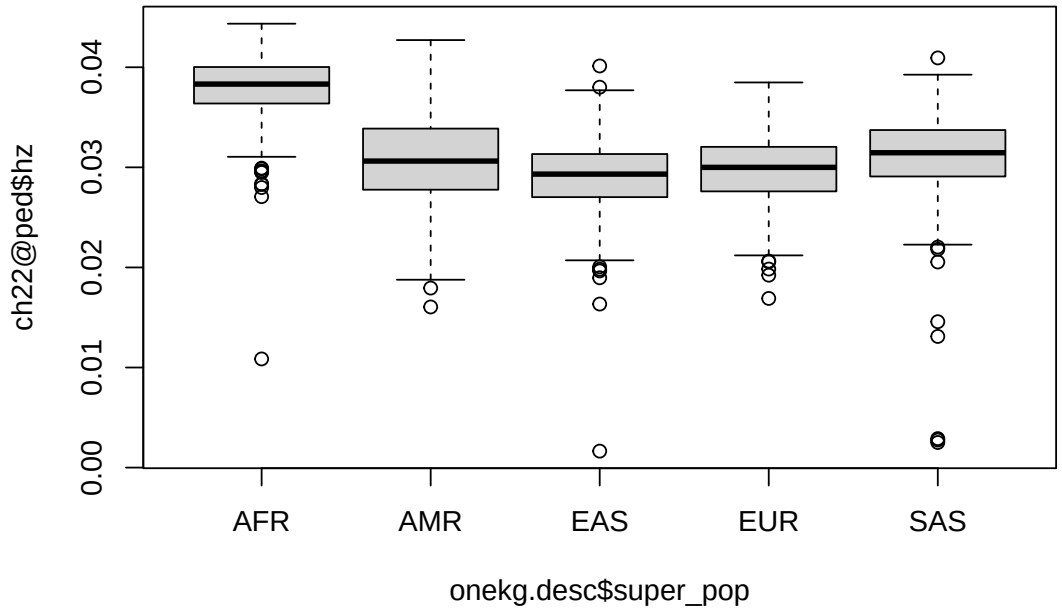
```
library(SNPRelate)
snpgdsVCF2GDS("chr22.vcf.gz","chr22.gds")
f<-snpgdsOpen("chr22.gds")
```

4. Show boxplots of individual heterozygosities as a function of their continent of origin (you will have to load the sample description for this, and make sure that the order of samples is the same in the bed file and the description file). This can be done with the following commands :

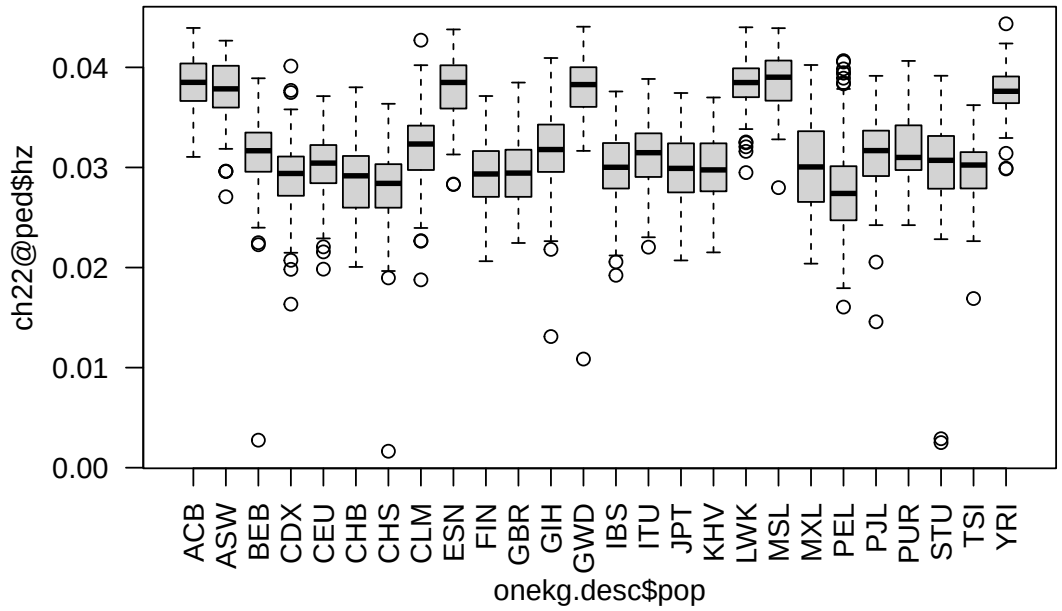
```
samp.desc.fname<-"integrated_call_samples_v3.20130502.ALL.panel"
ftp.path<-"ftp://ftp-trace.ncbi.nih.gov/1000genomes/ftp/release/20130502/"
samp.desc.url<-paste0(ftp.path,samp.desc.fname)
onekg.desc<-read.table(samp.desc.url,header=TRUE,stringsAsFactors = TRUE)
#checks that the order of samples in bed file and description file are the same
all.equal(as.character(ch22@ped$id),as.character(onekg.desc$sample))
```

