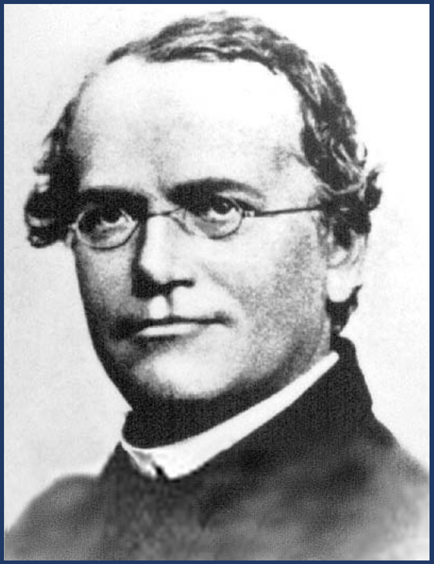
SnpArrays.jl, VCFTools.jl, and ADMIXTURE.jl



OpenMendel Workshop

ASHG Annual Meeting 2020

Hua Zhou (huazhou@ucla.edu)

Motivation

SnpArrays.jl and VCFTools.jl are two gateway packages that

- Translate (potentially big) genotype data files to vectors/matrices for statistical analysis
- Perform common manipulations such as subsetting, summary statistics, filtering, quality control, GRM calculation, PCA calculation, and so on in a **high-level language** environment

I'll demonstrate functionalities of SnpArrays.jl here. Those for VCFTools.jl are similar.

Read in Plink binary format

Given a set of Plink files

```
cd("/Users/huazhou/.julia/dev/SnpArrays/data/")
readdir(glob"mouse.*")
```

```
3-element Array{String,1}:
 "mouse.bed"
 "mouse.bim"
 "mouse.fam"
```

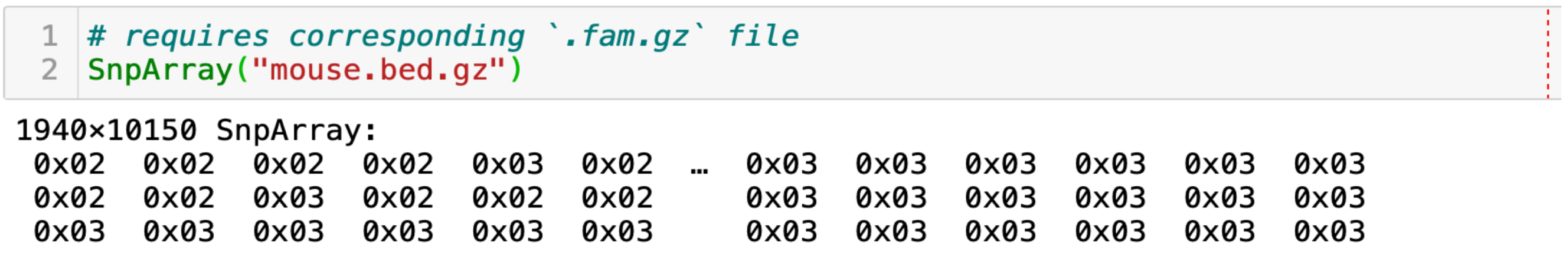
we initialize a SnpArray

```
mouse = SnpArray("mouse.bed")
```

```
1940×10150 SnpArray:
 0x02  0x02  0x02  0x02  0x03  0x02  …  0x03  0x03  0x03  0x03  0x03  0x03
 0x02  0x02  0x03  0x02  0x02  0x02     0x03  0x03  0x03  0x03  0x03  0x03
 0x03  0x03  0x03  0x03  0x03  0x03     0x03  0x03  0x03  0x03  0x03  0x03
 0x02  0x02  0x02  0x02  0x02  0x02     0x03  0x03  0x03  0x03  0x03  0x03
 0x03  0x03  0x03  0x03  0x03  0x03     0x02  0x02  0x02  0x02  0x02  0x02
```

Gzipped Plink files?

