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 = 0.2
# 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 = 0.2
XSim.init(numChr,numLoci,chrLength,geneFreq,mapPos,qtlMarker,qtlEffects,mutRate)
```

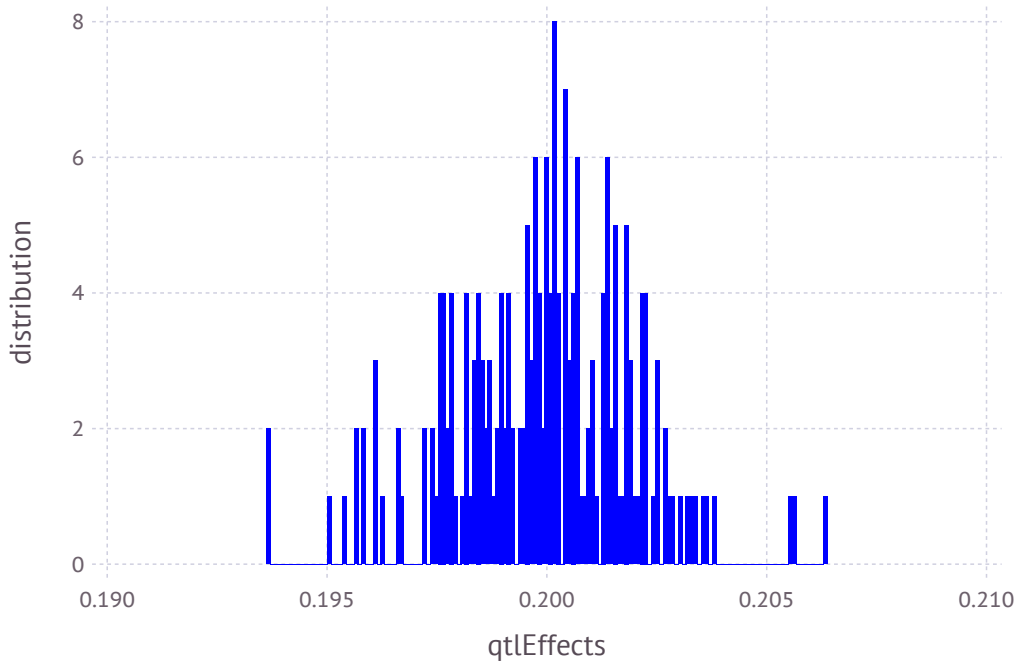
In [7]:
```
qtlEffects
```

Out[7]:
```
200-element Array{Float64,1}:
 0.201302
 0.199901
 0.196268
 0.197527
 0.201424
 0.19872
 0.200441
 0.202236
 0.200399
 0.200172
 0.199606
 0.203673
 0.199847
 ⋮
 0.197397
 0.199804
 0.201921
 0.199523
 0.201008
 0.197604
 0.201017
 0.203367
 0.202202
 0.20005
 0.19876
 0.20091
```

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]:
```
0.1998978885302878
```

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

Out[11]:
```
4.247585729703995e-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
```

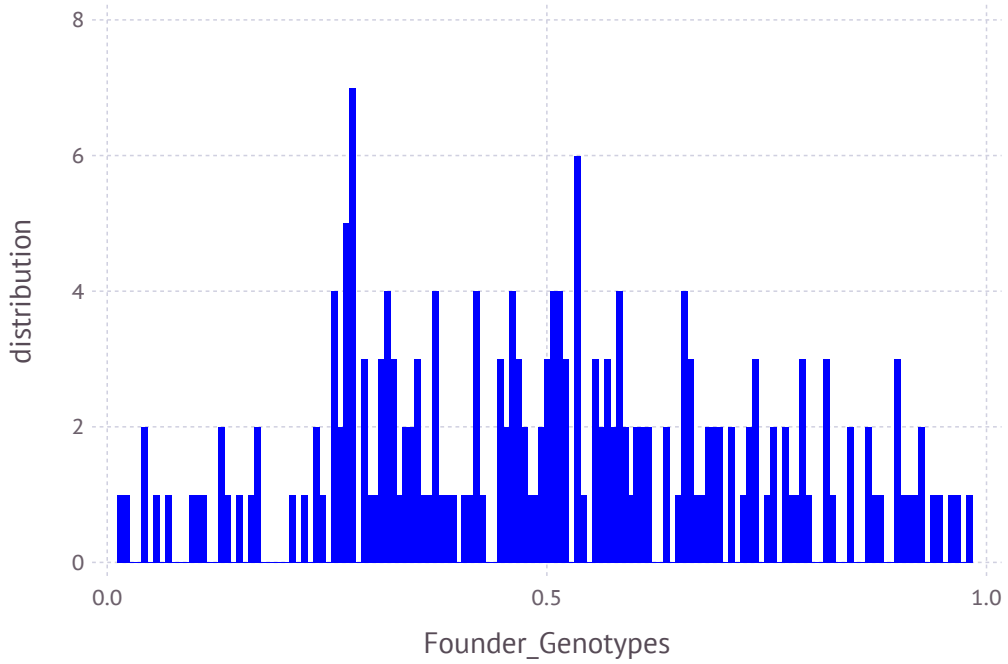
In [19]:
```
gSPSire = XSim.getOurGenotypes(popSP[1])
gSPDam = XSim.getOurGenotypes(popSP[2])
gSP = [gSPSire;gSPSire];
```

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

Out[20]:
```
1x200 Array{Float64,2}:
 0.058375  0.84375  0.290625  0.9475  …  0.37225  0.37325  0.901875  0.54375
```

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);
```