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

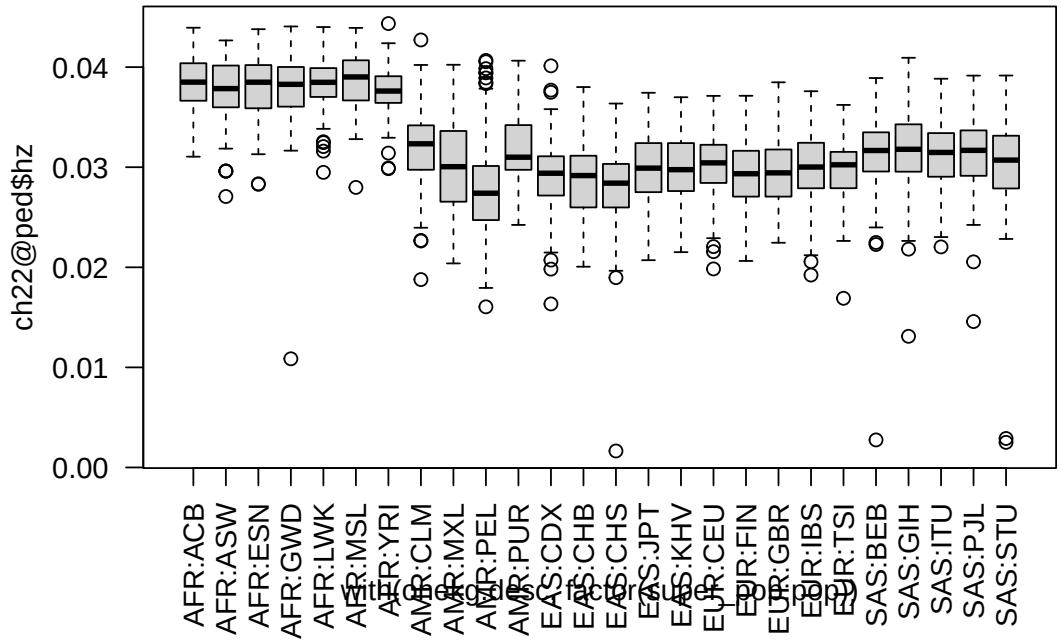
```
boxplot(ch22@ped$hz~onekg.desc$super_pop)
```




```
#and per population
boxplot(ch22@ped$hz~onekg.desc$pop,las=2)
```



```
#same, sorted by continent
boxplot(ch22@ped$hz~with(onekg.desc,factor(super_pop:pop)),las=2)
```




— Which continent show the largest individual heterozygosities ? The lowest ? The most variable ? There is also a slot `ch22@snps$hz` in the `ch22` object. How does it differ from `ch22@ped$hz` ?

```
we clearly see from the boxplots that african genomes are more heterozygous
than genomes from other regions, while East Asian genomes
are least heterozygous. The American genomes
shows the largest variability on heterozygosity
because many individuals from these samples have mixed ancestries and in
particular some african ancestry (more diverse)
while some are closer to native americans, and thus less diverse (less heterozygosity).

ch22@snps$hz contains for each SNP the proportion of individuals
who are heterozygous at this SNP, while ch22@ped@hz contains for each individual
the proportion of SNPs that are heterozygous
```

## Simulating Genetic/Genomic data

5. Simulating genetic data : using `hierfstat` and its function `sim.genot` (read its help page), generate a data set with the genotypes of 50 sampled individuals at 100 biallelic loci from each of 4 populations, where the populations are made of 1000 individuals and exchange migrants at a rate of 0.001. Leave mutation rate and $f$ to their default. From the `R` prompt :

```
library(hierfstat)
#precede the function name with ? to get help, e.g. ?sim.genot
```

```
dat<-sim.genot(nbpop=4,nbloc=100,nbal=2,size=50,N=1000,mig=0.001)
```