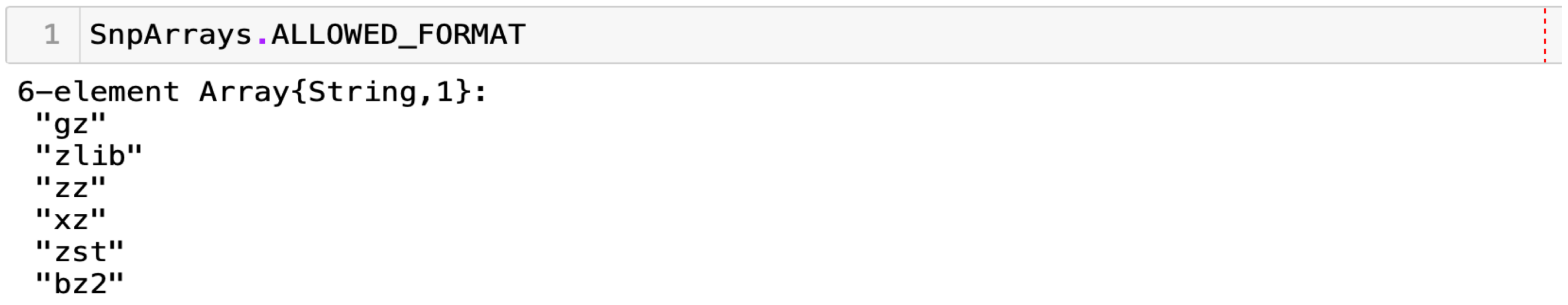
No problem. Use the same command



```
# requires corresponding `.fam.gz` file
SnpArray("mouse.bed.gz")
```

```
1940×10150 SnpArray:
 0x02  0x02  0x02  0x02  0x03  0x02  …  0x03  0x03  0x03  0x03  0x03  0x03
 0x02  0x02  0x03  0x02  0x02  0x02     0x03  0x03  0x03  0x03  0x03  0x03
 0x03  0x03  0x03  0x03  0x03  0x03     0x03  0x03  0x03  0x03  0x03  0x03
```

Other supported compressed formats are



```
SnpArrays.ALLOWED_FORMAT
```

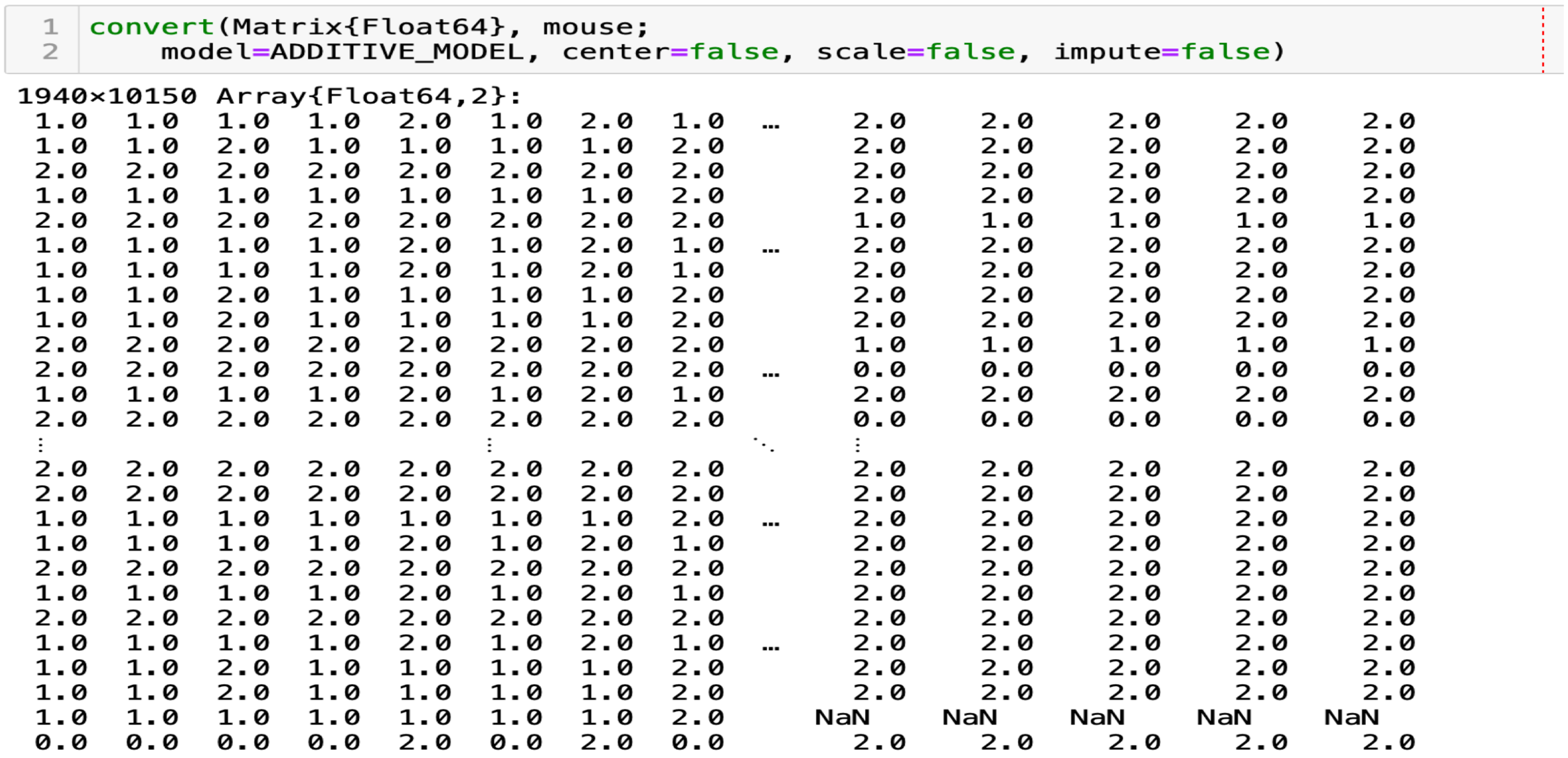
```
6-element Array{String,1}:
 "gz"
 "zlib"
 "zz"
 "xz"
 "zst"
 "bz2"
```

