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 = 0.1
# 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

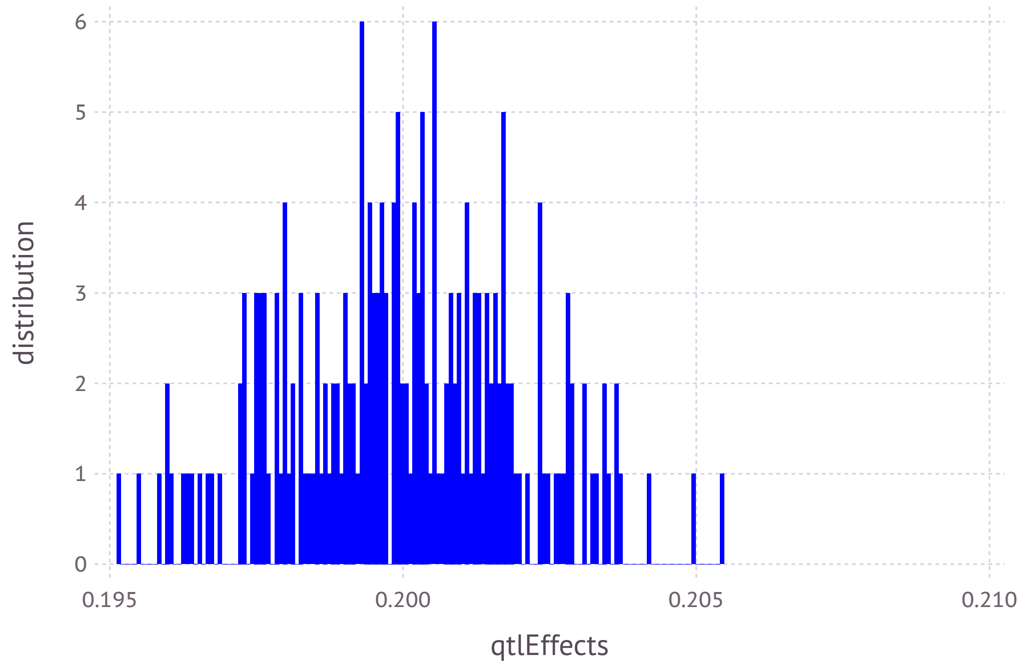
In [6]:
```
numChr    = 10
chrLength = 0.01
numLoci   = 20
nQTL      = 5
mutRate   = 0.0
locusInt  = chrLength/numLoci
mapPos    = collect(locusInt/2:locusInt:chrLength)
geneFreq  = fill(0.5,numLoci)
QTL = sample(1:numLoci,nQTL,replace=false)
qtlMarker = fill(false,numLoci)
qtlMarker[QTL] = true
mu = 100                                #  alpha ~ N(100,1)
Va = nQTL*numChr*0.5*mu*mu              # Va= nQTL*2pq*mean(alpha)^2
qtlEffects= rand(Normal(mu/sqrt(Va), sqrt(1/Va)),numLoci*numChr)  # Let alpha mu = 0.115
XSim.init(numChr,numLoci,chrLength,geneFreq,mapPos,qtlMarker,qtlEffects,mutRate)
```

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

```
Out[7]: 200-element Array{Float64,1}:
         0.199275
         0.201814
         0.201059
         0.201297
         0.197664
         0.196233
         0.201626
         0.197521
         0.199312
         0.200417
         0.199443
         0.20096
         0.196662
         ⋮
         0.196874
         0.201407
         0.202095
         0.199711
         0.197216
         0.198897
         0.199624
         0.201105
         0.200554
         0.200517
         0.197283
         0.203624
```

```
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]: 0.19998254028271392
```

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

```
Out[11]: 3.864368442083477e-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 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
```

