In [1]:
```
# Founders: real haplotype data (ch1to10.200SNP)
# "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 = 0.0
# 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

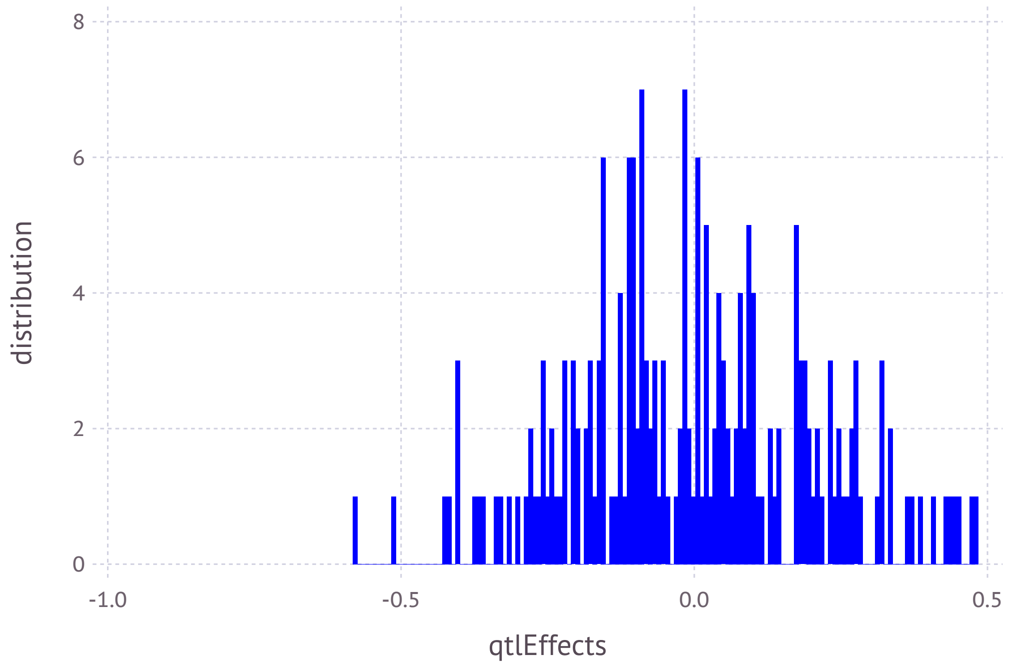
In [6]:
```
numChr    = 10
chrLength = 0.01
numLoci   = 20
nQTL      = 5
mutRate   = 0.0
locusInt  = chrLength/numLoci
mapPos    = collect(locusInt/2:locusInt:chrLength)
geneFreq  = fill(0.5,numLoci)
QTL = sample(1:numLoci,nQTL,replace=false)
qtlMarker = fill(false,numLoci)
qtlMarker[QTL] = true
Va = nQTL*numChr*0.5              # Va= nQTL*2pq*mean(alpha)^2; mean(alpha) = 0
qtlEffects= rand(Normal(0, sqrt(1/Va)),numLoci*numChr)  # Let alpha mu = 0.0
XSim.init(numChr,numLoci,chrLength,geneFreq,mapPos,qtlMarker,qtlEffects,mutRate)
```

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

```
Out[7]: 200-element Array{Float64,1}:
         -0.177566  
          0.17458   
          0.2783    
         -0.367997  
          0.0375599 
          0.0543789 
          0.211577  
          0.02186   
         -0.207689  
         -0.362845  
         -0.0959016 
         -0.0858694 
          0.43224   
          ⋮         
         -0.3011    
          0.439094  
          0.404024  
         -0.331992  
          0.309721  
         -0.178818  
         -0.0130945 
         -0.515316  
         -0.252505  
         -0.156184  
         -0.241069  
         -0.00572175
```

```
In [8]: writedlm("qtlEffects",qtlEffects)
In [9]: plot(x=qtlEffects, Geom.histogram, Guide.XLabel("qtlEffects"), Guide.YLabel("distribution"), Theme(defaul
```

Out[9]:



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

```
Out[10]: -0.004894489124528416
```

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