— Describe the data set you have just generated. How many rows and columns ? What contains the first column ? the second ? the third ?

```
#dim to get the number of rows and columns
dim(dat)
```

```
## [1] 200 101
```

```
#column names
head(names(dat))
```

```
## [1] "Pop"   "loc.1" "loc.2" "loc.3" "loc.4" "loc.5"
```

```
#structure of the data set

str(dat)
```

```
## 'data.frame':    200 obs. of  101 variables:
##  $ Pop    : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ loc.1  : num  12 12 21 12 12 21 22 11 21 21 ...
##  $ loc.2  : num  11 11 12 11 11 11 11 11 11 11 ...
##  $ loc.3  : num  22 21 11 12 22 12 12 11 21 22 ...
##  $ loc.4  : num  11 11 11 11 12 11 11 11 11 11 ...
##  $ loc.5  : num  22 22 22 22 22 22 22 22 22 22 ...
##  $ loc.6  : num  22 22 22 22 22 22 22 22 22 22 ...
##  $ loc.7  : num  12 11 22 21 12 11 12 22 11 21 ...
##  $ loc.8  : num  21 22 22 22 21 12 22 22 21 12 ...
##  $ loc.9  : num  22 22 21 22 22 22 22 22 22 22 ...
##  $ loc.10 : num  11 11 12 22 12 12 12 22 11 22 ...
##  $ loc.11 : num  11 12 11 11 11 11 11 11 11 11 ...
##  $ loc.12 : num  22 22 22 22 22 22 22 22 22 22 ...
##  $ loc.13 : num  11 11 11 11 11 11 11 11 11 11 ...
##  $ loc.14 : num  11 22 21 21 21 11 12 22 21 22 ...
##  $ loc.15 : num  22 21 11 22 12 22 22 22 12 12 ...
##  $ loc.16 : num  11 11 11 11 12 12 11 11 11 21 ...
##  $ loc.17 : num  12 21 11 11 12 21 11 11 12 11 ...
##  $ loc.18 : num  22 22 22 22 22 22 22 12 22 22 ...
##  $ loc.19 : num  21 11 21 21 22 21 22 21 11 12 ...
##  $ loc.20 : num  11 12 11 11 12 11 11 11 11 12 ...
##  $ loc.21 : num  21 12 12 21 11 12 12 11 21 11 ...
##  $ loc.22 : num  22 22 22 22 22 22 21 22 22 22 ...
##  $ loc.23 : num  22 22 22 11 12 22 21 22 12 22 ...
##  $ loc.24 : num  12 22 21 11 12 22 22 12 22 12 ...
##  $ loc.25 : num  22 22 22 22 22 12 21 22 12 22 ...
##  $ loc.26 : num  22 22 22 12 22 22 22 12 22 22 ...
##  $ loc.27 : num  22 22 21 12 12 22 22 22 21 21 ...
##  $ loc.28 : num  11 11 11 11 11 11 11 11 11 11 ...
##  $ loc.29 : num  21 12 12 12 12 22 22 12 22 22 ...
##  $ loc.30 : num  22 22 22 21 22 22 22 22 22 21 ...
##  $ loc.31 : num  12 21 11 22 22 22 21 22 22 11 ...
##  $ loc.32 : num  11 11 11 11 11 11 11 11 11 11 ...
##  $ loc.33 : num  11 11 11 11 11 11 11 21 12 11 ...
##  $ loc.34 : num  12 22 22 21 22 22 22 22 11 11 ...
```

```
##  $ loc.35 : num  22 12 22 22 22 22 12 22 22 11 ...
##  $ loc.36 : num  22 22 22 22 22 22 12 22 22 22 ...
##  $ loc.37 : num  11 11 11 11 11 11 11 11 12 11 ...
##  $ loc.38 : num  11 21 11 11 11 11 11 11 12 11 ...
##  $ loc.39 : num  11 22 11 11 22 11 12 11 12 12 ...
##  $ loc.40 : num  22 22 22 22 22 22 22 22 22 22 ...
##  $ loc.41 : num  22 12 22 12 12 12 12 22 22 12 ...
##  $ loc.42 : num  21 22 22 22 21 12 22 22 22 11 ...
##  $ loc.43 : num  21 21 21 22 22 22 22 21 22 22 ...
##  $ loc.44 : num  11 11 11 11 11 22 11 11 11 12 ...
##  $ loc.45 : num  22 22 22 12 22 22 21 22 12 22 ...
##  $ loc.46 : num  11 21 12 12 22 21 12 11 22 12 ...
##  $ loc.47 : num  22 21 22 21 11 21 12 21 21 22 ...
##  $ loc.48 : num  22 22 22 22 22 22 22 22 22 22 ...
##  $ loc.49 : num  22 22 21 12 21 21 21 11 12 12 ...
##  $ loc.50 : num  11 11 12 11 11 21 11 21 11 11 ...
##  $ loc.51 : num  11 22 12 11 11 12 11 21 21 11 ...
##  $ loc.52 : num  11 11 11 11 11 11 11 11 11 11 ...
##  $ loc.53 : num  11 11 11 11 11 11 11 11 11 11 ...
##  $ loc.54 : num  11 11 11 11 11 11 11 11 11 11 ...
##  $ loc.55 : num  12 22 11 11 11 12 22 11 11 11 ...
##  $ loc.56 : num  22 22 22 22 22 22 21 22 22 22 ...
##  $ loc.57 : num  21 11 11 12 21 21 12 11 11 12 ...
##  $ loc.58 : num  12 12 11 21 11 12 11 12 22 11 ...
##  $ loc.59 : num  21 22 21 21 11 11 21 22 22 22 ...
##  $ loc.60 : num  12 22 21 21 22 22 22 12 22 22 ...
##  $ loc.61 : num  22 22 21 22 22 22 22 22 22 22 ...
##  $ loc.62 : num  11 11 11 11 11 21 11 11 21 21 ...
##  $ loc.63 : num  22 22 22 22 12 12 22 22 21 11 ...
##  $ loc.64 : num  11 22 12 11 11 12 21 11 11 11 ...
##  $ loc.65 : num  11 21 11 12 11 11 11 21 11 21 ...
##  $ loc.66 : num  22 22 22 12 22 22 22 22 22 22 ...
##  $ loc.67 : num  22 22 12 22 22 22 21 21 12 22 ...
##  $ loc.68 : num  22 22 21 22 12 22 22 22 22 22 ...
##  $ loc.69 : num  11 11 11 11 11 11 11 11 11 11 ...
##  $ loc.70 : num  21 11 11 21 11 11 11 11 11 11 ...
##  $ loc.71 : num  21 21 12 21 11 12 21 11 11 11 ...
##  $ loc.72 : num  22 12 22 22 11 12 22 22 12 22 ...
##  $ loc.73 : num  11 22 22 21 12 12 12 22 21 12 ...
##  $ loc.74 : num  22 11 21 11 11 21 11 21 21 11 ...
##  $ loc.75 : num  22 22 22 22 22 22 22 22 22 22 ...
##  $ loc.76 : num  22 22 22 22 22 22 22 22 22 22 ...
##  $ loc.77 : num  21 21 22 22 22 22 22 12 22 11 ...
##  $ loc.78 : num  21 22 22 11 22 22 22 21 21 11 ...
##  $ loc.79 : num  22 22 22 21 22 22 22 22 22 22 ...
##  $ loc.80 : num  11 21 12 11 11 21 11 11 11 21 ...
##  $ loc.81 : num  21 11 11 11 11 11 11 11 11 11 ...
##  $ loc.82 : num  22 12 11 21 21 12 22 21 12 21 ...
##  $ loc.83 : num  11 11 21 11 22 11 12 11 11 12 ...
##  $ loc.84 : num  22 21 11 11 21 21 11 12 12 11 ...
##  $ loc.85 : num  22 22 22 22 22 22 22 22 22 22 ...
##  $ loc.86 : num  21 22 21 22 11 12 22 12 21 22 ...
##  $ loc.87 : num  11 22 22 22 22 21 21 12 12 22 ...
##  $ loc.88 : num  21 22 22 22 12 22 22 22 22 22 ...
```

```
##  $ loc.89 : num  11 22 12 12 11 21 21 22 22 21 ...
##  $ loc.90 : num  12 22 11 22 11 12 12 22 21 22 ...
##  $ loc.91 : num  11 11 11 11 11 11 11 11 11 11 ...
##  $ loc.92 : num  21 11 11 12 11 11 11 11 12 11 ...
##  $ loc.93 : num  12 11 21 22 22 22 12 12 12 12 ...
##  $ loc.94 : num  21 11 22 21 11 11 11 11 11 11 ...
##  $ loc.95 : num  12 12 12 11 21 22 11 22 21 11 ...
##  $ loc.96 : num  11 11 11 11 11 11 11 11 11 11 ...
##  $ loc.97 : num  11 12 21 11 11 11 11 11 11 21 ...
##  $ loc.98 : num  22 12 21 21 22 22 22 22 22 22 ...
##   [list output truncated]
```

```
First column contains the identifier of the population to which the individual belong
columns 2 and 3 contains the genotypes of individuals at the first and second locus respectively.
The first digit is the first allele (1 or 2), the second digit the second allele
```