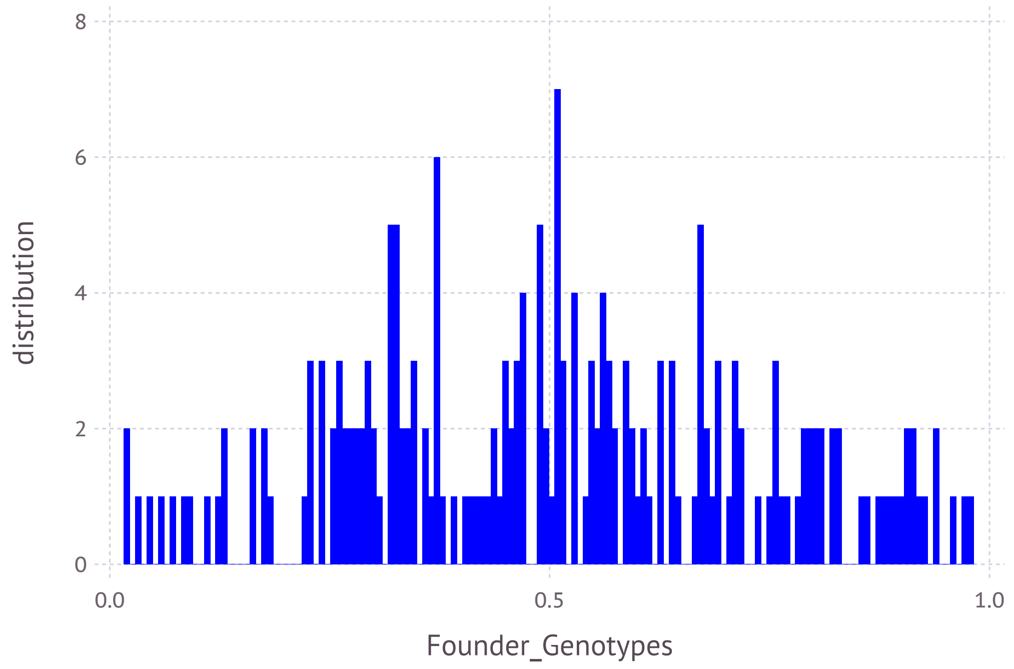
```
In [19]: gSPSire = XSim.getOurGenotypes(popSP[1])
         gSPDam = XSim.getOurGenotypes(popSP[2])
         gSP = [gSPSire;gSPSire];
```

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

```
Out[20]: 1x200 Array{Float64,2}:
          0.0615  0.82625  0.29225  0.940625  …  0.3715  0.37425  0.8945  0.551625
```

```
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);
```

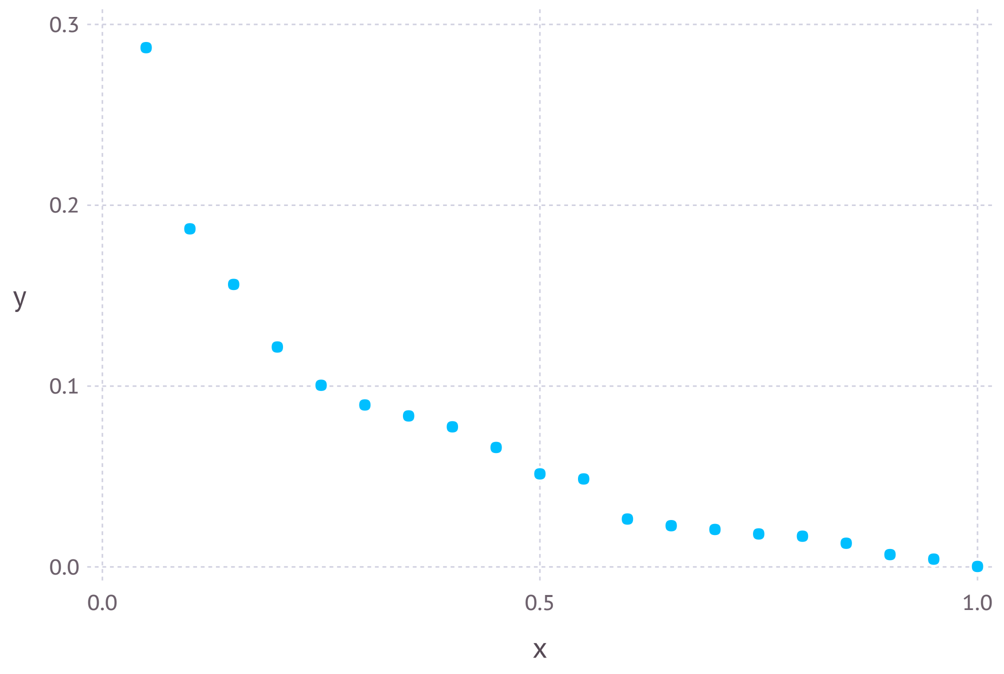
```
In [23]: for i=1:(nRows-20)
         LDMat[i,:] = corMat[i,(i+1):(i+20)].^2
         end
```