```
Out[11]: 0.043034794132427716
```

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 animsla
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
```

```
rm: cannot remove 'SS.ped': No such file or directory
rm: cannot remove 'SS.gen': No such file or directory
rm: cannot remove 'SS.phe': No such file or directory
```

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
```

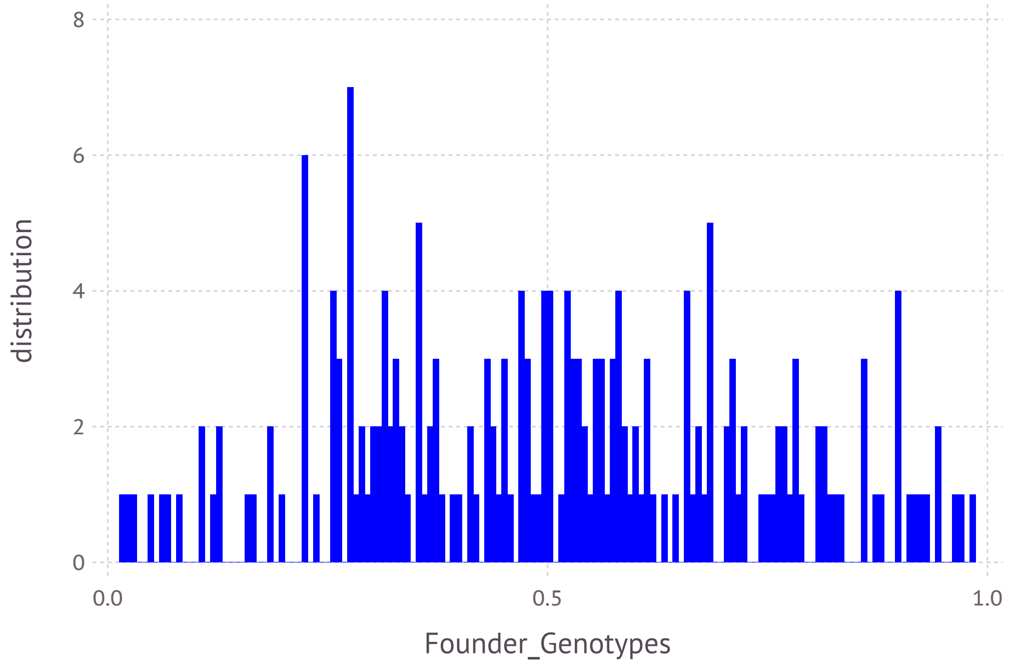
In [19]:
```
gSPSire = XSim.getOurGenotypes(popSP[1])
gSPDam = XSim.getOurGenotypes(popSP[2])
gSP = [gSPSire;gSPSire];
```

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

Out[20]:
```
1x200 Array{Float64,2}:
 0.067875  0.828625  0.297375  0.942  …  0.368625  0.3935  0.90175  0.550875
```

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

Out[21]:



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

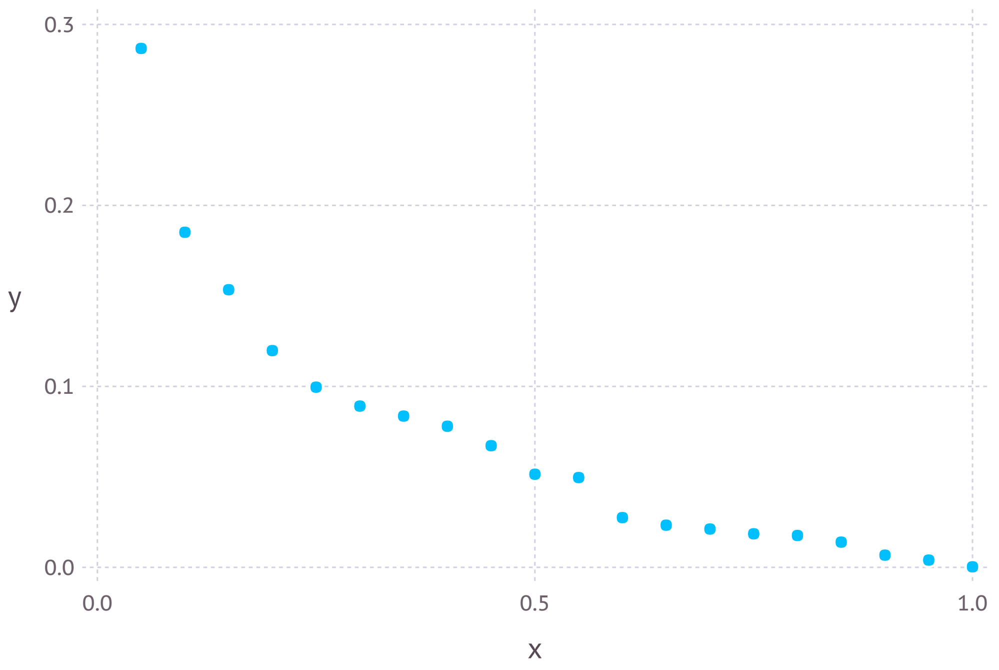
In [23]:
```
for i=1:(nRows-20)
LDMat[i,:] = corMat[i,(i+1):(i+20)].^2
end
```

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

Out[24]:
```
1x20 Array{Float64,2}:
 0.000328804  0.00401691  0.00676754  …  0.153413  0.185174  0.28676
```

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]:
```
1.2755789273511373
```

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

Out[31]:
```
0.5944510765095155
```

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

Out[32]:
```
0.5944510765095155
```

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

Out[33]:
```
0.5944510765095155
```

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]: 3.9629389662425214

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

Out[36]: 3.9522688483061303

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

Out[37]: 0.5052366993035663

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

Out[38]: 0.527085165324778
```

