

```
In [1]: # Founders: real haplotype data (chlto10.200SNP)
# "Sample animals for sire and dam candidates" & "Dams and Sires": dam size = N of dams
# One Dam -> 1 male & 1 female (1 dam -> 2 offsprings)
# selection: increase
# 5 generation selection: increase
# heritability = 0.5
# Phenotypes : all animals in G0 to G4
# Genotypes : all progeny in G5 and all sires in each generation
# Change muAlpha = 0.2
# 10 chromosomes; 20 loci per chromosome = > 200 Loci (50 QTL & 150 Markers)
```

```
In [2]: include("/home/nicole/Jupyter/XSimSel.jl")
```

```
Out[2]: XSim
```

```
In [3]: using DataFrames
```

```
In [4]: using Distributions
```

```
In [5]: using(Gadfly)
```

## Initialize XSim

```
In [6]: numChr      = 10
        chrLength  = 0.01
        numLoci    = 20
        nQTL       = 5
        mutRate    = 0.0
        locusInt   = chrLength/numLoci
        mapPos     = collect(locusInt/2:locusInt:chrLength)
        geneFreq   = fill(0.5,numLoci)
        QTL        = sample(1:numLoci,nQTL,replace=false)
        qtlMarker  = fill(false,numLoci)
        qtlMarker[QTL] = true
        mu = 100                                     #  $\alpha \sim N(100,1)$ 
        Va = nQTL*numChr*0.5*mu*mu                  #  $Va = nQTL * 2pq * \text{mean}(\alpha)^2$ 
        qtlEffects= rand(Normal(mu/sqrt(Va), sqrt(1/Va)),numLoci*numChr) # Let  $\alpha \mu = 0.2$ 
        XSim.init(numChr,numLoci,chrLength,geneFreq,mapPos,qtlMarker,qtlEffects,mutRate)
```

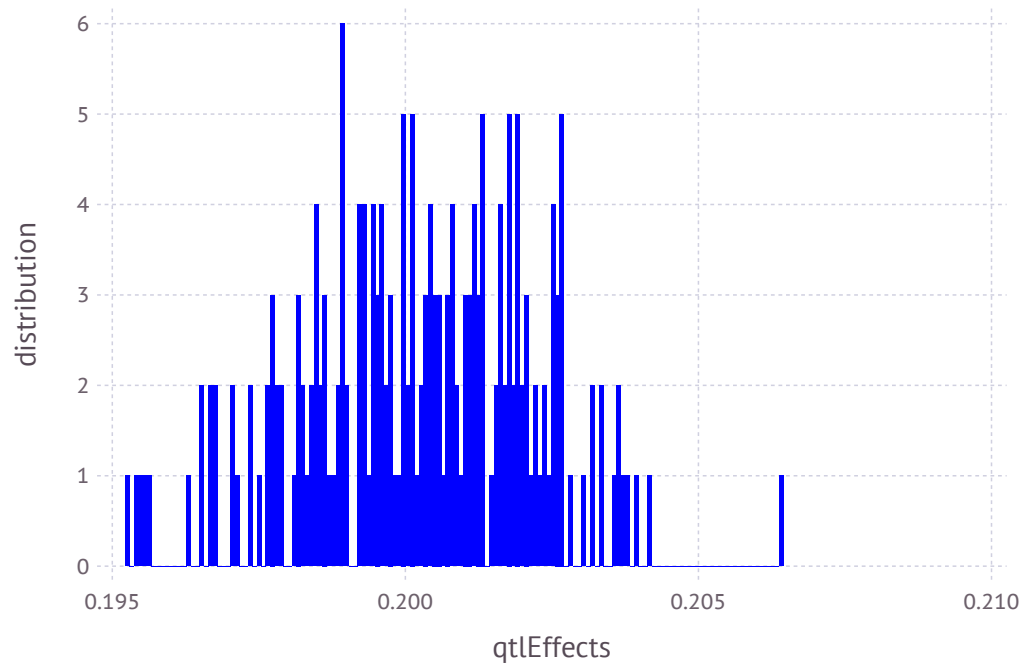
```
In [7]: qtlEffects
```

```
Out[7]: 200-element Array{Float64,1}:  
 0.200743  
 0.201942  
 0.199253  
 0.198258  
 0.201963  
 0.198452  
 0.200052  
 0.201519  
 0.199948  
 0.199  
 0.20118  
 0.201056  
 0.200105  
 ⋮  
 0.201163  
 0.199429  
 0.199978  
 0.20103  
 0.202469  
 0.198936  
 0.201294  
 0.20305  
 0.198952  
 0.196315  
 0.201296  
 0.200039
```

```
In [8]: writedlm("qtlEffects",qtlEffects)
```

```
In [9]: plot(x=qtEffects, Geom.histogram, Guide.XLabel("qtEffects"), Guide.YLabel("distribution"), Theme(default
```

```
Out[9]:
```



```
In [10]: mean(qtEffects)
```

```
Out[10]: 0.20022548823402345
```

```
In [11]: var(qtEffects)
```

```
Out[11]: 3.890880317566901e-6
```

```
In [12]: # Base Population
gen=0
nGenBase    = 5
popSizeBase = 8000

# Sample 20 sire and 400 dams
popSizeSP = 8000

# Purbred Populations - mating
popSize = 8000
nGener  = 5
nSires  = 200
nDams   = 4000
npop    = 1
;
```

```

In [13]: # Animals with genotypes
posOFF0S = 1
posOFF0E = popSizeSP
posOFF1S = posOFF0E + 1
posOFF1E = posOFF0E + popSize
posOFF2S = posOFF1E + 1
posOFF2E = posOFF1E + popSize
posOFF3S = posOFF2E + 1
posOFF3E = posOFF2E + popSize
posOFF4S = posOFF3E + 1
posOFF4E = posOFF3E + popSize
posOFF5S = posOFF4E + 1
posOFF5E = posOFF4E + popSize
println("posOFF1S: ",posOFF1S,"; posOFF1E: ",posOFF1E)
println("posOFF2S: ",posOFF2S,"; posOFF2E: ",posOFF2E)
println("posOFF3S: ",posOFF3S,"; posOFF3E: ",posOFF3E)
println("posOFF4S: ",posOFF4S,"; posOFF4E: ",posOFF4E)
println("posOFF5S: ",posOFF5S,"; posOFF5E: ",posOFF5E)

# FileName :
PedAll = "PedAll.txt"           # pedigree file with all animals
PheAll = "PheAll.txt"          # phenotype file with all animals
GenAll = "GenAll.txt"           # genotype file with all animals

Gen = "Gen.txt"                 # genotype file with all progeny in G5 and all sires in each generation
Phe = "Phe.txt"                 # phenotype file with all animals in G1 to G4
QTL = "QTL.txt"                 # QTL with with all progeny in G5 and all sires in each generation
Mar = "Mar.txt"                 # Marker with with all progeny in G5 and all sires in each generation
GenNF = "GenNF.txt"             # remove fixed genes from genotype file
QTLNF = "QTLNF.txt"             # remove fixed genes from QTL file
MarNF = "MarNF.txt"             # remove fixed genes from Marker file
;

posOFF1S: 8001; posOFF1E: 16000
posOFF2S: 16001; posOFF2E: 24000
posOFF3S: 24001; posOFF3E: 32000
posOFF4S: 32001; posOFF4E: 40000
posOFF5S: 40001; posOFF5E: 48000