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

Out[24]: 1x20 Array{Float64,2}:
          0.000258184  0.00433984  0.00682853  …  0.156237  0.186966  0.287159
```

```
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]:
```
10.310189862698259
```

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

Out[31]:
```
0.8701619626472299
```

In [32]:
```
XSim.common.varRes = 9*varGen    #heritability = 0.1
```

Out[32]:
```
7.831457663825069
```

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

Out[33]:
```
7.831457663825069
```

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]: 12.02463829325232

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

Out[36]: 12.00498881454154

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

Out[37]: 0.8131290020394336

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

Out[38]: 0.8053363394626679
```

Pedigree: All animals

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

Out[39]:
```
48000x3 Array{Int64,2}:
 40722  35322  37430
 40723  34664  38225
 40724  36112  37513
 40725  36422  40139
 40726  36469  36867
 40727  34419  38094
 40728  34095  39204
 40729  35642  37974
 40730  36613  37399
 40731  36120  39732
 40732  35794  40495
 40733  34198  39665
 40734  33157  40455
     ⋮
 88710  75568  78115
 88711  75817  79108
 88712  73034  78475
 88713  76451  79276
 88714  75520  80325
 88715  73448  78600
 88716  73760  77269
 88717  74787  80581
 88718  75503  80417
 88719  76024  79390
 88720  72868  79577
 88721  76325  78232
```

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

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  1  1  1  1  0  1  1  1  1  1  0  1  …  1  1  2  1  1  1  1  1  1  1  2  1
          0  2  1  2  1  0  0  0  2  0  0  2  1     2  2  1  1  1  1  1  1  1  1  2  1
          0  2  1  2  2  1  1  1  2  0  0  0  0     1  0  2  2  2  1  0  1  1  0  2  1
          0  1  1  1  1  0  2  2  0  2  2  0  2     2  0  1  2  0  2  1  0  0  1  0  2
          0  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  2  2  0  0  0  0  2  0  0  2  1  …  2  1  1  1  1  2  0  1  1  0  2  1
          0  1  1  1  2  0  1  1  1  1  1  0  0     1  1  2  1  1  1  1  1  1  1  2  1
          0  1  1  1  1  0  1  1  1  1  1  0  1     2  2  2  0  2  2  0  2  2  0  2  0
          0  1  2  2  1  1  1  1  1  1  1  1  2     1  1  2  1  1  1  1  1  1  1  2  1
          2  2  0  2  2  0  0  0  2  0  0  2  1     2  1  1  1  1  2  0  1  1  0  2  1
          0  2  0  2  2  0  1  1  1  1  1  0  0  …  2  2  2  1  2  2  0  1  1  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  1  1  2  0  0  1  1     2  2  2  0  2  2  0  2  2  0  2  0
          ⋮              ⋮              ⋮           ⋱           ⋮              ⋮
          0  1  1  2  2  0  2  2  1  1  1  1  1     2  2  2  0  2  2  0  2  2  0  2  0
          1  2  0  2  2  0  2  2  1  1  1  0  0     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  …  1  1  2  1  1  1  1  1  1  1  2  1
          1  2  0  2  2  0  0  0  2  0  0  2  1     1  1  2  1  1  1  1  1  1  1  2  1
          0  2  0  2  2  0  1  1  1  1  1  1  1     2  2  2  1  2  2  0  1  1  1  2  1
          0  2  0  2  2  0  0  0  2  0  0  0  0     2  1  2  1  2  1  0  1  1  0  2  1
          1  2  0  2  2  0  0  0  2  0  0  1  1     2  1  2  1  2  2  0  1  1  1  2  0
          2  2  0  2  2  0  2  2  0  2  2  0  0  …  2  2  2  1  2  2  0  1  1  0  2  1
          1  1  1  2  2  0  2  2  0  2  2  0  0     2  2  2  0  2  2  0  2  2  0  2  0
          1  1  1  2  2  0  2  2  0  2  2  0  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  2  2  0  2  2  0  2  2  0  2  0
          0  2  0  2  2  1  2  2  0  2  2  0  2     2  2  2  0  2  2  0  2  2  0  2  0
```

```
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}:
          43954
          42510
          41620
          41950
          41555
          44410
          41935
          44108
          41740
          40731
          41195
          40872
          41360
              ⋮
          75568
          75817
          73034
          76451
          75520
          73448
          73760
          74787
          75503
          76024
          72868
          76325
```