Pedigree: All animals

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

Out[39]:
```
48000x3 Array{Int64,2}:
 40722  35949  38612
 40723  36305  38696
 40724  36444  39253
 40725  35240  40434
 40726  36141  38375
 40727  36276  37663
 40728  35182  38422
 40729  35345  40685
 40730  33478  40682
 40731  35667  39928
 40732  35083  39306
 40733  35519  37389
 40734  35056  40134
     ⋮
 88710  75085  80317
 88711  76231  80090
 88712  76491  79722
 88713  73316  77886
 88714  73541  79125
 88715  73374  80083
 88716  74029  80454
 88717  76158  79428
 88718  73905  80686
 88719  73732  80609
 88720  74195  80681
 88721  76161  79847
```

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  0  0  0  2  0  …  1  1  2  1  1  1  1  1  1  1  2  1
 40723  0  2  0  2  2  0  0  0  2  0     2  1  2  1  2  2  0  2  2  0  2  0
 40724  0  2  0  2  2  0  0  0  2  0     1  1  2  1  1  1  1  1  1  1  2  1
 40725  0  2  0  2  2  0  0  0  2  0     2  0  2  2  1  1  0  0  0  1  1  2
 40726  0  2  1  2  1  1  1  1  1  1     2  1  2  1  2  2  0  1  1  0  2  1
 40727  0  0  2  1  1  1  2  2  0  2  …  2  0  0  2  1  2  0  0  0  1  2  1
 40728  0  2  0  2  2  1  1  1  1  1     2  0  0  2  2  2  0  0  0  2  1  1
 40729  0  2  0  2  2  0  0  0  2  0     2  1  2  0  2  2  0  2  2  0  2  0
 40730  0  0  2  1  1  1  2  2  0  2     1  1  2  1  1  1  1  1  1  1  2  1
 40731  1  2  0  2  2  1  1  1  1  1     1  1  2  1  1  1  1  1  1  1  2  1
 40732  0  2  0  2  2  1  1  1  1  1  …  2  2  2  0  2  2  0  2  2  0  2  0
 40733  0  1  1  2  2  2  2  2  0  2     1  1  2  1  1  1  1  1  1  1  2  1
 40734  0  0  2  2  1  0  1  1  1  1     2  0  1  2  2  2  0  0  0  2  1  1
     ⋮                 ⋮                 ⋮  ⋱     ⋮              ⋮              ⋮
 88710  0  2  2  2  0  0  0  0  2  0     2  1  0  2  1  2  0  0  0  0  2  2
 88711  0  2  1  2  1  0  1  1  2  0     2  1  1  1  1  2  1  1  1  1  1  1
 88712  0  2  2  2  0  0  0  0  2  0  …  2  2  1  1  1  1  0  1  1  0  1  1
 88713  0  1  1  2  2  0  1  1  1  1     1  1  2  2  1  1  1  0  0  2  1  2
 88714  0  2  1  2  1  0  0  0  2  0     1  0  1  2  0  1  2  0  0  1  1  2
 88715  0  2  0  2  2  0  0  0  2  0     2  1  1  2  1  1  0  0  0  0  2  2
 88716  0  1  1  2  2  0  1  1  1  1     2  1  2  1  2  1  0  1  1  0  2  1
 88717  1  1  1  2  2  1  2  2  0  2  …  2  1  1  1  2  2  0  1  1  1  2  0
 88718  0  2  1  2  1  0  0  0  2  0     2  1  1  2  0  1  1  0  0  0  1  2
 88719  0  1  2  2  1  1  1  1  1  1     2  0  0  2  1  2  1  0  0  2  1  1
 88720  0  0  2  2  2  1  2  2  0  2     2  1  2  1  2  1  0  1  1  0  2  1
 88721  0  1  1  1  1  0  0  0  2  0     2  0  1  2  2  1  0  0  0  1  2  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  0  0  0  2  0  0  1  1  …  1  1  2  1  1  1  1  1  1  1  2  1
 0  2  0  2  2  0  0  0  2  0  0  1  1     2  1  2  1  2  2  0  2  2  0  2  0
 0  2  0  2  2  0  0  0  2  0  0  2  2     1  1  2  1  1  1  1  1  1  1  2  1
 0  2  0  2  2  0  0  0  2  0  0  1  1     2  0  2  2  1  1  0  0  0  1  1  2
 0  2  1  2  1  1  1  1  1  1  1  1  1     2  1  2  1  2  2  0  1  1  0  2  1
 0  0  2  1  1  1  2  2  0  2  2  0  2  …  2  0  0  2  1  2  0  0  0  1  2  1
 0  2  0  2  2  1  1  1  1  1  1  0  1     2  0  0  2  2  2  0  0  0  2  1  1
 0  2  0  2  2  0  0  0  2  0  0  1  1     2  1  2  0  2  2  0  2  2  0  2  0
 0  0  2  1  1  1  2  2  0  2  2  0  2     1  1  2  1  1  1  1  1  1  1  2  1
 1  2  0  2  2  1  1  1  1  1  1  0  1     1  1  2  1  1  1  1  1  1  1  2  1
 0  2  0  2  2  1  1  1  1  1  1  0  1  …  2  2  2  0  2  2  0  2  2  0  2  0
 0  1  1  2  2  2  2  2  0  2  2  0  2     1  1  2  1  1  1  1  1  1  1  2  1