— Use the function `hierfstat::biall2dos` to convert `dat` to dosage format :

```
dos<-biall2dos(dat[,-1])
str(dos)
```

```
##  num [1:200, 1:100] 1 1 1 1 1 1 2 0 1 1 ...
##  - attr(*, "dimnames")=List of 2
##   ..$ : NULL
##   ..$ : chr [1:100] "loc.1" "loc.2" "loc.3" "loc.4" ...
```

— Why the `[,-1]` one in the previous command ?

```
The[,-1] is for omitting the first column from teh data frame
as it contains the population identifier rather than a genotype
```

— Could `fstat` format dataset with more than 2 alleles be converted to dosage format ? To answer this, browse `hierfstat` help

```
function `fstat2dos` convert multiallelic markers to dosage data
```

— [optional] explore `hierfstat` help and identify other functions allowing to simulate genotypic data. Describe them briefly

```
sim.genot simulates an island model at equilibrium
sim.genot.t simulates an island model for t generations
sim.genot.metapop.t simulates a metapopulation where the migration pattern
between populations has to be specifide via a matrix
```

6. [optional] Using `ms` or `mspms` (outside `R`, read the manual), simulate a similar dataset (you might want more loci), and import it into R using the function `hierfstat::ms2bed`. To simulate the data, issue at the command prompt (outside `R`) the following command :

```
ms 400 2 -s 50 -r 40 100000 -I 4 100 100 100 100 4 > islM1.txt
```

7. using the file you have just generated or a similar one from the course website (in which case, you'll have to download it first) load it into R :

```
islm1<-ms2bed("islM1.txt")
```