Translate genotypes to numbers

Codebook:

Genotype	`SnpArray`	`model=ADDITIVE_MODEL`	`model=DOMINANT_MODEL`	`model=RECESSIVE_MODEL`
A1,A1	0x00	0	0	0
missing	0x01	NaN	NaN	NaN
A1,A2	0x02	1	1	0
A2,A2	0x03	2	1	1

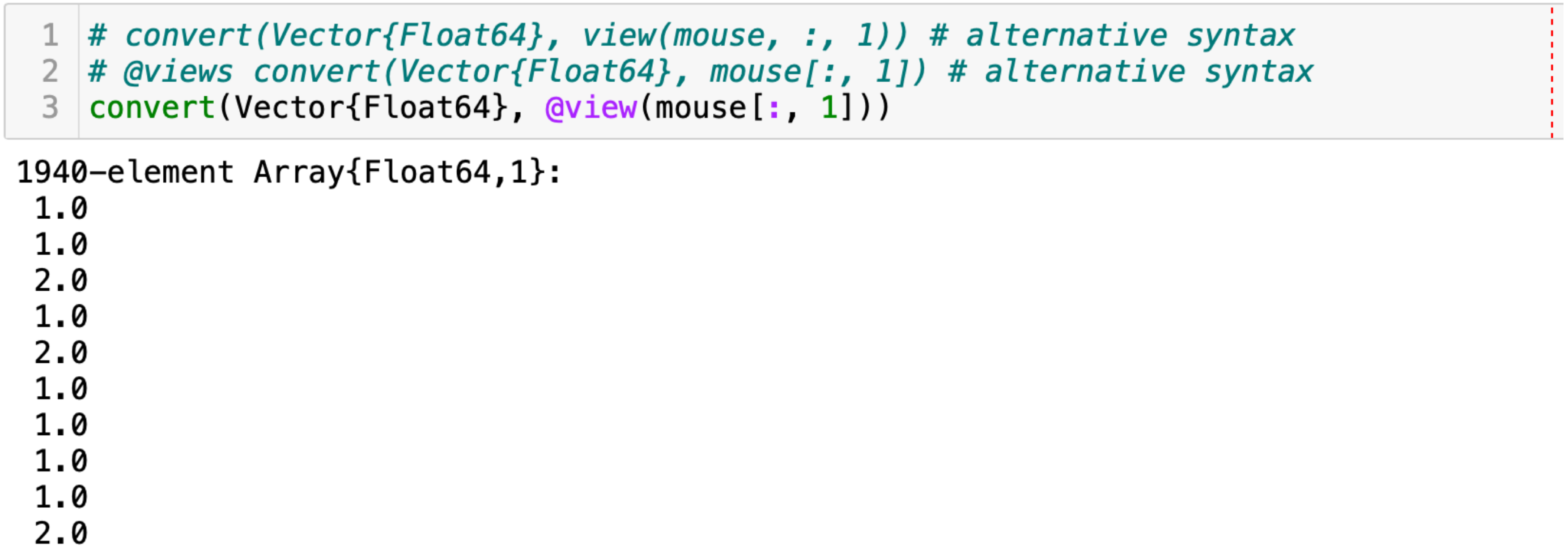
Convert SnpArray to a numeric matrix



```julia
convert(Matrix{Float64}, mouse;
    model=ADDITIVE_MODEL, center=false, scale=false, impute=false)
```

```
1940×10150 Array{Float64,2}:
 1.0  1.0  1.0  1.0  2.0  1.0  2.0  1.0  …    2.0    2.0    2.0    2.0    2.0
 1.0  1.0  2.0  1.0  1.0  1.0  1.0  2.0       2.0    2.0    2.0    2.0    2.0
 2.0  2.0  2.0  2.0  2.0  2.0  2.0  2.0       2.0    2.0    2.0    2.0    2.0
 1.0  1.0  1.0  1.0  1.0  1.0  1.0  2.0       2.0    2.0    2.0    2.0    2.0
 2.0  2.0  2.0  2.0  2.0  2.0  2.0  2.0       1.0    1.0    1.0    1.0    1.0
 1.0  1.0  1.0  1.0  2.0  1.0  2.0  1.0  …    2.0    2.0    2.0    2.0    2.0
 1.0  1.0  1.0  1.0  2.0  1.0  2.0  1.0       2.0    2.0    2.0    2.0    2.0
 1.0  1.0  2.0  1.0  1.0  1.0  1.0  2.0       2.0    2.0    2.0    2.0    2.0
 1.0  1.0  2.0  1.0  1.0  1.0  1.0  2.0       2.0    2.0    2.0    2.0    2.0
 2.0  2.0  2.0  2.0  2.0  2.0  2.0  2.0       1.0    1.0    1.0    1.0    1.0