 0  0  2  2  1  0  1  1  1  1  1  1  1     2  0  1  2  2  2  0  0  0  2  1  1
 ⋮              ⋮              ⋮        ⋱        ⋮              ⋮
 0  2  2  2  0  0  0  0  2  0  0  2  1     2  1  0  2  1  2  0  0  0  0  2  2
 0  2  1  2  1  0  1  1  2  0  0  2  1     2  1  1  1  1  2  1  1  1  1  1  1
 0  2  2  2  0  0  0  0  2  0  0  2  0  …  2  2  1  1  1  1  0  1  1  0  1  1
 0  1  1  2  2  0  1  1  1  1  1  1  2     1  1  2  2  1  1  1  0  0  2  1  2
 0  2  1  2  1  0  0  0  2  0  0  1  1     1  0  1  2  0  1  2  0  0  1  1  2
 0  2  0  2  2  0  0  0  2  0  0  0  0     2  1  1  2  1  1  0  0  0  0  2  2
 0  1  1  2  2  0  1  1  1  1  1  0  1     2  1  2  1  2  1  0  1  1  0  2  1
 1  1  1  2  2  1  2  2  0  2  2  0  1  …  2  1  1  1  2  2  0  1  1  1  2  0
 0  2  1  2  1  0  0  0  2  0  0  1  1     2  1  1  2  0  1  1  0  0  0  1  2
 0  1  2  2  1  1  1  1  1  1  1  1  1     2  0  0  2  1  2  1  0  0  2  1  1
 0  0  2  2  2  1  2  2  0  2  2  0  1     2  1  2  1  2  1  0  1  1  0  2  1
 0  1  1  1  1  0  0  0  2  0  0  0  0     2  0  1  2  2  1  0  0  0  1  2  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}:
          42672
          40941
          41360
          44231
          41389
          44554
          41717
          41069
          43076
          43346
          42122
          44244
          43956
              ⋮
          75085
          76231
          76491
          73316
          73541
          73374
          74029
          76158
          73905
          73732
          74195
          76161
```

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

Out[53]: 1000-element Array{Int64,1}:
          42672
          40941
          41360
          44231
          41389
          44554
          41717
          41069
          43076
          43346
          42122
          44244
          43956
              ⋮
          73867
          76081
          75986
          73679
          73712
          76374
          76417
          76135
          75431
          73893
          74233
          75266
```

```
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}:
          42672
          40941
          41360
          44231
          41389
          44554
          41717
          41069
          43076
          43346
          42122
          44244
          43956
              ⋮
          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  -2.28   -0.383
          40723  -0.892   0.543
          40724  -0.449   1.29
          40725   0.762   1.374
          40726   0.811   1.662
          40727   0.831   1.322
          40728   1.436   1.482
          40729   1.382   0.842
          40730   0.391   0.277
          40731   0.776   0.964
          40732  -0.518   1.145
          40733   0.06    1.612
          40734   0.813   1.353
              ⋮
          88710   5.77    4.645
          88711   3.765   4.108
          88712   6.133   6.086
          88713   4.096   4.871
          88714   4.55    3.381
          88715   5.0     4.651
          88716   3.552   4.213
          88717   4.968   3.845
          88718   3.726   4.105
          88719   3.809   4.111
          88720   5.813   4.873
          88721   6.456   5.997
```

```
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 animnls 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}:
            1
            2
            3
            8
           11
           21
           22
           23
           28
           31
           41
           42
           43
            ⋮
          148
          151
          161
          162
          163
          168
          171
          181
          182
          183
          188
          191
```