```

```
In [14]: fileName = "SS"
pedText = fileName * ".ped"
genText = fileName * ".gen"
pheText = fileName * ".phe"
;
```

```
In [15]: ;rm $pedText $genText $pheText
```

## Sampling Founders

```
In [16]: sires = XSim.sampleFounders("/home/nicole/Jupyter/HERdata/data/Clean/newdata/ch1to10.200SNP.group1")
dams = XSim.sampleFounders("/home/nicole/Jupyter/HERdata/data/Clean/newdata/ch1to10.200SNP.group2");

Sampling 360 animals into base population.
Sampling 361 animals into base population.
```

## Base population

```
In [17]: baseSires, baseDams, baseGen = XSim.sampleRan(popSizeBase, nGenBase, sires, dams, gen=gen);

Generation      1: sampling  4000 males and  4000 females
Generation      2: sampling  4000 males and  4000 females
Generation      3: sampling  4000 males and  4000 females
Generation      4: sampling  4000 males and  4000 females
Generation      5: sampling  4000 males and  4000 females
```

## Sample animals for sire and dam candidates

```
In [18]: popSP = XSim.sampleRan(popSizeSP, 1, baseSires, baseDams, gen=baseGen, fileName=fileName);

Generation      6: sampling  4000 males and  4000 females
```

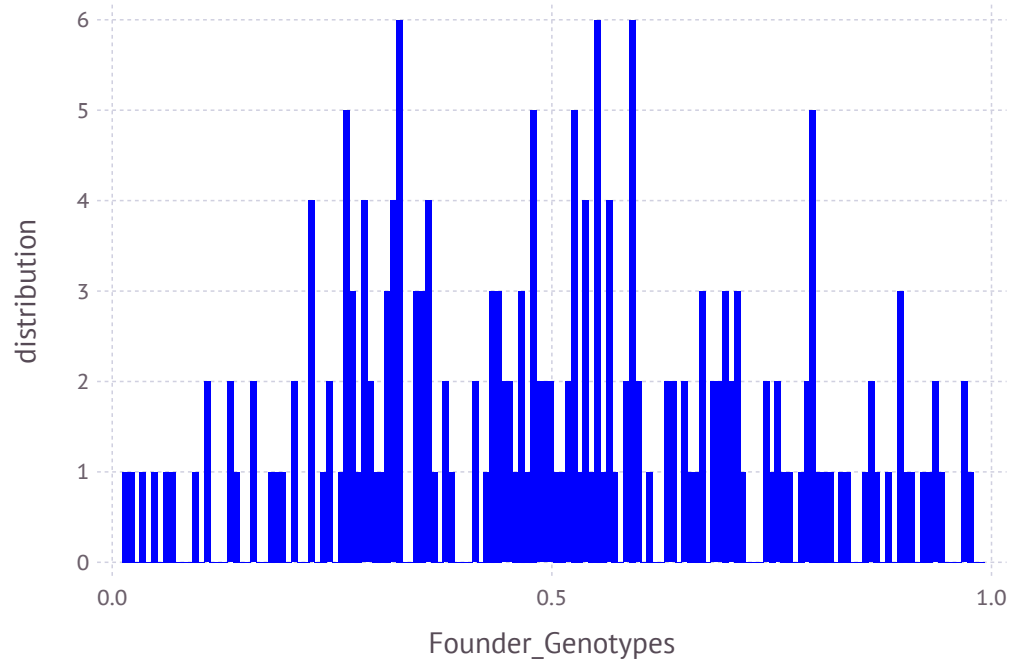
```
In [19]: gSPSire = XSim.getOurGenotypes(popSP[1])
gSPDam = XSim.getOurGenotypes(popSP[2])
gSP = [gSPSire;gSPDam];
```

```
In [20]: FCM = mean(gSP/2,1)
```

```
Out[20]: 1x200 Array{Float64,2}:
 0.0625  0.830125  0.2955  0.94075 ...  0.35975  0.3835  0.896625  0.5495
```

```
In [21]: plot(x=FCM, Geom.histogram, Guide.XLabel("Founder_Genotypes"), Guide.YLabel("distribution"), Theme(default
```

```
Out[21]:
```



```
In [22]: V=var(gSP,1)
Mark=gSP[:,V.>0]
corMat=cor(Mark)
nRows =size(corMat,1)
LDMat =zeros(nRows-1,20);
```

```
In [23]: for i=1:(nRows-20)
LDMat[i,:] = corMat[i,(i+1):(i+20)].^2
end
```