 2.0  2.0  2.0  2.0  2.0  2.0  2.0  2.0  …    0.0    0.0    0.0    0.0    0.0
 1.0  1.0  1.0  1.0  2.0  1.0  2.0  1.0       2.0    2.0    2.0    2.0    2.0
 2.0  2.0  2.0  2.0  2.0  2.0  2.0  2.0       0.0    0.0    0.0    0.0    0.0
 ⋮                        ⋮                ⋱   ⋮
 2.0  2.0  2.0  2.0  2.0  2.0  2.0  2.0       2.0    2.0    2.0    2.0    2.0
 2.0  2.0  2.0  2.0  2.0  2.0  2.0  2.0       2.0    2.0    2.0    2.0    2.0
 1.0  1.0  1.0  1.0  1.0  1.0  1.0  2.0  …    2.0    2.0    2.0    2.0    2.0
 1.0  1.0  1.0  1.0  2.0  1.0  2.0  1.0       2.0    2.0    2.0    2.0    2.0
 2.0  2.0  2.0  2.0  2.0  2.0  2.0  2.0       2.0    2.0    2.0    2.0    2.0
 1.0  1.0  1.0  1.0  2.0  1.0  2.0  1.0       2.0    2.0    2.0    2.0    2.0
 2.0  2.0  2.0  2.0  2.0  2.0  2.0  2.0       2.0    2.0    2.0    2.0    2.0
 1.0  1.0  1.0  1.0  2.0  1.0  2.0  1.0  …    2.0    2.0    2.0    2.0    2.0
 1.0  1.0  2.0  1.0  1.0  1.0  1.0  2.0       2.0    2.0    2.0    2.0    2.0
 1.0  1.0  2.0  1.0  1.0  1.0  1.0  2.0       2.0    2.0    2.0    2.0    2.0
 1.0  1.0  1.0  1.0  1.0  1.0  1.0  2.0     NaN    NaN    NaN    NaN    NaN
 0.0  0.0  0.0  0.0  2.0  0.0  2.0  0.0       2.0    2.0    2.0    2.0    2.0
```