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

```
Out[83]: 150-element Array{Int64,1}:
            4
            5
            6
            7
            9
           10
           12
           13
           14
           15
           16
           17
           18
            ⋮
          187
          189
          190
          192
          193
          194
          195
          196
          197
          198
          199
          200
```

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}:
 0  2  0  2  2  0  0  0  2  0  0  1  1  …  2  0  1  2  1  1  2  0  0  1  2  2
 0  2  1  2  2  0  1  1  1  1  1  0  0     2  1  1  1  1  2  0  2  2  0  2  0
 0  1  1  1  1  0  1  1  1  1  1  1  2     1  1  2  1  1  1  1  1  1  1  2  1
 0  2  0  2  2  0  1  1  2  0  0  2  2     2  0  0  2  0  2  0  0  0  0  2  2
 0  1  1  1  1  0  1  1  1  1  1  1  2     1  1  2  1  1  1  1  1  1  1  2  1
 0  2  0  2  2  0  0  0  2  0  0  1  1  …  2  1  1  1  1  2  0  0  0  0  2  2
 0  2  1  2  1  0  1  1  1  1  1  1  0     2  1  2  2  2  2  0  0  0  0  2  1
 0  1  1  2  2  1  1  1  1  1  1  1  2     2  2  2  0  2  2  0  2  2  0  2  0
 0  2  0  2  2  0  0  0  2  0  0  1  1     1  1  1  2  0  0  1  0  0  1  1  2
 0  1  2  2  1  0  0  0  2  0  0  2  1     1  1  1  2  0  0  1  0  0  1  1  2
 0  2  1  2  1  0  1  1  1  1  1  1  1  …  2  0  2  2  0  1  0  0  0  1  1  2
 0  0  2  2  2  1  2  2  0  2  2  0  2     2  1  1  2  1  2  0  0  0  1  2  1
 0  2  0  2  2  1  1  1  1  1  1  1  2     1  0  1  2  0  1  1  0  0  1  2  2
 ⋮              ⋮              ⋮        ⋱        ⋮              ⋮
 0  2  2  2  0  0  0  0  2  0  0  2  1     2  1  0  2  1  2  0  0  0  0  2  2
 0  2  1  2  1  0  1  1  2  0  0  2  1     2  1  1  1  1  2  1  1  1  1  1  1
 0  2  2  2  0  0  0  0  2  0  0  2  0  …  2  2  1  1  1  1  0  1  1  0  1  1
 0  1  1  2  2  0  1  1  1  1  1  1  2     1  1  2  2  1  1  1  0  0  2  1  2
 0  2  1  2  1  0  0  0  2  0  0  1  1     1  0  1  2  0  1  2  0  0  1  1  2
 0  2  0  2  2  0  0  0  2  0  0  0  0     2  1  1  2  1  1  0  0  0  0  2  2
 0  1  1  2  2  0  1  1  1  1  1  0  1     2  1  2  1  2  1  0  1  1  0  2  1
 1  1  1  2  2  1  2  2  0  2  2  0  1  …  2  1  1  1  2  2  0  1  1  1  2  0
 0  2  1  2  1  0  0  0  2  0  0  1  1     2  1  1  2  0  1  1  0  0  0  1  2
 0  1  2  2  1  1  1  1  1  1  1  1  1     2  0  0  2  1  2  1  0  0  2  1  1