— explore the structure of the `islm1` object. How many individuals ? How many loci ?

```
str(islm1)
```

```
## Formal class 'bed.matrix' [package "gaston"] with 8 slots
##   ..@ ped                 :'data.frame': 200 obs. of  30 variables:
##   .. ..$ famid      : int [1:200] 1 2 3 4 5 6 7 8 9 10 ...
##   .. ..$ id         : int [1:200] 1 2 3 4 5 6 7 8 9 10 ...
##   .. ..$ father     : num [1:200] 0 0 0 0 0 0 0 0 0 0 ...
##   .. ..$ mother     : num [1:200] 0 0 0 0 0 0 0 0 0 0 ...
##   .. ..$ sex        : num [1:200] 0 0 0 0 0 0 0 0 0 0 ...
##   .. ..$ pheno      : num [1:200] 0 0 0 0 0 0 0 0 0 0 ...
##   .. ..$ N0         : int [1:200] 72 76 74 79 74 79 73 77 74 77 ...
##   .. ..$ N1         : int [1:200] 21 18 20 10 20 12 22 17 17 17 ...
##   .. ..$ N2         : int [1:200] 7 6 6 11 6 9 5 6 9 6 ...
##   .. ..$ NAs        : int [1:200] 0 0 0 0 0 0 0 0 0 0 ...
##   .. ..$ N0.x       : int [1:200] 0 0 0 0 0 0 0 0 0 0 ...
##   .. ..$ N1.x       : int [1:200] 0 0 0 0 0 0 0 0 0 0 ...
##   .. ..$ N2.x       : int [1:200] 0 0 0 0 0 0 0 0 0 0 ...
##   .. ..$ NAs.x      : int [1:200] 0 0 0 0 0 0 0 0 0 0 ...
##   .. ..$ N0.y       : int [1:200] 0 0 0 0 0 0 0 0 0 0 ...
##   .. ..$ N1.y       : int [1:200] 0 0 0 0 0 0 0 0 0 0 ...
##   .. ..$ N2.y       : int [1:200] 0 0 0 0 0 0 0 0 0 0 ...
##   .. ..$ NAs.y      : int [1:200] 0 0 0 0 0 0 0 0 0 0 ...
##   .. ..$ N0.mt      : int [1:200] 0 0 0 0 0 0 0 0 0 0 ...
##   .. ..$ N1.mt      : int [1:200] 0 0 0 0 0 0 0 0 0 0 ...
##   .. ..$ N2.mt      : int [1:200] 0 0 0 0 0 0 0 0 0 0 ...
##   .. ..$ NAs.mt     : int [1:200] 0 0 0 0 0 0 0 0 0 0 ...
##   .. ..$ callrate   : num [1:200] NaN NaN NaN NaN NaN NaN NaN NaN NaN NaN ...
##   .. ..$ hz         : num [1:200] Inf Inf Inf Inf Inf ...
##   .. ..$ callrate.x : num [1:200] NaN NaN NaN NaN NaN NaN NaN NaN NaN NaN ...
##   .. ..$ hz.x       : num [1:200] NaN NaN NaN NaN NaN NaN NaN NaN NaN NaN ...
##   .. ..$ callrate.y : num [1:200] NaN NaN NaN NaN NaN NaN NaN NaN NaN NaN ...
##   .. ..$ hz.y       : num [1:200] NaN NaN NaN NaN NaN NaN NaN NaN NaN NaN ...
##   .. ..$ callrate.mt: num [1:200] NaN NaN NaN NaN NaN NaN NaN NaN NaN NaN ...
##   .. ..$ hz.mt      : num [1:200] NaN NaN NaN NaN NaN NaN NaN NaN NaN NaN ...
##   ..@ snps                :'data.frame': 100 obs. of  17 variables:
##   .. ..$ chr     : int [1:100] 1 1 1 1 1 1 1 1 1 1 ...
##   .. ..$ id      : chr [1:100] "M_1" "M_2" "M_3" "M_4" ...
##   .. ..$ dist    : logi [1:100] NA NA NA NA NA NA ...
##   .. ..$ pos     : num [1:100] 3500 8010 8250 11620 19260 ...
##   .. ..$ A1      : logi [1:100] NA NA NA NA NA NA ...
##   .. ..$ A2      : logi [1:100] NA NA NA NA NA NA ...
##   .. ..$ N0      : int [1:100] 199 199 199 198 144 193 115 199 24 26 ...
##   .. ..$ N1      : int [1:100] 1 1 1 2 51 7 60 1 59 60 ...
##   .. ..$ N2      : int [1:100] 0 0 0 0 5 0 25 0 117 114 ...
##   .. ..$ NAs     : int [1:100] 0 0 0 0 0 0 0 0 0 0 ...
##   .. ..$ N0.f    : int [1:100] NA NA NA NA NA NA NA NA NA NA ...
##   .. ..$ N1.f    : int [1:100] NA NA NA NA NA NA NA NA NA NA ...
##   .. ..$ N2.f    : int [1:100] NA NA NA NA NA NA NA NA NA NA ...
##   .. ..$ NAs.f   : int [1:100] NA NA NA NA NA NA NA NA NA NA ...
##   .. ..$ callrate: num [1:100] 1 1 1 1 1 1 1 1 1 1 ...
##   .. ..$ maf     : num [1:100] 0.0025 0.0025 0.0025 0.005 0.1525 ...
##   .. ..$ hz      : num [1:100] 0.005 0.005 0.005 0.01 0.255 0.035 0.3 0.005 0.295 0.3 ...
##   ..@ bed                 :<externalptr>
```

```
##   ..@ p                    : num [1:100] 0.0025 0.0025 0.0025 0.005 0.1525 ...
##   ..@ mu                   : num [1:100] 0.005 0.005 0.005 0.01 0.305 ...
##   ..@ sigma                : num [1:100] 0.0707 0.0707 0.0707 0.0997 0.5131 ...
##   ..@ standardize_p        : logi FALSE
##   ..@ standardize_mu_sigma: logi FALSE
```

```
dim(islm1)
```

```
## [1] 200 100
```