Convert a column of genotypes

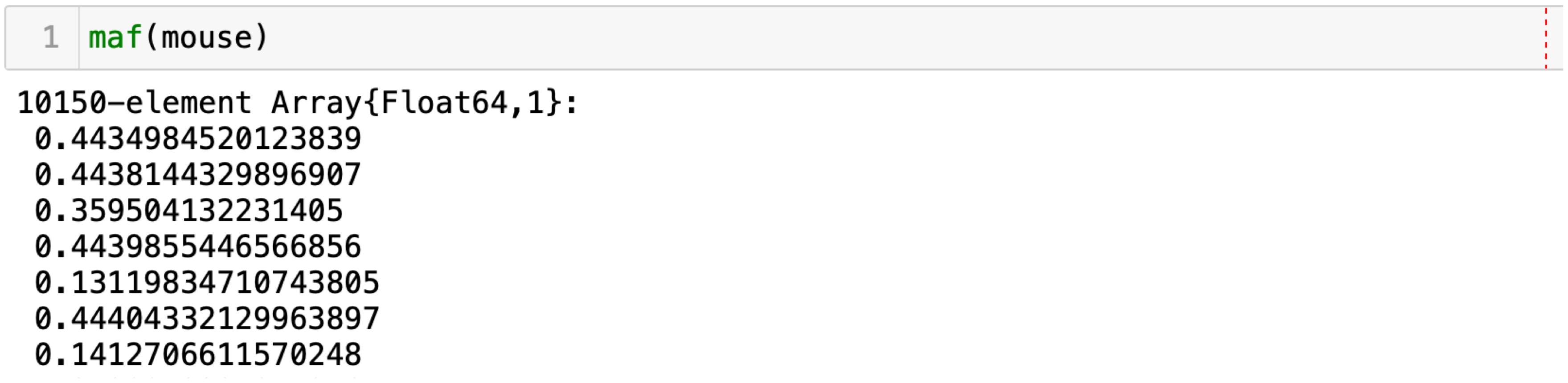


```julia
# convert(Vector{Float64}, view(mouse, :, 1)) # alternative syntax
# @views convert(Vector{Float64}, mouse[:, 1]) # alternative syntax
convert(Vector{Float64}, @view(mouse[:, 1]))
```

```
1940-element Array{Float64,1}:
 1.0
 1.0
 2.0
 1.0
 2.0
 1.0
 1.0
 1.0
 1.0
 2.0
```

Summary statistics

- Minor allele frequencies (MAF)



```
maf(mouse)
```

```
10150-element Array{Float64,1}:
 0.4434984520123839
 0.4438144329896907
 0.359504132231405
 0.4439855446566856
 0.1311983471074380 5
 0.4440433212996389 7
 0.1412706611570248
```

Summary statistics

Missing rate by SNP

```
missingrate(mouse, 1)
```

```
10150-element Array{Float64,1}:
 0.0010309278350515464
 0.0
 0.002061855670103093
```