 0  0  2  2  2  1  2  2  0  2  2  0  1     2  1  2  1  2  1  0  1  1  0  2  1
 0  1  1  1  1  0  0  0  2  0  0  0  0     2  0  1  2  2  1  0  0  0  1  2  1
```

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

```
Out[86]: 9000x50 Array{Int64,2}:
 0  2  0  0  0  2  1  1  1  0  0  2  0  …  1  1  0  1  1  2  1  2  1  2  2  1
 0  2  1  1  1  2  0  2  0  0  2  1  0     0  0  1  1  1  1  2  2  0  2  1  1
 0  1  1  1  1  2  1  1  0  1  1  1  1     0  1  1  2  2  1  0  2  0  1  1  2
 0  2  0  1  0  1  0  2  0  0  0  2  2     0  1  1  1  1  2  2  2  0  2  0  0
 0  1  1  1  1  1  0  2  1  0  0  2  0     0  1  2  0  0  2  1  2  0  1  1  2
 0  2  0  0  0  1  0  2  1  0  2  0  0  …  1  2  1  1  1  1  1  2  0  2  1  1
 0  2  1  1  1  0  0  2  2  0  1  1  0     1  1  2  1  1  2  0  1  2  2  1  2
 0  1  1  1  1  0  0  2  2  0  1  1  1     1  2  1  1  1  2  1  2  0  2  2  2
 0  2  0  0  0  2  2  0  0  0  0  2  1     1  1  0  2  2  2  1  1  1  1  0  1
 0  1  2  0  0  2  2  0  1  0  1  1  1     0  1  1  1  1  2  1  1  1  1  0  1
 0  2  1  1  1  2  0  2  0  0  1  1  1  …  0  1  0  2  2  2  1  2  1  2  0  2
 0  0  2  2  2  2  2  0  0  0  2  0  1     0  1  0  1  1  2  0  2  0  2  1  1
 0  2  0  1  1  0  0  2  1  1  0  2  1     1  1  0  2  2  1  1  2  0  1  0  1
 ⋮              ⋮              ⋮        ⋱        ⋮              ⋮
 0  2  2  0  0  1  0  2  1  0  1  1  1     0  1  1  1  0  1  1  1  2  2  0  0
 0  2  1  1  0  0  0  2  1  0  0  2  1     0  0  0  1  1  2  0  1  1  2  1  1
 0  2  2  0  0  1  1  1  2  0  0  2  2  …  0  1  1  1  1  2  0  0  2  2  1  1
 0  1  1  1  1  2  0  2  0  0  0  2  2     0  1  0  2  2  2  2  1  1  1  0  2
 0  2  1  0  0  2  0  2  0  0  0  2  0     0  0  2  2  2  2  0  1  1  2  0  1
 0  2  0  0  0  1  0  2  1  0  0  2  2     0  1  0  0  0  1  0  2  2  2  0  1
 0  1  1  1  1  1  1  1  1  0  0  2  0     0  2  0  2  2  2  1  1  1  2  1  2
 1  1  1  2  2  0  0  2  2  0  1  1  2  …  0  0  2  1  1  2  0  2  1  2  1  1
 0  2  1  0  0  0  0  2  2  0  1  1  1     0  0  0  1  1  2  1  1  1  2  0  1
 0  1  2  1  1  1  1  1  1  0  0  2  1     1  1  0  0  0  2  0  1  2  2  0  0
 0  0  2  2  2  0  0  2  2  0  2  0  1     0  0  1  2  2  2  1  0  2  2  1  2
 0  1  1  0  0  0  0  2  2  0  0  2  1     0  0  0  1  1  2  0  1  2  2  0  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]:
```
0.6847763303979175
```