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

```
Out[53]: 1000-element Array{Int64,1}:
          43954
          42510
          41620
          41950
          41555
          44410
          41935
          44108
          41740
          40731
          41195
          40872
          41360
              ⋮
          76367
          76132
          75508
          75846
          73523
          74518
          72920
          74574
          76397
          74936
          75755
          74604
```

```
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}:
 43954
 42510
 41620
 41950
 41555
 44410
 41935
 44108
 41740
 40731
 41195
 40872
 41360
     ⋮
 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   9.747  10.182
          40723  11.655  10.963
          40724  12.267  11.174
          40725   8.604   9.381
          40726  10.547   9.567
          40727  11.059   9.378
          40728   9.588   9.589
          40729   7.437   9.98
          40730  11.209  11.385
          40731  12.609  11.967
          40732   9.663   9.575
          40733   9.055  10.373
          40734  11.817  10.186
              ⋮
          88710  13.378  12.784
          88711   7.048  12.761
          88712  15.066  11.174
          88713  12.602  12.968
          88714  15.119  11.773
          88715  11.673  10.577
          88716  10.851  11.567
          88717  11.392  12.58
          88718  13.837  12.783
          88719  14.569  12.374
          88720  15.782  12.785
          88721  12.952  11.568
```

```
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
            5
            7
           16
           17
           21
           25
           27
           36
           37
           41
           45
           47
            ⋮
          156
          157
          161
          165
          167
          176
          177
          181
          185
          187
          196
          197
```

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

```
Out[83]: 150-element Array{Int64,1}:
            2
            3
            4
            6
            8
            9
           10
           11
           12
           13
           14
           15
           18
            ⋮
          186
          188
          189
          190
          191
          192
          193
          194
          195
          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}:
          2  2  0  2  2  0  0  0  2  0  0  2  1  …  2  1  1  1  1  2  0  1  1  0  2  1
          0  2  0  2  2  1  1  1  1  1  1  1  2     2  2  1  1  1  1  1  1  1  1  2  1
          2  2  0  2  2  0  2  2  0  2  2  0  0     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  2  2  0  2  2  0  2  2  0  2  0
          1  2  0  2  2  0  1  1  1  1  1  0  0     2  2  2  0  2  2  0  2  2  0  2  0
          1  1  1  2  2  2  2  2  0  2  2  0  2  …  2  2  1  1  1  1  0  1  1  0  1  1
          0  2  0  2  2  0  0  0  2  0  0  2  1     2  1  0  2  1  1  1  0  0  1  0  2
          0  1  1  2  2  0  1  1  1  1  1  1  1     2  2  1  1  1  1  0  1  1  0  1  1
          0  2  0  2  2  0  1  1  1  1  1  1  2     2  2  2  1  2  2  0  1  1  1  2  1
          0  2  0  2  2  0  1  1  1  1  1  1  2     2  1  1  2  2  2  0  0  0  2  2  1
          0  1  1  2  2  1  1  1  1  1  1  1  2  …  2  0  0  2  1  2  0  0  0  1  2  1
          0  1  1  1  1  0  1  1  1  1  1  0  1     2  1  2  1  1  2  0  1  1  0  1  1
          0  2  0  2  2  0  0  0  2  0  0  2  2     1  1  1  2  1  1  1  0  0  1  2  2
          ⋮              ⋮              ⋮           ⋱     ⋮              ⋮
          0  1  1  2  2  0  2  2  1  1  1  1  1     2  2  2  0  2  2  0  2  2  0  2  0
          1  2  0  2  2  0  2  2  1  1  1  0  0     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  …  1  1  2  1  1  1  1  1  1  1  2  1
          1  2  0  2  2  0  0  0  2  0  0  2  1     1  1  2  1  1  1  1  1  1  1  2  1
          0  2  0  2  2  0  1  1  1  1  1  1  1     2  2  2  1  2  2  0  1  1  1  2  1
          0  2  0  2  2  0  0  0  2  0  0  0  0     2  1  2  1  2  1  0  1  1  0  2  1
          1  2  0  2  2  0  0  0  2  0  0  1  1     2  1  2  1  2  2  0  1  1  1  2  0
          2  2  0  2  2  0  2  2  0  2  2  0  0  …  2  2  2  1  2  2  0  1  1  0  2  1
          1  1  1  2  2  0  2  2  0  2  2  0  0     2  2  2  0  2  2  0  2  2  0  2  0
          1  1  1  2  2  0  2  2  0  2  2  0  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  2  2  0  2  2  0  2  2  0  2  0
          0  2  0  2  2  1  2  2  0  2  2  0  2     2  2  2  0  2  2  0  2  2  0  2  0
```