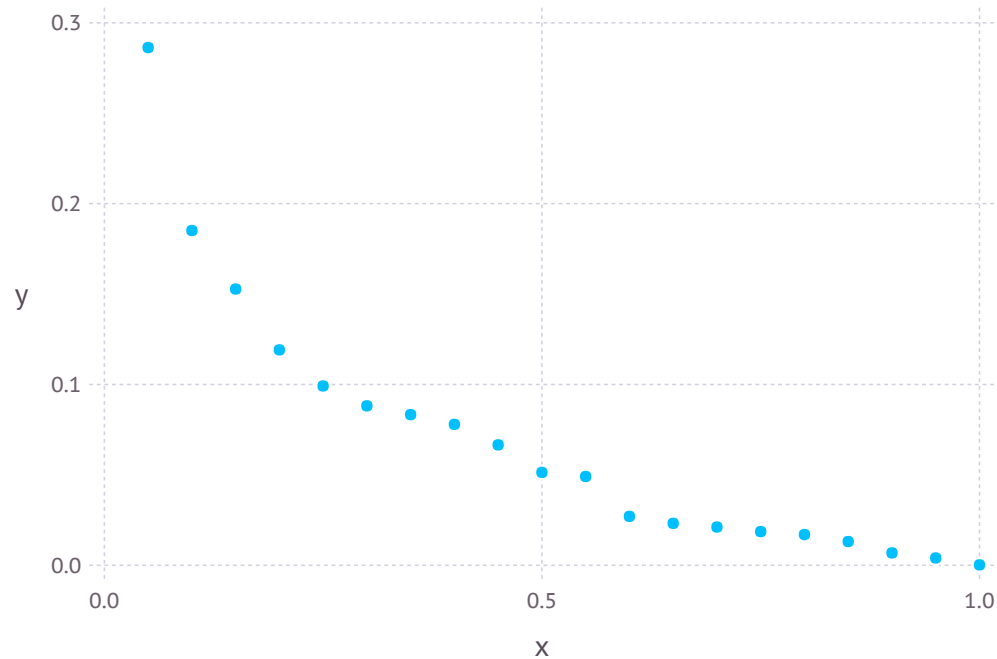


```
In [24]: y=mean(LDMat,1)
         sort(y,2)
```

```
Out[24]: 1x20 Array{Float64,2}:
          0.000258222  0.00409163  0.00688406  ...  0.152801  0.18519  0.286333
```

```
In [25]: plot(x=(1:20)/20*1,y=y)
```

Out[25]:



```
In [26]: FCMstream = open("SNPCMF.txt", "w")
```

```
Out[26]: IOStream(<file SNPCMF.txt>)
```

```
In [27]: for i in 1:size(FCM,2)
         @printf(FCMstream, "%6.4f ", FCM[1,i])
         end
```

```
In [28]: close(FCMstream)
```

## Selection - increase

```
In [29]: aSPSire = XSim.getOurGenVals(popSP[1])
aSPDam = XSim.getOurGenVals(popSP[2])
aSP = [aSPSire;aSPDam];
```

```
In [30]: mean(aSP)
```

```
Out[30]: 10.13647884948534
```

```
In [31]: varGen=var(aSP)
```

```
Out[31]: 0.6686641409621547
```

```
In [32]: XSim.common.varRes = varGen    #heritability = 0.5
```

```
Out[32]: 0.6686641409621547
```

```
In [33]: varRes = XSim.common.varRes
```

```
Out[33]: 0.6686641409621547
```

```
In [34]: popRMP = XSim.sampleSelNoLap(nSires, nDams, nGener, popSP[1], popSP[2], gen=popSP[3], fileName=fileName, direc
```

```
Generation      7: sampling 4000 males and 4000 females
Generation      8: sampling 4000 males and 4000 females
Generation      9: sampling 4000 males and 4000 females
Generation     10: sampling 4000 males and 4000 females
Generation     11: sampling 4000 males and 4000 females
```

```
In [35]: ymRMP = XSim.getOurGenVals(popRMP[1])    # for males: pop[1]
mean(ymRMP)
```

```
Out[35]: 12.574862863034017
```

```
In [36]: yfRMP = XSim.getOurGenVals(popRMP[2])    # for females: pop[2]
mean(yfRMP)
```

```
Out[36]: 12.572905426241759
```

```
In [37]: amRMP = XSim.getOurGenVals(popRMP[1])  
var(amRMP)
```

```
Out[37]: 0.4678352598774469
```

```
In [38]: afRMP = XSim.getOurGenVals(popRMP[2])  
var(afRMP)
```

```
Out[38]: 0.47590582867509934
```

## Pedigree: All animals

```
In [39]: PED = convert(Array,readtable(pedText,separator=' ',header=false))
```

```
Out[39]: 48000x3 Array{Int64,2}:  
  40722  32990  39746  
  40723  35429  39971  
  40724  36384  37375  
  40725  34808  37924  
  40726  33829  40089  
  40727  34822  39989  
  40728  32799  38210  
  40729  36507  37849  
  40730  33956  39668  
  40731  32847  37223  
  40732  35882  40027  
  40733  33100  36903  
  40734  36304  37608  
      ⋮  
  88710  74245  80435  
  88711  75176  79870  
  88712  76122  79555  
  88713  73429  79987  
  88714  75093  77758  
  88715  73299  78443  
  88716  74045  78802  
  88717  75868  78703  
  88718  75324  80642  
  88719  75070  80672  
  88720  74957  79168  
  88721  75968  78937
```

```
In [40]: PEDstream = open(PedAll, "w")
```

```
Out[40]: IOStream(<file PedAll.txt>)
```

```
In [41]: for i in 1:size(PED,1)  
          @printf(PEDstream, "%19d%19d%19d \n", PED[i,1], PED[i,2], PED[i,3])  
        end
```

```
In [42]: close(PEDstream)
```