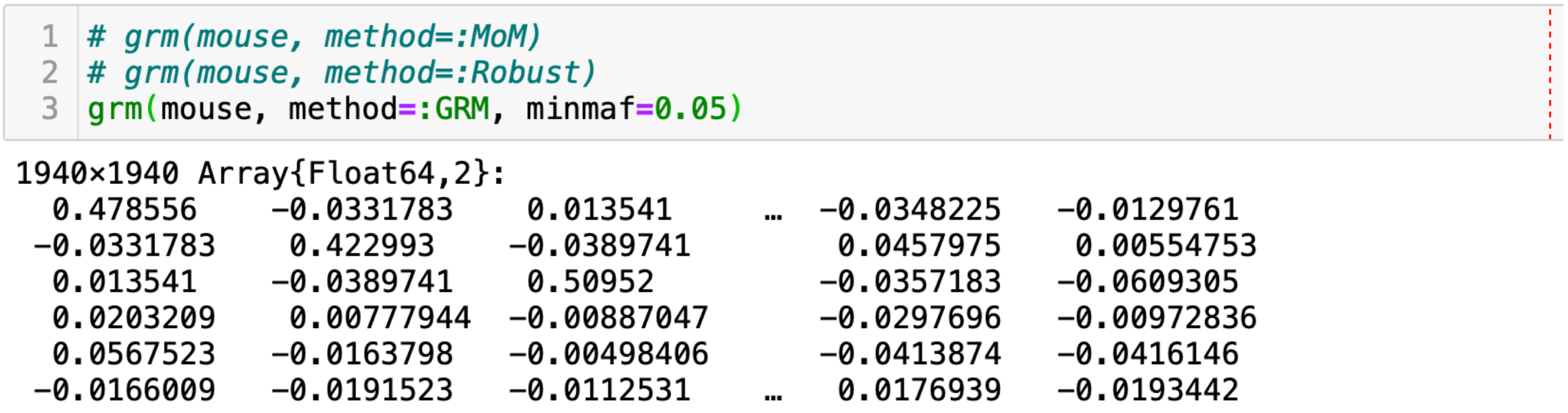
Missing rate by individual

```
missingrate(mouse, 2)
```

```
1940-element Array{Float64,1}:
 0.00019704433497536947
 0.0
 0.018423645320197045
```

Empirical kinship matrix

SnpArrays.jl implements commonly used methods for calculating empirical kinship matrix.



```julia
# grm(mouse, method=:MoM)
# grm(mouse, method=:Robust)
grm(mouse, method=:GRM, minmaf=0.05)
```

```
1940×1940 Array{Float64,2}:
  0.478556    -0.0331783    0.013541     …  -0.0348225   -0.0129761
 -0.0331783    0.422993    -0.0389741        0.0457975    0.00554753
  0.013541    -0.0389741    0.50952         -0.0357183   -0.0609305
  0.0203209    0.00777944  -0.00887047      -0.0297696   -0.00972836
  0.0567523   -0.0163798   -0.00498406      -0.0413874   -0.0416146
 -0.0166009   -0.0191523   -0.0112531    …   0.0176939   -0.0193442
```

Not covered here

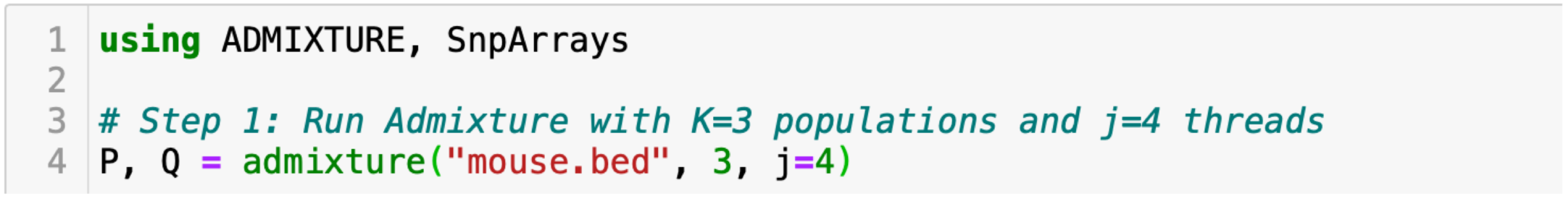
- Filtering
- Linear algebra on SnpArray directly without converting to numerical matrix
- Linear algebra on SnpArray directly on GPU
- Split, concatenate, merge, and reordering of Plink files
- VCF to Plink

Learn from

- Package documentation: https://openmendel.github.io/SnpArrays.jl/latest/
- Workshop tutorial (Binder link): https://mybinder.org/v2/gh/OpenMendel/ASHG-OpenMendelWorkshop-2020-Oct/master?filepath=04-SnpArrays-HZhou%2FSnpArraysTutorial.ipynb