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

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

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]:
```
0.14335193677822455
```

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

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

```
Out[100]: 0.37769037901198876
```

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

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

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

```
Out[102]: 50-element Array{Float64,1}:
           0.199275
           0.197664
           0.201626
           0.19931
           0.200331
           0.200121
           0.199984
           0.199516
           0.196013
           0.197651
           0.196305
           0.201727
           0.195979
           ⋮
           0.199445
           0.201647
           0.200778
           0.200637
           0.197307
           0.198344
           0.201007
           0.20235
           0.201203
           0.201717
           0.201105
           0.200554
```

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

Out[103]: 48000-element Array{Float64,1}:
           10.1903
           10.9991
           11.1879
            9.38933
            9.6025
            9.39864
            9.58662
           10.0096
           11.3836
           11.9852
            9.60953
           10.3868
           10.1988
            ⋮
           12.7962
           12.7997
           11.2059
           12.9901
           11.7877
           10.5959
           11.5861
           12.5972
           12.7807
           12.3811
           12.7803
           11.6177
```

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

Out[104]: 10.329442119521262
```

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

Out[105]: 10.9623971133817
```

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

Out[106]: 11.236842472409307
```

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

Out[107]: 11.50667808002454

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

Out[108]: 11.764331576348168

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

Out[109]: 12.036067916898244

In [110]: EAlphaG=onlyQTL*QTLo

Out[110]: 9000-element Array{Float64,1}:
           11.9852
           10.3895
           11.996
           12.391
           11.1904
           12.7936
           10.986
           12.5864
           12.3922
           11.7859
           12.5891
           11.8132
           11.5894
             ⋮
           12.7962
           12.7997
           11.2059
           12.9901
           11.7877
           10.5959
           11.5861
           12.5972
           12.7807
           12.3811
           12.7803
           11.6177
```

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

Out[111]: 12.015277860852247

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

Out[112]: 1.685835741330985

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

Out[113]: 11.61612355797032

In [114]: meanBreedingValueSelectedG0=meanEAlphaS0-meanEAlphaG0

Out[114]: 1.2866814384490581

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

Out[115]: 11.495982797447557

In [116]: meanBreedingValueSelectedG1=meanEAlphaS1-meanEAlphaG0

Out[116]: 1.1665406779262941

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

Out[117]: 11.772129870148065

In [118]: meanBreedingValueSelectedG2=meanEAlphaS2-meanEAlphaG0

Out[118]: 1.4426877506268028

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

Out[119]: 12.03631633507932

In [120]: meanBreedingValueSelectedG3=meanEAlphaS3-meanEAlphaG0

Out[120]: 1.7068742155580576
```

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

Out[121]: 12.324234501776015

In [122]: meanBreedingValueSelectedG4=meanEAlphaS4-meanEAlphaG0

Out[122]: 1.99479238225475 26
```

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

Out[123]: 12.036067916898244

In [124]: meanBreedingValueSelectedG5=meanEAlphaS5-meanEAlphaG0

Out[124]: 1.7066257973769812
```