# Create matrix of "genotype" covariates for all animals

```
In [43]: nObs = countlines(pedText)
```

```
Out[43]: 48000
```

```
In [44]: nMarker = numChr*numLoci
```

```
Out[44]: 200
```

```
In [45]: GT = convert(Array,readtable(genText,separator=' ',header=false))
```

```
Out[45]: 48000x201 Array{Int64,2}:
```

```
40722 0 2 0 2 2 1 1 1 1 1 ... 1 0 1 2 0 1 1 0 0 1 2 2
40723 1 2 0 2 2 1 1 1 1 1 ... 2 1 2 1 2 1 1 1 1 1 2 1
40724 0 2 0 2 2 0 0 0 2 0 ... 2 1 2 1 1 2 0 1 1 1 1 1
40725 0 2 1 2 1 1 1 1 1 1 ... 2 0 0 2 1 2 0 0 0 1 2 1
40726 0 2 0 2 2 0 0 0 2 0 ... 2 1 1 2 2 2 0 0 0 1 1 1
40727 0 2 0 2 2 0 0 0 2 0 ... 1 0 1 2 0 1 1 0 0 1 2 2
40728 0 1 1 1 1 0 1 1 1 1 ... 2 1 1 1 2 2 0 1 1 1 2 0
40729 0 1 1 2 2 0 1 1 1 1 ... 2 1 1 1 1 2 0 1 1 0 2 1
40730 1 2 0 2 2 0 0 0 2 0 ... 1 0 1 2 1 1 1 0 0 2 2 1
40731 0 2 0 2 2 0 0 0 2 0 ... 1 1 2 2 1 1 1 0 0 2 2 2
40732 0 2 2 2 0 0 0 0 2 0 ... 1 1 2 1 1 1 1 1 1 1 2 1
40733 0 2 1 2 1 0 0 0 2 0 ... 1 0 1 2 0 1 1 0 0 1 2 2
40734 1 2 0 2 2 0 1 1 1 1 ... 2 1 1 2 0 1 0 0 0 1 1 2
      ⋮      ⋮      ⋮      ⋮      ⋮      ⋮      ⋮      ⋮      ⋮      ⋮
88710 0 1 2 2 1 1 1 1 1 1 ... 2 2 2 1 2 2 0 1 1 1 2 0
88711 0 1 1 1 1 0 1 1 1 1 ... 2 1 0 2 1 2 0 0 0 0 2 2
88712 0 2 1 2 1 0 0 0 2 1 ... 1 0 1 2 0 1 1 0 0 1 2 2
88713 0 2 0 2 2 2 2 2 0 2 ... 2 1 2 1 2 2 0 1 1 0 2 1
88714 0 2 0 2 2 0 0 0 2 0 ... 2 2 2 0 2 2 0 2 2 0 2 1
88715 0 2 1 2 1 1 1 1 1 1 ... 2 0 2 1 2 2 0 1 1 1 1 2
88716 0 1 1 1 1 0 1 1 1 1 ... 2 2 1 1 2 2 0 1 1 1 1 1
88717 0 2 0 2 2 0 1 1 1 1 ... 2 2 1 1 2 2 0 1 2 0 2 0
88718 0 2 1 2 1 0 0 0 2 0 ... 2 1 2 2 2 2 0 0 0 2 1 2
88719 0 2 1 2 1 0 0 0 2 0 ... 2 1 2 1 2 2 0 1 1 0 2 1
88720 0 2 0 2 2 0 1 1 1 1 ... 2 2 2 0 2 2 0 2 2 0 2 0
88721 0 2 0 2 2 1 1 1 1 1 ... 2 2 2 1 2 2 0 1 1 1 1 1
```

```
In [46]: allID = GT[:,1];
```

```
In [47]: GTM = GT[:,2:end];
```

## Create marker file for all animals

```
In [48]: M = GTM      # maker file for Julia
```

```
Out[48]: 48000x200 Array{Int64,2}:
```

```
 0  2  0  2  2  1  1  1  1  1  1  0  1  ...  1  0  1  2  0  1  1  0  0  1  2  2
 1  2  0  2  2  1  1  1  1  1  1  0  1  ...  2  1  2  1  2  1  1  1  1  1  2  1
 0  2  0  2  2  0  0  0  2  0  0  1  1  ...  2  1  2  1  1  2  0  1  1  1  1  1
 0  2  1  2  1  1  1  1  1  1  1  1  1  ...  2  0  0  2  1  2  0  0  0  1  2  1
 0  2  0  2  2  0  0  0  2  0  0  2  2  ...  2  1  1  2  2  2  0  0  0  1  1  1
 0  2  0  2  2  0  0  0  2  0  0  0  0  ...  1  0  1  2  0  1  1  0  0  1  2  2
 0  1  1  1  1  0  1  1  1  1  1  0  1  ...  2  1  1  1  2  2  0  1  1  1  2  0
 0  1  1  2  2  0  1  1  1  1  1  0  1  ...  2  1  1  1  1  2  0  1  1  0  2  1
 1  2  0  2  2  0  0  0  2  0  0  1  1  ...  1  0  1  2  1  1  1  0  0  2  2  1
 0  2  0  2  2  0  0  0  2  0  0  0  0  ...  1  1  2  2  1  1  1  0  0  2  2  2
 0  2  2  2  0  0  0  0  2  0  0  2  0  ...  1  1  2  1  1  1  1  1  1  2  1
 0  2  1  2  1  0  0  0  2  0  0  2  1  ...  1  0  1  2  0  1  1  0  0  1  2  2
 1  2  0  2  2  0  1  1  1  1  1  0  0  ...  2  1  1  2  0  1  0  0  0  1  1  2
⋮          ⋮          ⋮          ⋮          ⋮
 0  1  2  2  1  1  1  1  1  1  1  1  ...  2  2  2  1  2  2  0  1  1  1  2  0
 0  1  1  1  1  0  1  1  1  1  1  1  2  ...  2  1  0  2  1  2  0  0  0  0  2  2
 0  2  1  2  1  0  0  0  2  1  1  1  2  ...  1  0  1  2  0  1  1  0  0  1  2  2
 0  2  0  2  2  2  2  2  0  2  1  0  2  ...  2  1  2  1  2  2  0  1  1  0  2  1
 0  2  0  2  2  0  0  0  2  0  0  1  1  ...  2  2  2  0  2  2  0  2  2  0  2  1
 0  2  1  2  1  1  1  1  1  1  1  1  1  ...  2  0  2  1  2  2  0  1  1  1  1  2
 0  1  1  1  1  0  1  1  1  1  1  1  2  ...  2  2  1  1  2  2  0  1  1  1  1  1
 0  2  0  2  2  0  1  1  1  1  1  1  2  ...  2  2  1  1  2  2  0  1  2  0  2  0
 0  2  1  2  1  0  0  0  2  0  0  2  2  ...  2  1  2  2  2  2  0  0  0  2  1  2
 0  2  1  2  1  0  0  0  2  0  0  1  0  ...  2  1  2  1  2  2  0  1  1  0  2  1
 0  2  0  2  2  0  1  1  1  1  1  0  1  ...  2  2  2  0  2  2  0  2  2  0  2  0
 0  2  0  2  2  1  1  1  1  1  1  1  2  ...  2  2  2  1  2  2  0  1  1  1  1  1
```

```
In [49]: Mstream = open(GenAll, "w")
```

```
Out[49]: IOStream(<file GenAll.txt>)
```

```
In [50]: for i in 1:size(M,1)
          @printf(Mstream, "%19d", allID[i])
          for j in 1:size(M,2)
              @printf(Mstream, "%3d", M[i,j])
          end
          @printf(Mstream, "\n")
        end
```

```
In [51]: close(Mstream)
```