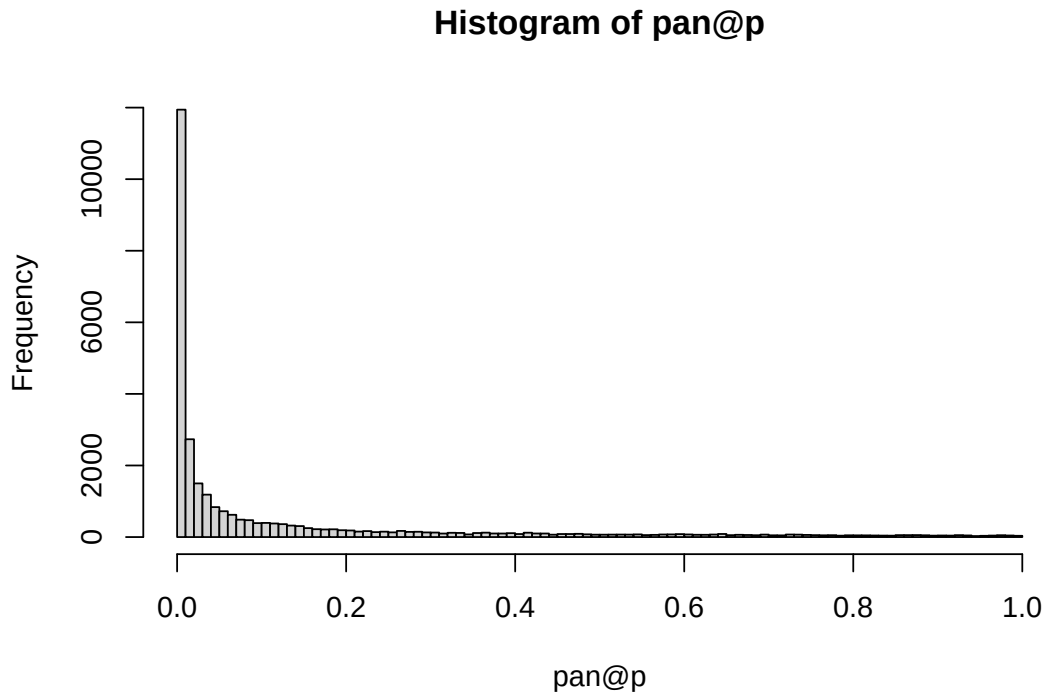
```
200 individuals, 100 loci from the dim command

Note that islm1@ped$famid contains id by default.
famid could be a good place to store the population identifier instead
```

## Allelic frequencies

```
library(gaston)
library(hierfstat)
library(JGTeach)
```

8. Using either data from a panmictic population you have simulated with `ms` or the file pan.txt provided, produce a histogram of allele frequencies

```
pan<-ms2bed("pan.txt")
hist(pan@p,breaks=101)
```




```
L shaped distribution, meaning derived alleles tend to be rare: Mutation model here is
the infinite site model where new mutations arise at places
```

```
where no mutation had occured before. The fate of most mutations is to disappear
but sometimes one is picked up and will dift toward fixation
through the drifting process
```

9. Produce a histogram of SNP allele frequencies of the first 20 megabases of chromosome 22 we imported yesterday, from the African samples, and then from the East Asian samples. Describe the shape of the distribution. How can this be explained ? Compare this distribution to the distribution of allele frequencies from the `ms` simulations we did during practical 1 ; and to the distribution of allelic frequencies generated with `sim.genot`. What can you say about how realistic these distributions are ?

```
ch22<-read.VCF("chr22_Mb0_20.recode.vcf.gz")
```

```
## ped stats and snps stats have been set.
## 'p' has been set.
## 'mu' and 'sigma' have been set.
```