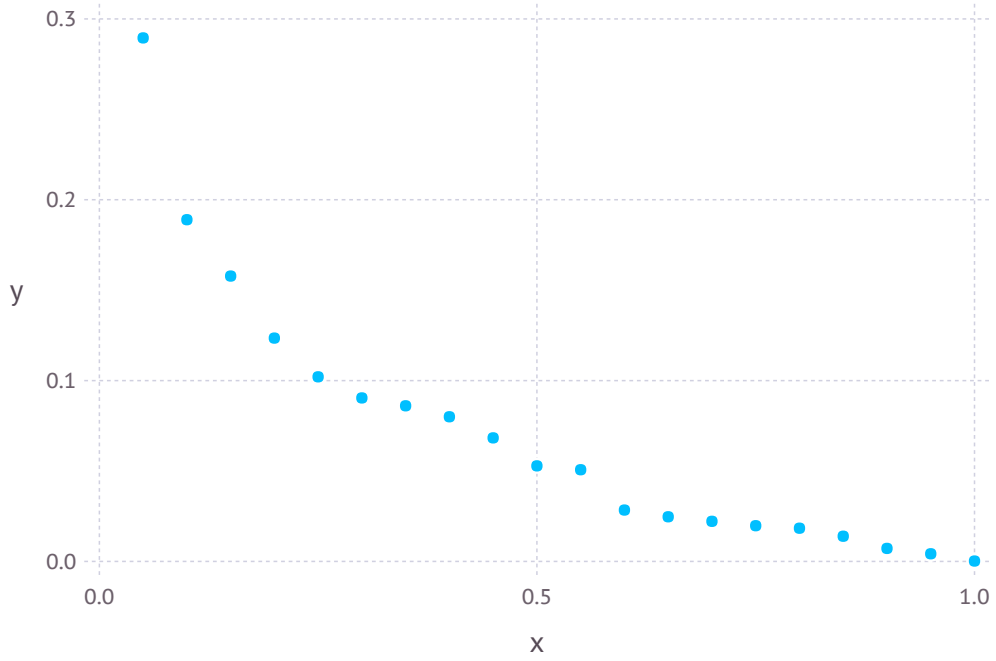
In [23]:
```
for i=1:(nRows-20)
LDMat[i,:] = corMat[i,(i+1):(i+20)].^2
end
```

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

Out[24]:
```
1x20 Array{Float64,2}:
 0.00024397  0.00425952  0.00721361  …  0.157819  0.18898  0.289519
```

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]:
```
9.911970027040221
```