## Genotypes - all sires and all offsprings in G5

```
In [52]: AllSire = PED[posOFF1S:posOFF5E,2]
```

```
Out[52]: 40000-element Array{Int64,1}:
```

```
42085
41368
41023
43665
44325
43478
44637
42376
44225
41945
41794
43094
42390
⋮
74245
75176
76122
73429
75093
73299
74045
75868
75324
75070
74957
75968
```



```
In [53]: SireID = unique(AllSire)
```

```
Out[53]: 1000-element Array{Int64,1}:
```

```
42085
41368
41023
43665
44325
43478
44637
42376
44225
41945
41794
43094
42390
⋮
76025
74283
75919
76406
75164
75078
75344
76289
73355
76445
75371
73852
```

```
In [54]: OFF5 = PED[posOFF5S:posOFF5E,1]
```

```
Out[54]: 8000-element Array{Int64,1}:
```

```
80722
80723
80724
80725
80726
80727
80728
80729
80730
80731
80732
80733
80734
      ⋮
88710
88711
88712
88713
88714
88715
88716
88717
88718
88719
88720
88721
```

```
In [55]: SireOFF5ID = [SireID;OFF5]
```

```
Out[55]: 9000-element Array{Int64,1}:
 42085
 41368
 41023
 43665
 44325
 43478
 44637
 42376
 44225
 41945
 41794
 43094
 42390
      ⋮
 88710
 88711
 88712
 88713
 88714
 88715
 88716
 88717
 88718
 88719
 88720
 88721
```

```
In [56]: SOFF5ID= DataFrame()
         SOFF5ID[:ID] = SireOFF5ID;
```

```
In [57]: typeof(SOFF5ID)
```

```
Out[57]: DataFrames.DataFrame
```

```
In [58]: MT = readtable(GenAll,separator=' ',header=false);
```

```
In [59]: rename!(MT,:x1,:ID);
```

```
In [60]: GSOFF5 = join(SOFF5ID, MT, on = :ID, kind = :inner);
```

```
In [61]: size(GSOFF5)
```

```
Out[61]: (9000,201)
```

```
In [62]: GSOFF5Row = size(GSOFF5,1)
```

```
Out[62]: 9000
```

```
In [63]: GSOFF5Col = size(GSOFF5,2)
```

```
Out[63]: 201
```

```
In [64]: GSOFF5stream = open(Gen, "w")
```

```
Out[64]: IOStream(<file Gen.txt>)
```

```
In [65]: for i in 1:size(GSOFF5,1)
         for j in 1
             @printf(GSOFF5stream, "%19d", GSOFF5[i,j])
         end
         for k in 2:size(GSOFF5,2)
             @printf(GSOFF5stream, "%3d", GSOFF5[i,k])
         end
         @printf(GSOFF5stream, "\n")
     end
```

```
In [66]: close(GSOFF5stream)
```

## Phenotypes - All animals

```
In [67]: PBV = convert(Array,readtable(pheText,separator=' ',header=false))
```

```
Out[67]: 48000x3 Array{Real,2}:
 40722    7.831    8.393
 40723   11.294   11.369
 40724    9.601    9.367
 40725    6.369    8.972
 40726    9.398    9.775
 40727   11.55    9.377
 40728   11.523   10.18
 40729    9.239    9.174
 40730   11.117   10.365
 40731   11.628   10.776
 40732   10.949   10.975
 40733    9.404    9.188
 40734    8.223    9.574
      ⋮
 88710   13.012   13.765
 88711   13.829   12.568
 88712   11.439   11.98
 88713   12.177   12.157
 88714   11.242   11.968
 88715   13.848   13.361
 88716   14.192   13.371
 88717   14.65    13.967
 88718   12.39    13.362
 88719   13.435   13.965
 88720   12.238   12.568
 88721   14.026   13.971
```

```
In [68]: PBVstream = open(PheAll, "w")
```

```
Out[68]: IOStream(<file PheAll.txt>)
```

```
In [69]: for i in 1:size(PBV,1)
          @printf(PBVstream, "%19d %10.3f %10.3f \n", PBV[i,1], PBV[i,2], PBV[i,3])
          end
```

```
In [70]: close(PBVstream )
```

# Phenotypes - all animals from G0 to G4

```
In [71]: OFFG0toG4 = PED[posOFF0S:posOFF4E,1]
```

```
Out[71]: 40000-element Array{Int64,1}:  
 40722  
 40723  
 40724  
 40725  
 40726  
 40727  
 40728  
 40729  
 40730  
 40731  
 40732  
 40733  
 40734  
  ⋮  
 80710  
 80711  
 80712  
 80713  
 80714  
 80715  
 80716  
 80717  
 80718  
 80719  
 80720  
 80721
```

```
In [72]: OFFG0toG4ID= DataFrame()  
OFFG0toG4ID[:ID] = OFFG0toG4;
```

```
In [73]: typeof(OFFG0toG4ID)
```

```
Out[73]: DataFrames.DataFrame
```

```
In [74]: AllPBV= readtable(pheText,separator=' ',header=false);
```

```
In [75]: rename!(AllPBV,:x1,:ID)
         head(AllPBV);
```

```
In [76]: OFFG0toG4PBV = join(OFFG0toG4ID, AllPBV, on = :ID, kind = :inner);
```

```
In [77]: Row = size(OFFG0toG4PBV,1)
```

```
Out[77]: 40000
```

```
In [78]: Col = size(OFFG0toG4PBV,2)
```

```
Out[78]: 3
```

```
In [79]: Phestream = open(Phe, "w")
```

```
Out[79]: IOStream(<file Phe.txt>)
```

```
In [80]: for i in 1:size(OFFG0toG4PBV,1)
         @printf(Phestream, "%19d %10.3f %10.3f \n", OFFG0toG4PBV[i,1], OFFG0toG4PBV[i,2], OFFG0toG4PBV[i,3])
         end
```

```
In [81]: close(Phestream)
```