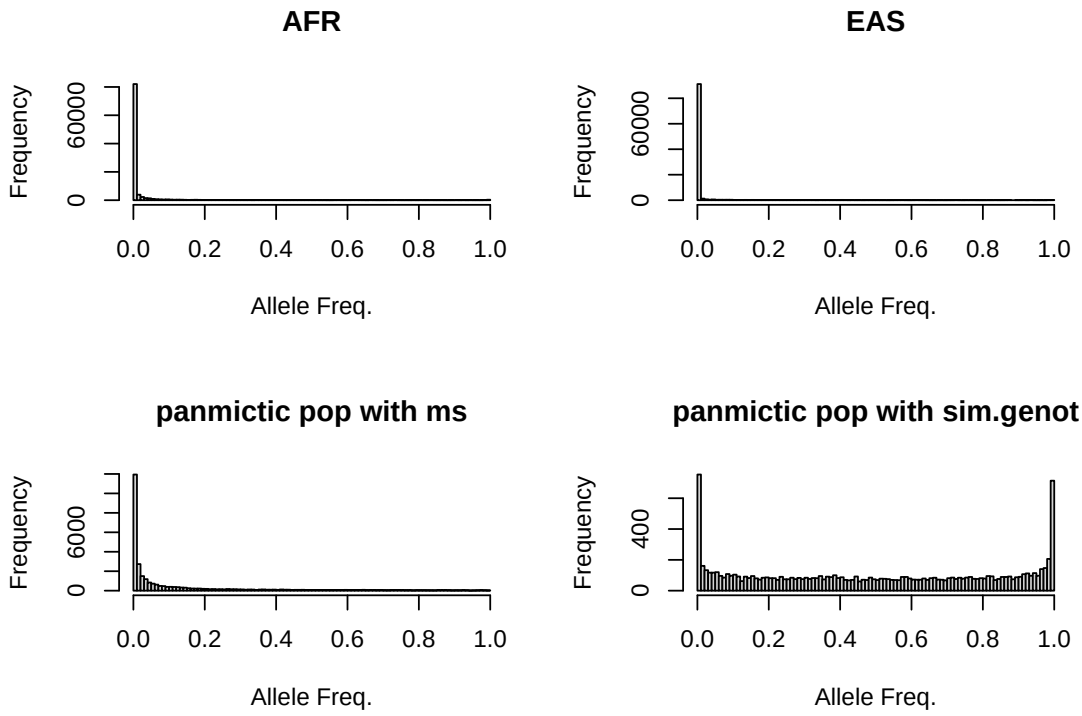
```
samp.desc.file<-"https://www2.unil.ch/popgen/teaching/SISG18/integrated_call_samples_v3.20130502.ALL
samp.desc<-read.table(samp.desc.file,header=TRUE)
names(samp.desc)
```

```
## [1] "sample"    "pop"       "super_pop" "gender"
```

```
str(samp.desc)
```

```
## 'data.frame':    2504 obs. of  4 variables:
##  $ sample   : chr  "HG00096" "HG00097" "HG00099" "HG00100" ...
##  $ pop      : chr  "GBR" "GBR" "GBR" "GBR" ...
##  $ super_pop: chr  "EUR" "EUR" "EUR" "EUR" ...
##  $ gender   : chr  "male" "female" "female" "female" ...
```

```
AFR<-which(samp.desc$super_pop=="AFR")
EAS<-which(samp.desc$super_pop=="EAS")
par(mfrow=c(2,2))
#AFR hist
hist(ch22[AFR,]@p,breaks=101,main="AFR",xlab="Allele Freq.")
#EAS hist
hist(ch22[EAS,]@p,breaks=101,main="EAS",xlab="Allele Freq.")
#PAN ms hist
hist(pan@p,breaks=101,main="panmictic pop with ms",xlab="Allele Freq.")
#simulate data from panmictic pop with sim.genot
dat<-sim.genot(nbal=2,nbpop=1,size=100,nbloc=10000)
#convert to dosage
dos<-biall2dos(dat[,-1])
#colMeans will give twice the frequencies
#->colMean(dos)/2 gives the frequencies
hist(colMeans(dos)/2,breaks=101,main="panmictic pop with sim.genot",xlab="Allele Freq.")
```




```
par(mfrow=c(1,1))
```

```
Shape for EA and AFR samples very similar to that for the panmictic population
obtained with ms. But very different from that obtained from sim.genot,
because the way polymorphism is generated there is different,
it is a KAM mutation model, meaning mutation back and forth, hence the symetric distribution
```

10. using the shiny app BinomNorm (based on function `JGTeach::comp.ci.binom`, compare the accuracy of the Wald, binomial, and bootstrap confidence intervals :

— For a frequency of 0.5 in a sample of 1000 genes.
— For a frequency of 0.1 in a sample of 10 genes.
— For a frequency of 0.1 in a sample of 1′000 genes.

— For a frequency of 0.001 in a sample of 1′000 genes.

Focusing on the Wald and exact confidence intervals, infer a rule for when it is not appropriate to use the Wald confidence interval.

```
The normal approximation works only when np>1 (5) (n is number of alleles sampled,
p the frequency),
so it does not work for 2 and 4, but does for 1 and 3.
The normal approximation should not be used for rare alleles
```

11. Estimate allelic frequency variance from the following genotype counts, where `R` designates the count of reference alleles and `A` the count of alternate alleles. When is the allelic frequency variance the same as binomial variance? Discuss the possible shortcomings of using the binomial variance instead of the true variance.

| $RR$ | $RA$ | $AA$ | $p_R$ | $V$ |
|---|---|---|---|---|
| 100 | 200 | 100 | | |
| 150 | 100 | 150 | | |
| 200 | 0 | 200 | | |
| 4 | 72 | 324 | | |
| 22 | 36 | 342 | | |
| 40 | 0 | 360 | | |

As seen in class the variance from genotypes count is p(-p)(1+f)/2/n
where p is allelic frequency, n number samples and f the inbreeding coefficient
[f=1-Ho/He, where Ho is the observed proportion of het and he=2p(1-p)]