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

Out[31]:
```
0.7114488232902924
```

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

Out[32]:
```
0.7114488232902924
```

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

Out[33]:
```
0.7114488232902924
```

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]: 12.460215463959909

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

Out[36]: 12.463828066287023

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

Out[37]: 0.5744328759044227

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

Out[38]: 0.5375654070338648

Pedigree: All animals

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

Out[39]:
```
48000x3 Array{Int64,2}:
 40722  35261  39163
 40723  35009  37084
 40724  36234  39895
 40725  36692  39041
 40726  33221  38954
 40727  33227  38365
 40728  34763  37723
 40729  35669  40562
 40730  34365  36823
 40731  35729  39211
 40732  35882  37893
 40733  33281  38466
 40734  34712  36930
     ⋮
 88710  74659  79473
 88711  76508  77319
 88712  76349  80693
 88713  74076  80217
 88714  76519  80112
 88715  76524  77420
 88716  73812  79810
 88717  72853  78732
 88718  74090  77996
 88719  73533  80398
 88720  73208  79232
 88721  76002  80337
```

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	1	1	2	0	…	2	1	1	1	1	2	0	1	1	0	2	1
40723	0	2	1	2	1	0	0	0	2	0		2	0	0	2	1	2	0	0	0	1	2	1
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		1	1	2	1	1	1	1	1	1	1	2	1
40726	0	1	2	1	0	0	1	1	1	1		2	2	2	0	2	2	0	2	2	0	2	0
40727	1	2	0	2	2	1	2	2	1	1	…	2	2	2	0	2	2	0	2	2	0	2	0
40728	0	2	1	2	1	0	1	1	2	0		1	0	2	2	2	2	1	0	0	0	1	1
40729	1	2	1	2	1	0	1	1	1	1		2	2	1	1	1	1	0	1	1	0	1	1
40730	1	1	1	1	1	0	1	1	1	1		2	1	1	1	1	2	1	1	1	1	1	1
40731	0	1	1	2	2	0	1	1	1	1		1	1	2	1	1	1	1	1	1	1	2	1
40732	0	2	1	2	1	0	1	1	2	0	…	1	2	2	1	1	2	0	1	1	1	1	1
40733	0	1	1	2	2	1	1	1	1	1		0	0	2	2	0	0	2	0	0	2	2	2
40734	0	2	0	2	2	0	0	0	2	0		2	1	1	2	0	2	0	0	0	0	1	2
⋮				⋮					⋮	⋱		⋮					⋮					⋮	
88710	0	2	1	2	1	0	0	0	2	0		2	1	0	2	1	1	1	0	0	1	2	2
88711	0	2	0	2	2	0	0	0	2	0		2	2	2	0	2	2	0	1	1	1	2	1
88712	0	2	2	2	0	0	0	0	2	0	…	2	2	1	1	2	2	0	1	1	0	2	1
88713	0	2	0	2	2	0	0	0	2	0		2	2	2	1	2	2	0	1	1	1	1	1
88714	0	0	2	1	1	0	1	1	1	1		2	1	2	1	2	1	0	1	1	0	2	1
88715	0	2	1	2	1	0	0	0	2	0		2	2	2	1	1	1	1	1	1	1	2	1
88716	0	1	2	2	1	0	1	1	1	1		2	0	0	2	2	2	1	1	1	1	2	1
88717	0	2	1	2	1	0	0	0	2	0	…	2	0	0	2	1	2	1	0	0	2	1	1
88718	0	2	1	2	1	0	0	0	2	0		2	1	1	2	1	2	0	0	0	1	1	2
88719	1	2	0	2	2	0	2	2	1	1		2	2	0	2	2	2	0	0	0	0	2	2
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	1	2	1	0	0	0	2	0		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  2  0  2  2  0  1  1  2  0  0  2  2  …  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     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     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     1  1  2  1  1  1  1  1  1  1  2  1
 0  1  2  1  0  0  1  1  1  1  1  1  1     2  2  2  0  2  2  0  2  2  0  2  0
 1  2  0  2  2  1  2  2  1  1  1  1  2  …  2  2  2  0  2  2  0  2  2  0  2  0
 0  2  1  2  1  0  1  1  2  0  0  2  1     1  0  2  2  2  2  1  0  0  0  1  1
 1  2  1  2  1  0  1  1  1  1  1  1  0     2  2  1  1  1  1  0  1  1  0  1  1
 1  1  1  1  1  0  1  1  1  1  1  1  1     2  1  1  1  1  2  1  1  1  1  1  1