## Get files with QTL only or Markers only

### QTL file and Markers file

```
In [82]: QTLPos = XSim.common.G.qtl_index
```

```
Out[82]: 50-element Array{Int64,1}:
```

```
 9
10
13
14
20
29
30
33
34
40
49
50
53
⋮
154
160
169
170
173
174
180
189
190
193
194
200
```



```
In [83]: k = size(M,2)
MarkerPos = deleteat!(collect(1:k),sort(QTLPos))
```

```
Out[83]: 150-element Array{Int64,1}:
```

```
 1
 2
 3
 4
 5
 6
 7
 8
11
12
15
16
17
 ⋮
184
185
186
187
188
191
192
195
196
197
198
199
```

## Genotype codes: 0, 1, 2

```
In [84]: IDgen = convert(Array,readtable(Gen,separator=' ',header=false));
```

```
In [85]: onlyID = IDgen[:,1]
        QTLMarker = IDgen[:, 2:end]
```

```
Out[85]: 9000x200 Array{Int64,2}:
```

```
 1  2  0  2  2  1  1  0  2  0  0  1  1  ...  2  0  2  2  1  2  0  0  0  1  0  2
 0  2  0  2  2  0  0  0  2  0  0  0  0  ...  2  1  1  1  2  2  0  1  1  1  2  0
 0  2  1  2  1  0  0  0  2  0  0  1  0  ...  1  1  2  1  1  1  1  1  1  1  2  1
 0  1  1  2  2  1  1  1  1  1  1  1  2  ...  2  1  1  1  1  2  0  1  1  0  2  1
 0  2  0  2  2  0  0  0  2  0  0  2  2  ...  2  1  1  2  2  2  0  0  0  1  1  2
 0  2  0  2  2  0  0  0  2  0  0  0  0  ...  2  1  1  1  1  2  0  1  1  0  2  1
 0  2  1  2  1  0  0  0  2  0  0  2  1  ...  1  1  2  2  1  1  1  0  0  2  2  1
 0  1  2  1  0  0  1  1  1  1  1  1  1  ...  2  0  0  2  1  2  0  0  0  1  2  1
 1  2  1  2  1  1  1  1  1  1  1  1  1  ...  1  2  1  2  1  2  0  0  0  1  1  2
 0  1  1  2  1  0  0  1  1  1  1  1  0  ...  2  2  1  1  1  2  1  1  1  1  2  1
 0  2  0  2  2  0  0  0  2  0  0  2  2  ...  2  2  2  1  2  2  0  1  1  1  2  1
 0  1  1  2  2  1  1  1  1  1  1  0  1  ...  2  0  1  2  0  2  0  0  0  0  1  2
 0  2  0  2  2  0  0  0  2  0  0  1  2  ...  1  0  2  2  1  1  1  0  0  2  1  2
⋮          ⋮          ⋮          ⋮          ⋮          ⋮
 0  1  2  2  1  1  1  1  1  1  1  1  ...  2  2  2  1  2  2  0  1  1  1  2  0
 0  1  1  1  1  0  1  1  1  1  1  1  2  ...  2  1  0  2  1  2  0  0  0  0  2  2
 0  2  1  2  1  0  0  0  2  1  1  1  2  ...  1  0  1  2  0  1  1  0  0  1  2  2
 0  2  0  2  2  2  2  2  0  2  1  0  2  ...  2  1  2  1  2  2  0  1  1  0  2  1
 0  2  0  2  2  0  0  0  2  0  0  1  1  ...  2  2  2  0  2  2  0  2  2  0  2  1
 0  2  1  2  1  1  1  1  1  1  1  1  1  ...  2  0  2  1  2  2  0  1  1  1  1  2
 0  1  1  1  1  0  1  1  1  1  1  1  2  ...  2  2  1  1  2  2  0  1  1  1  1  1
 0  2  0  2  2  0  1  1  1  1  1  1  2  ...  2  2  1  1  2  2  0  1  2  0  2  0
 0  2  1  2  1  0  0  0  2  0  0  2  2  ...  2  1  2  2  2  2  0  0  0  2  1  2
 0  2  1  2  1  0  0  0  2  0  0  1  0  ...  2  1  2  1  2  2  0  1  1  0  2  1
 0  2  0  2  2  0  1  1  1  1  1  0  1  ...  2  2  2  0  2  2  0  2  2  0  2  0
 0  2  0  2  2  1  1  1  1  1  1  1  2  ...  2  2  2  1  2  2  0  1  1  1  1  1
```

```
In [86]: onlyQTL = QTLMarker[:,QTLPos]
```

```
Out[86]: 9000x50 Array{Int64,2}:
```

```
 2  0  1  1  2  1  0  1  1  1  1  1  2  ...  0  0  0  1  0  2  2  2  0  1  2  2
 2  0  0  2  1  0  0  0  2  2  1  1  0      0  1  2  2  0  2  1  2  1  2  2  0
 2  0  0  2  1  2  0  1  1  1  0  0  0      0  1  2  2  1  1  1  1  1  1  1  1
 1  1  2  0  2  2  0  2  0  1  2  2  2      0  0  1  1  0  2  1  2  1  1  2  1
 2  0  2  0  2  2  0  1  1  1  1  1  0      1  0  2  2  0  2  2  2  1  2  2  2
 2  0  0  2  0  1  0  1  1  1  2  2  2  ...  1  2  0  2  0  2  1  2  1  1  2  1
 2  0  1  1  2  2  0  1  1  0  1  0  1      0  1  1  1  0  1  2  1  1  1  1  1
 1  1  1  2  1  2  0  2  0  2  2  2  1      1  1  1  2  0  1  2  2  0  1  2  1
 1  1  1  1  2  2  0  2  0  1  1  1  1      0  0  1  1  0  2  2  1  2  1  2  2
 1  1  0  2  0  1  0  1  1  1  0  1  1      1  1  0  2  0  2  2  2  2  1  2  1
 2  0  2  0  2  2  0  2  0  2  2  2  2  ...  1  0  1  1  0  2  2  2  2  2  2  1
 1  1  1  1  0  2  0  0  2  1  2  2  1      0  1  2  2  0  2  1  2  0  0  2  2
 2  0  2  0  2  1  0  0  2  2  0  0  0      0  2  1  2  0  2  2  1  0  1  1  2
⋮          ⋮          ⋮          ⋮          ⋮          ⋮
 1  1  1  1  2  2  0  0  2  2  2  2  1      0  1  1  2  1  2  1  2  2  2  2  0
 1  1  2  1  1  2  1  1  1  2  1  1  1      0  1  1  2  1  2  1  2  1  1  2  2
 2  1  2  1  2  2  0  1  1  2  2  1  1  ...  0  1  2  2  0  2  2  1  0  0  1  2
 0  2  2  2  1  2  0  0  2  1  2  1  2      1  1  0  1  0  2  2  2  1  2  2  1
 2  0  1  1  1  2  0  1  1  1  2  2  1      0  0  2  2  0  2  2  2  2  2  2  1
 1  1  1  2  1  1  0  0  2  1  2  2  0      0  1  2  2  1  2  2  2  0  2  2  2
 1  1  2  1  1  2  0  1  1  1  2  2  2      0  1  2  2  0  2  2  2  2  2  2  1
 1  1  2  1  1  2  0  1  1  2  2  2  2  ...  0  2  2  2  1  2  2  2  2  2  2  0
 2  0  2  1  0  2  0  0  2  1  2  2  2      1  1  2  2  1  2  1  2  1  2  2  2
 2  0  0  2  2  2  0  1  1  2  2  2  1      0  1  1  2  0  2  2  2  1  2  2  1
 1  1  1  2  0  2  0  2  0  0  2  1  2      0  1  2  2  1  2  2  2  2  2  2  0
 1  1  2  1  2  2  0  2  0  2  2  2  2      1  1  2  2  1  2  2  2  2  2  2  1
```

```
In [87]: onlyMar = QTLMarker[:,MarkerPos];
```

```
In [88]: QTLstream = open(QTL, "w")
Marstream = open(Mar, "w");
```

```
In [89]: for i in 1:size(onlyID,1)
          @printf(QTLstream, "%19d", onlyID[i])
          for j in 1:size(onlyQTL,2)
              @printf(QTLstream, "%3d", onlyQTL[i,j])
          end
          @printf(QTLstream, "\n")
      end
```

```
In [90]: for i in 1:size(onlyID,1)
          @printf(Marstream, "%19d", onlyID[i])
          for j in 1:size(onlyMar,2)
              @printf(Marstream, "%3d", onlyMar[i,j])
          end
          @printf(Marstream, "\n")
      end
```

```
In [91]: close(QTLstream)
          close(Marstream)
```