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

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

Out[100]:
```
0.8257611682338226
```

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

Out[101]:
```
48000x50 Array{Int64,2}:
 0  2  0  0  0  2  0  2  0  0  1  1  1  …  1  1  2  0  0  1  1  2  0  1  1  2
 0  2  0  0  0  2  1  1  0  0  1  1  1     0  1  2  0  0  2  1  2  0  2  1  2
 0  2  0  0  0  1  1  1  1  0  2  0  2     1  1  1  0  0  2  1  2  0  1  1  2
 0  2  0  0  0  2  0  2  0  1  1  1  1     1  1  2  0  0  2  0  2  1  2  1  2
 0  2  1  1  1  1  0  2  1  0  1  1  1     0  2  2  0  0  1  0  2  1  2  1  2
 0  0  2  2  2  0  0  2  1  1  2  1  0  …  0  2  1  0  0  1  1  2  1  2  0  0
 0  2  0  1  1  1  1  1  1  0  1  1  1     0  1  0  1  1  1  0  2  2  2  0  0
 0  2  0  0  0  1  1  1  1  0  2  0  0     0  1  2  1  1  2  0  2  0  2  2  2
 0  0  2  2  2  1  1  1  1  0  2  0  0     1  1  1  1  0  1  2  2  0  1  1  2
 1  2  0  1  1  2  2  0  1  0  1  1  0     1  1  1  1  1  2  1  2  0  1  1  2
 0  2  0  1  1  1  2  0  1  0  1  1  1  …  2  2  1  1  0  2  1  2  0  2  2  2
 0  1  1  2  2  0  0  2  2  0  1  1  0     0  1  1  0  0  2  0  2  0  1  1  2
 0  0  2  1  1  2  0  2  0  0  0  2  0     0  1  2  0  0  2  1  2  2  2  1  1
 ⋮              ⋮              ⋮        ⋱        ⋮              ⋮
 0  2  2  0  0  1  0  2  1  0  1  1  1     0  1  1  1  0  1  1  1  2  2  0  0
 0  2  1  1  0  0  0  2  1  0  0  2  1     0  0  0  1  1  2  0  1  1  2  1  1
 0  2  2  0  0  1  1  1  2  0  0  2  2  …  0  1  1  1  1  2  0  0  2  2  1  1
 0  1  1  1  1  2  0  2  0  0  0  2  2     0  1  0  2  2  2  2  1  1  1  0  2
 0  2  1  0  0  2  0  2  0  0  0  2  0     0  0  2  2  2  2  0  1  1  2  0  1
 0  2  0  0  0  1  0  2  1  0  0  2  2     0  1  0  0  0  1  0  2  2  2  0  1
 0  1  1  1  1  1  1  1  1  0  0  2  0     0  2  0  2  2  2  1  1  1  2  1  2
 1  1  1  2  2  0  0  2  2  0  1  1  2  …  0  0  2  1  1  2  0  2  1  2  1  1
 0  2  1  0  0  0  0  2  2  0  1  1  1     0  0  0  1  1  2  1  1  1  2  0  1
 0  1  2  1  1  1  1  1  1  0  0  2  1     1  1  0  0  0  2  0  1  2  2  0  0
 0  0  2  2  2  0  0  2  2  0  2  0  1     0  0  1  2  2  2  1  0  2  2  1  2
 0  1  1  0  0  0  0  2  2  0  0  2  1     0  0  0  1  1  2  0  1  2  2  0  1
```

```
In [102]: QTLo=qtlEffects[QTLPos]
```

```
Out[102]: 50-element Array{Float64,1}:
           -0.177566
            0.17458
            0.2783
            0.02186
           -0.0959016
           -0.425432
            0.0725654
           -0.0916106
           -0.0163798
           -0.151692
           -0.125982
           -0.285777
            0.333185
            ⋮
           -0.0916457
           -0.0245496
           -0.100869
           -0.108177
            0.127595
            0.0211537
            0.484105
            0.276304
            0.136807
           -0.050276
            0.0911079
            0.404024
```