Demo: empirical kinship from admixed samples

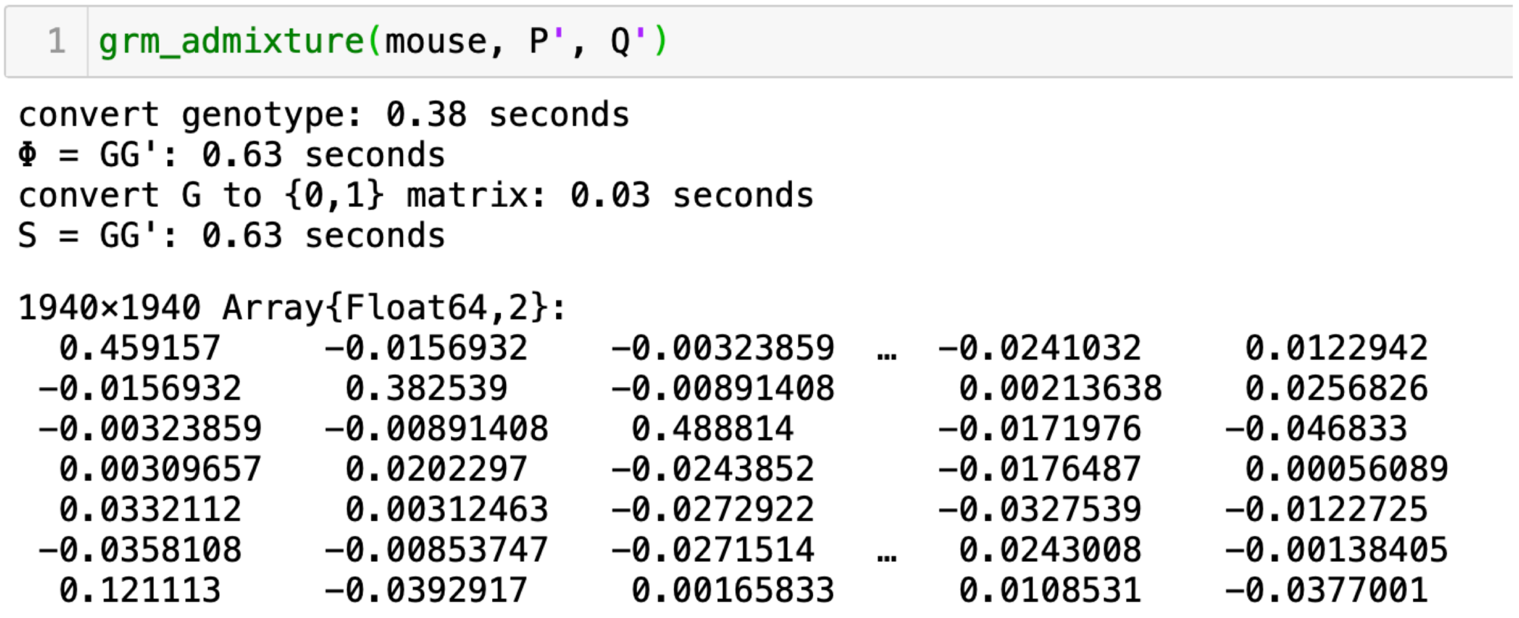
Step 1: Run Admixture program (available on Mac and Linux) to calculate ancestry fractions (per sample) and population specific allele frequencies (per SNP)



```julia
using ADMIXTURE, SnpArrays

# Step 1: Run Admixture with K=3 populations and j=4 threads
P, Q = admixture("mouse.bed", 3, j=4)
```

Demo: empirical kinship from admixed samples

Step 2: Use the P (allele frequencies) and Q (ancestry fractions) matrix from Admixture to compute the kinship coefficients



```
grm_admixture(mouse, P', Q')
```