## Remove Fixed Gene from the panel

```
In [92]: VQM = var(QTLMarker,1)
          QMnoFixed = QTLMarker[:,VQM .> 0]
          VQ = var(onlyQTL,1)
          QnoFixed = onlyQTL[:,VQ .> 0]
          VM = var(onlyMar,1)
          MnoFixed = onlyMar[:,VM .> 0];
```

```
In [93]: GenNFstream = open(GenNF, "w")
          QTLNFstream = open(QTLNF, "w")
          MarNFstream = open(MarNF, "w");
```

```
In [94]: for i in 1:size(onlyID,1)
          @printf(GenNFstream, "%19d", onlyID[i])
          for j in 1:size(QMnoFixed,2)
              @printf(GenNFstream, "%3d", QMnoFixed[i,j])
          end
          @printf(GenNFstream, "\n")
        end
```

```
In [95]: for i in 1:size(onlyID,1)
          @printf(QTLNFstream, "%19d", onlyID[i])
          for j in 1:size(QnoFixed,2)
              @printf(QTLNFstream, "%3d", QnoFixed[i,j])
          end
          @printf(QTLNFstream, "\n")
        end
```

```
In [96]: for i in 1:size(onlyID,1)
          @printf(MarNFstream, "%19d", onlyID[i])
          for j in 1:size(MnoFixed,2)
              @printf(MarNFstream, "%3d", MnoFixed[i,j])
          end
          @printf(MarNFstream, "\n")
        end
```

```
In [97]: close(GenNFstream)
          close(QTLNFstream)
          close(MarNFstream)
```