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

```
Out[103]: 48000-element Array{Float64,1}:
           2.21786 
           1.89701 
           3.98301 
           2.48107 
           2.97875 
           2.03293 
           0.959531
           2.17872 
           4.02043 
           3.19751 
           2.59128 
           3.49557 
           1.40712 
           ⋮       
           2.73586 
           2.19662 
           3.09294 
           3.48942 
           1.33222 
           2.64163 
           2.96979 
           3.969   
           3.67126 
           2.92082 
           4.21514 
           2.77279 
```

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

```
Out[104]: 2.7737742852940364
```

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

```
Out[105]: 2.8316988522603377
```

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

```
Out[106]: 2.9730191559732577
```

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

Out[107]: 3.0411178833504127

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

Out[108]: 3.1372013107283765

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

Out[109]: 3.1910002355779006

In [110]: EAlphaG=onlyQTL*QTLo

Out[110]: 9000-element Array{Float64,1}:
           1.03725
           3.02123
           2.10021
           2.73378
           2.88768
           2.04094
           3.13771
           4.28348
           0.938042
           3.77132
           2.5857
           1.56264
           2.96499
           ⋮
           2.73586
           2.19662
           3.09294
           3.48942
           1.33222
           2.64163
           2.96979
           3.969
           3.67126
           2.92082
           4.21514
           2.77279
```

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

Out[111]: 3.183046089895449

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

Out[112]: 0.40927180460141255

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

Out[113]: 2.8928923271713525

In [114]: meanBreedingValueSelectedG0=meanEAlphaS0-meanEAlphaG0

Out[114]: 0.11911804187731612

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

Out[115]: 3.0966312795508175

In [116]: meanBreedingValueSelectedG1=meanEAlphaS1-meanEAlphaG0

Out[116]: 0.32285699425678116

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

Out[117]: 3.1121273401168428

In [118]: meanBreedingValueSelectedG2=meanEAlphaS2-meanEAlphaG0

Out[118]: 0.3383530548228064

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

Out[119]: 3.2405956474993616

In [120]: meanBreedingValueSelectedG3=meanEAlphaS3-meanEAlphaG0

Out[120]: 0.4668213622053252
```

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

Out[121]: 3.254818027840758

In [122]: meanBreedingValueSelectedG4=meanEAlphaS4-meanEAlphaG0

Out[122]: 0.48104374254672155

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

Out[123]: 3.1910002355779006

In [124]: meanBreedingValueSelectedG5=meanEAlphaS5-meanEAlphaG0

Out[124]: 0.4172259502838642
```