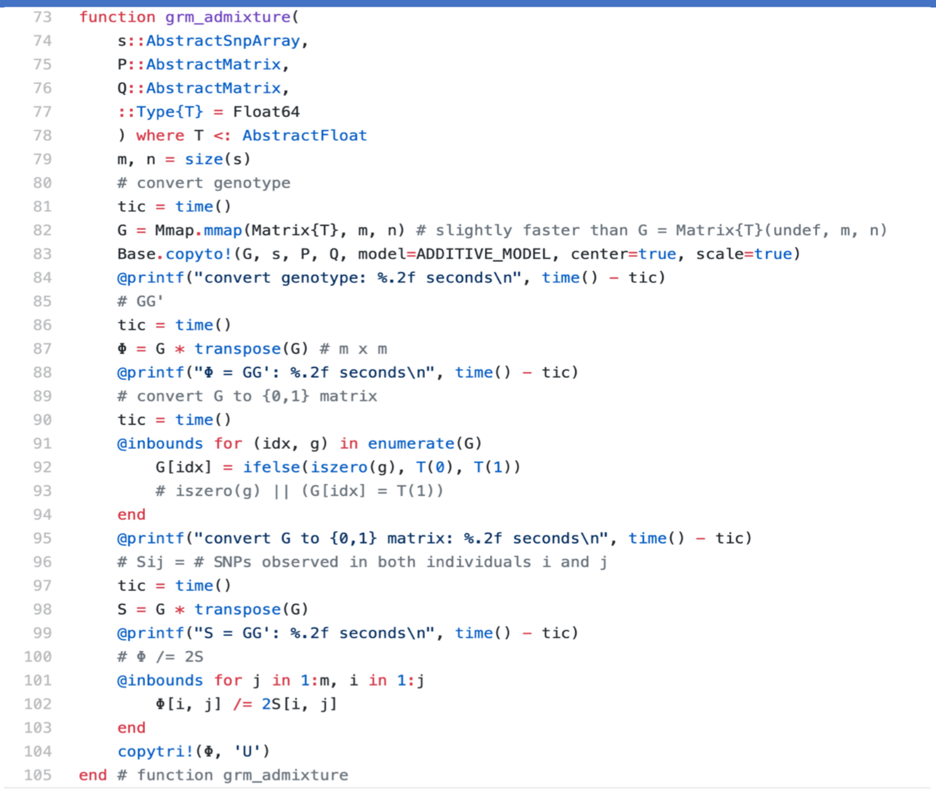
```
convert genotype: 0.38 seconds
Φ = GG': 0.63 seconds
convert G to {0,1} matrix: 0.03 seconds
S = GG': 0.63 seconds

1940×1940 Array{Float64,2}:
  0.459157     -0.0156932    -0.00323859  …  -0.0241032     0.0122942
 -0.0156932     0.382539     -0.00891408      0.00213638    0.0256826
 -0.00323859   -0.00891408    0.488814       -0.0171976    -0.046833
  0.00309657    0.0202297    -0.0243852      -0.0176487     0.00056089
  0.0332112     0.00312463   -0.0272922      -0.0327539    -0.0122725
 -0.0358108    -0.00853747   -0.0271514   …   0.0243008    -0.00138405
  0.121113     -0.0392917     0.00165833      0.0108531    -0.0377001
```

Demo: calculate empirical kinship from admixed samples

Timing results on the ACCORD data with ~6,000 individuals from 4 populations, each with ~800,000 SNPs

- SnpArrays.jl takes about **6 minutes**
- REAP (C Program) takes about **10 hours**

Is that easy to write Julia code?



```julia
function grm_admixture(
    s::AbstractSnpArray,
    P::AbstractMatrix,
    Q::AbstractMatrix,
    ::Type{T} = Float64
    ) where T <: AbstractFloat
    m, n = size(s)
    # convert genotype
    tic = time()
    G = Mmap.mmap(Matrix{T}, m, n) # slightly faster than G = Matrix{T}(undef, m, n)
    Base.copyto!(G, s, P, Q, model=ADDITIVE_MODEL, center=true, scale=true)
    @printf("convert genotype: %.2f seconds\n", time() - tic)
    # GG'
    tic = time()
    Φ = G * transpose(G) # m x m
    @printf("Φ = GG': %.2f seconds\n", time() - tic)
    # convert G to {0,1} matrix
    tic = time()
    @inbounds for (idx, g) in enumerate(G)
        G[idx] = ifelse(iszero(g), T(0), T(1))
        # iszero(g) || (G[idx] = T(1))
    end
    @printf("convert G to {0,1} matrix: %.2f seconds\n", time() - tic)
    # Sij = # SNPs observed in both individuals i and j
    tic = time()
    S = G * transpose(G)
    @printf("S = GG': %.2f seconds\n", time() - tic)
    # Φ /= 2S
    @inbounds for j in 1:m, i in 1:j
        Φ[i, j] /= 2S[i, j]
    end
    copytri!(Φ, 'U')
end # function grm_admixture
```