## Check heritability

```
In [98]: P = AllPBV[:,2]
          BV = AllPBV[:,3];
```

```
In [99]: VP = var(P)
          VBV = var(BV)
          H = VBV/VP
```

```
Out[99]: 0.6292579552729338
```

```
In [100]: cor=cor(P,BV)
```

```
WARNING: imported binding for cor overwritten in module Main
```

```
Out[100]: 0.797679671644373
```

```
In [101]: QTLAll = M[:,QTLPos]
```

```
Out[101]: 48000x50 Array{Int64,2}:
```

```
 1  1  1  2  2  1  0  1  1  2  0  0  0  ...  1  0  0  0  0  0  2  1  0  0  1  2
 1  1  1  2  1  1  0  0  2  2  1  1  1  ...  0  1  2  2  0  1  1  2  1  2  1  1
 2  0  1  1  1  2  0  1  1  1  1  1  0  ...  0  0  0  1  0  2  1  2  1  1  2  1
 1  1  1  2  2  2  0  1  1  0  1  2  2  ...  0  1  0  0  0  2  1  2  0  1  2  1
 2  0  2  0  1  2  0  0  2  1  2  2  1  ...  0  1  0  1  0  2  0  2  1  2  2  1
 2  0  0  2  1  2  0  1  1  0  1  1  0  ...  0  1  0  1  0  2  1  1  0  0  1  2
 1  1  1  1  0  2  0  2  0  0  1  2  2  ...  0  1  2  2  0  1  2  2  1  2  2  0
 1  1  1  1  0  1  0  1  1  1  1  1  1  ...  1  0  1  2  0  2  2  2  1  1  2  1
 2  0  1  2  2  1  0  1  1  1  2  2  2  ...  1  1  1  2  0  1  2  1  0  1  1  1
 2  0  0  2  1  2  1  0  2  1  1  1  0  ...  0  2  2  2  1  2  2  1  1  1  1  2
 2  0  0  2  2  2  0  1  1  1  1  1  1  ...  0  2  1  2  1  2  1  1  1  1  1  1
 2  0  1  1  2  2  0  1  1  2  1  1  0  ...  0  0  0  2  0  1  2  1  0  0  1  2
 1  1  0  2  0  2  0  0  2  2  1  1  1  ...  0  0  0  0  0  2  1  2  1  0  1  2
 ⋮          ⋮          ⋮          ⋮          ⋮          ⋮
 1  1  1  1  2  2  0  0  2  2  2  2  1  ...  0  1  1  2  1  2  1  2  2  2  2  0
 1  1  2  1  1  2  1  1  1  2  1  1  1  ...  0  1  1  2  1  2  1  2  1  1  2  2
 2  1  2  1  2  2  0  1  1  2  2  1  1  ...  0  1  2  2  0  2  2  1  0  0  1  2
 0  2  2  2  1  2  0  0  2  1  2  1  2  ...  1  1  0  1  0  2  2  2  1  2  2  1
 2  0  1  1  1  2  0  1  1  1  2  2  1  ...  0  0  2  2  0  2  2  2  2  2  2  1
 1  1  1  2  1  1  0  0  2  1  2  2  0  ...  0  1  2  2  1  2  2  2  0  2  2  2
 1  1  2  1  1  2  0  1  1  1  2  2  2  ...  0  1  2  2  0  2  2  2  2  2  2  1
 1  1  2  1  1  2  0  1  1  2  2  2  2  ...  0  2  2  2  1  2  2  2  2  2  2  0
 2  0  2  1  0  2  0  0  2  1  2  2  2  ...  1  1  2  2  1  2  1  2  1  2  2  2
 2  0  0  2  2  2  0  1  1  2  2  2  1  ...  0  1  1  2  0  2  2  2  1  2  2  1
 1  1  1  2  0  2  0  2  0  0  2  1  2  ...  0  1  2  2  1  2  2  2  2  2  2  0
 1  1  2  1  2  2  0  2  0  2  2  2  2  ...  1  1  2  2  1  2  2  2  2  2  2  1
```

```
In [102]: QTL=qtEffects[QTLPos]
```

```
Out[102]: 50-element Array{Float64,1}:
```

```
0.199948  
0.199  
0.200105  
0.196709  
0.201449  
0.199533  
0.199602  
0.20111  
0.197139  
0.201705  
0.19811  
0.201933  
0.198864  
:  
0.199774  
0.201042  
0.201795  
0.198727  
0.201079  
0.20216  
0.201178  
0.201163  
0.199429  
0.202469  
0.198936  
0.200039
```

```
In [103]: EAlpha=QTLAll*QTLo
```

```
Out[103]: 48000-element Array{Float64,1}:
```

```
 8.39184
11.4023
 9.40235
 8.99602
 9.80978
 9.39354
10.2093
 9.20465
10.4026
10.7999
10.9944
 9.20108
 9.57871
  ⋮
13.8093
12.6052
12.003
12.2034
12.0013
13.4022
13.4083
14.0085
13.3983
14.0208
12.604
14.0119
```

```
In [104]: meanEAlphaG0=mean(EAlpha[1:8000])    # our mu_g
```

```
Out[104]: 10.163368239389532
```

```
In [105]: meanEAlphaG1=mean(EAlpha[8001:16000])
```

```
Out[105]: 10.663102845544074
```

```
In [106]: meanEAlphaG2=mean(EAlpha[16001:24000])
```

```
Out[106]: 11.206920255531472
```



```
In [107]: meanEAlphaG3=mean(EAlpha[24001:32000])
```

```
Out[107]: 11.716955922606932
```

```
In [108]: meanEAlphaG4=mean(EAlpha[32001:40000])
```

```
Out[108]: 12.186201841400958
```

```
In [109]: meanEAlphaG5=mean(EAlpha[40001:48000])
```

```
Out[109]: 12.611249079081954
```

```
In [110]: EAlphaG=onlyQTL*QTLo
```

```
Out[110]: 9000-element Array{Float64,1}:
```

```
11.0099
10.7992
10.3882
11.2188
11.6119
11.7907
10.2099
11.6073
10.8111
10.3843
12.62
12.0015
10.8198
⋮
13.8093
12.6052
12.003
12.2034
12.0013
13.4022
13.4083
14.0085
13.3983
14.0208
12.604
14.0119
```

```
In [111]: meanEAlphaG=mean(EAlphaG)
```

```
Out[111]: 12.562530639542114
```

```
In [112]: meanBreedingValueSelected=meanEAlphaG-meanEAlphaG0    # Legarra mu_g
```

```
Out[112]: 2.3991624001525818
```

```
In [113]: meanEAlphaS0=mean(EAlphaG[1:200])
```

```
Out[113]: 11.16808242551392
```

```
In [114]: meanBreedingValueSelectedG0=meanEAlphaS0-meanEAlphaG0
```

```
Out[114]: 1.004714186124387
```

```
In [115]: meanEAlphaS1=mean(EAlphaG[201:400])
```

```
Out[115]: 11.754574812462284
```

```
In [116]: meanBreedingValueSelectedG1=meanEAlphaS1-meanEAlphaG0
```

```
Out[116]: 1.5912065730727516
```

```
In [117]: meanEAlphaS2=mean(EAlphaG[401:600])
```

```
Out[117]: 12.241703997890102
```

```
In [118]: meanBreedingValueSelectedG2=meanEAlphaS2-meanEAlphaG0
```

```
Out[118]: 2.07833575850057
```

```
In [119]: meanEAlphaS3=mean(EAlphaG[601:800])
```

```
Out[119]: 12.659312708376596
```

```
In [120]: meanBreedingValueSelectedG3=meanEAlphaS3-meanEAlphaG0
```

```
Out[120]: 2.495944468987064
```

```
In [121]: meanEAlphaS4=mean(EAlphaG[801:1000])
```

```
Out[121]: 13.040241671874021
```

```
In [122]: meanBreedingValueSelectedG4=meanEAlphaS4-meanEAlphaG0
```

```
Out[122]: 2.876873432484489
```

```
In [123]: meanEAlphaS5=mean(EAlphaG[1001:9000])
```

```
Out[123]: 12.611249079081954
```

```
In [124]: meanBreedingValueSelectedG5=meanEAlphaS5-meanEAlphaG0
```

```
Out[124]: 2.447880839692422
```