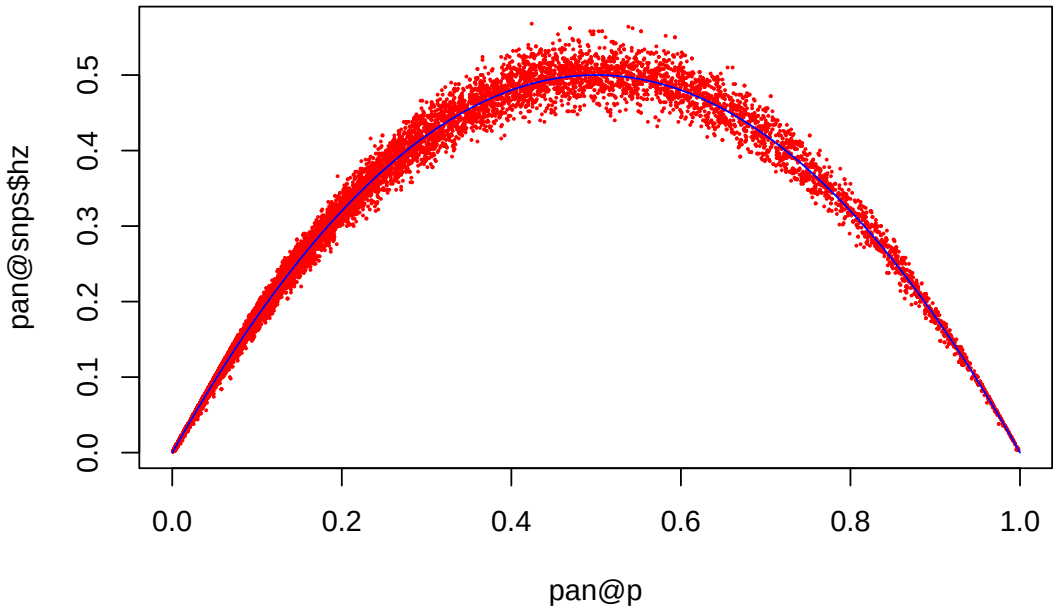
In the table above:
p=0.5 for lines 1-3, 0.1 for lines 4-6
for lines 1 and 4 f=0
for lines 2 and 5, f=0.5
for lines 3 and 6, f=1.0
The following code does the calculation

```
tab<-matrix(c(100,200,100,150,100,150,200,
      0,200,4,72,324,22,36,342,40,0,360),ncol=3,byrow=TRUE)
pr<-(tab[,1]+tab[,2]/2)/400 #freq of reference
f<-1-tab[,2]/400/(2*pr*(1-pr)) # inbreeding coefficient f
v<-pr*(1-pr)*(1+f)/2/400 # variance from genotype counts
round(cbind(tab,pr,f,v),digits=5)
```

```
##                    pr   f       v
## [1,] 100 200 100 0.5 0.0 0.00031
## [2,] 150 100 150 0.5 0.5 0.00047
## [3,] 200   0 200 0.5 1.0 0.00062
## [4,]   4  72 324 0.1 0.0 0.00011
## [5,]  22  36 342 0.1 0.5 0.00017
## [6,]  40   0 360 0.1 1.0 0.00023
```

12. Using either data from a panmictic population you have simulated with `ms` or the file pan.txt provided, plot the SNPs heterozygosity against the frequency of the alternate allele, and add the line of expected heterozygosity.

```
#read pan.txt into a bed object
pan<-ms2bed("https://www2.unil.ch/popgen/teaching/SISGData/pan.txt")
#plot snps hz against p
plot(pan@p,pan@snps$hz,col="red",pch=16,cex=0.3)

#add expected prop of heterozygotes given the frequency
x<-0:1000/1000
lines(x,2*x*(1-x),col="blue")
```

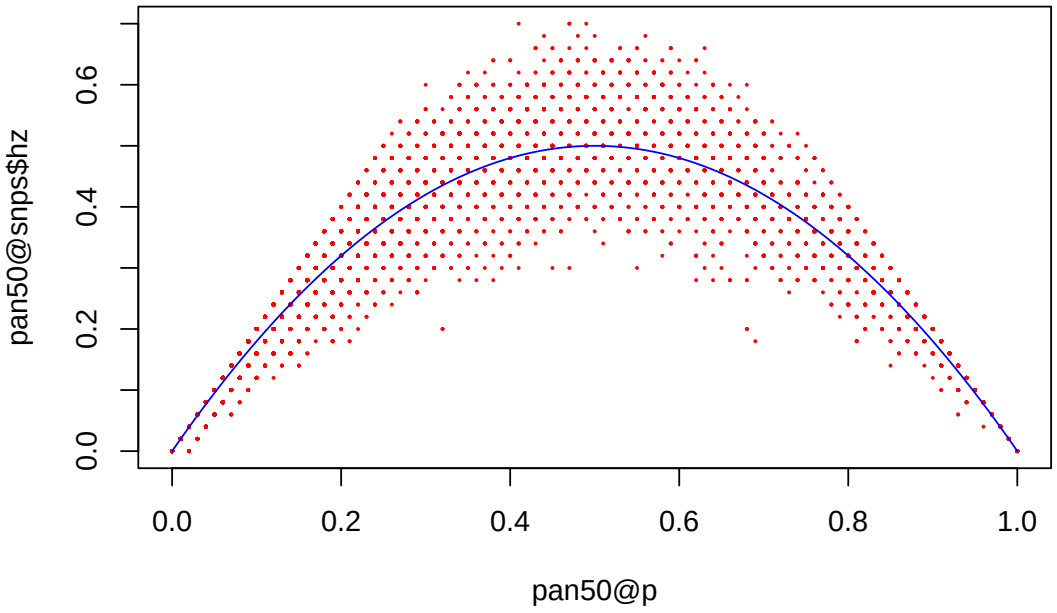



Describe what you see.

```
Nice fit between expectation and data generated under random mating,
this gives an idea of the noise to expect with a sample of 500
```

What would the figure look like if there was only 50 individuals in the sample ?

```
pan50<-pan[1:50,]
plot(pan50@p,pan50@snps$hz,col="red",pch=16,cex=0.3)
lines(x,2*x*(1-x),col="blue")
```




```
# With 50 individuals, the red dots are much more scattered
# than with 500
```