 0  1  1  2  2  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  1  1  2  0  0  2  1  …  1  2  2  1  1  2  0  1  1  1  1  1
 0  1  1  2  2  1  1  1  1  1  1  0  2     0  0  2  2  0  0  2  0  0  2  2  2
 0  2  0  2  2  0  0  0  2  0  0  0  0     2  1  1  2  0  2  0  0  0  0  1  2
 ⋮              ⋮              ⋮        ⋱        ⋮              ⋮
 0  2  1  2  1  0  0  0  2  0  0  2  1     2  1  0  2  1  1  1  0  0  1  2  2
 0  2  0  2  2  0  0  0  2  0  0  2  2     2  2  2  0  2  2  0  1  1  1  2  1
 0  2  2  2  0  0  0  0  2  0  0  2  0  …  2  2  1  1  2  2  0  1  1  0  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  1  1
 0  0  2  1  1  0  1  1  1  1  1  0  0     2  1  2  1  2  1  0  1  1  0  2  1
 0  2  1  2  1  0  0  0  2  0  0  2  1     2  2  2  1  1  1  1  1  1  1  2  1
 0  1  2  2  1  0  1  1  1  1  1  1  0     2  0  0  2  2  2  1  1  1  1  2  1
 0  2  1  2  1  0  0  0  2  0  0  2  1  …  2  0  0  2  1  2  1  0  0  2  1  1
 0  2  1  2  1  0  0  0  2  0  0  1  0     2  1  1  2  1  2  0  0  0  1  1  2
 1  2  0  2  2  0  2  2  1  1  1  1  1     2  2  0  2  2  2  0  0  0  0  2  2
 0  2  0  2  2  0  0  0  2  0  0  2  2     2  2  2  0  2  2  0  2  2  0  2  0
 0  2  1  2  1  0  0  0  2  0  0  1  1     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}:
 41648
 43895
 40756
 42768
 43995
 44110
 42453
 40847
 42168
 42690
 43745
 43117
 41563
     ⋮
 74659
 76508
 76349
 74076
 76519
 76524
 73812
 72853
 74090
 73533
 73208
 76002
```

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

Out[53]:
```
1000-element Array{Int64,1}:
 41648
 43895
 40756
 42768
 43995
 44110
 42453
 40847
 42168
 42690
 43745
 43117
 41563
     ⋮
 74090
 75570
 75985
 76130
 74500
 73637
 73844
 74843
 76438
 74222
 76147
 75898
```

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}:
 41648
 43895
 40756
 42768
 43995
 44110
 42453
 40847
 42168
 42690
 43745
 43117
 41563
     ⋮
 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   8.35    8.928
 40723  10.276  10.124
 40724  12.078   9.341
 40725  10.019  10.122
 40726   9.699   9.925
 40727   8.347   9.333
 40728  10.618   8.539
 40729  10.083   9.93
 40730   9.54   10.131
 40731  11.146   9.534
 40732   8.028   8.543
 40733  10.283   9.738
 40734   8.098   9.124
     ⋮
 88710  12.053  12.519
 88711  12.677  13.31
 88712  12.794  13.318
 88713  13.746  12.917
 88714  13.782  13.711
 88715  13.407  13.317
 88716  12.21   12.898
 88717  15.926  13.909
 88718  14.359  12.92
 88719  12.185  12.911
 88720   9.502  10.921
 88721  13.008  13.691
```

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}:
   2
   3
  13
  17
  20
  22
  23
  33
  37
  40
  42
  43
  53
   ⋮
 157
 160
 162
 163
 173
 177
 180
 182
 183
 193
 197
 200
```

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

Out[83]:
```
150-element Array{Int64,1}:
   1
   4
   5
   6
   7
   8
   9
  10
  11
  12
  14
  15
  16
   ⋮
 186
 187
 188
 189
 190
 191
 192
 194
 195
 196
 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}:

0	2	0	2	2	0	0	0	2	0	0	0	0	…	2	1	2	2	2	2	0	0	0	0	2	1
0	1	1	1	1	0	1	1	1	1	1	1	2		2	1	2	1	2	2	0	1	1	1	1	1
0	2	0	2	2	0	0	0	2	0	0	1	1		1	0	2	1	2	2	0	2	2	0	2	0
0	1	1	2	2	0	0	0	2	0	0	2	2		2	0	0	2	1	2	0	0	0	1	1	2
0	2	2	2	0	0	0	0	2	0	0	2	1		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	…	1	1	1	2	0	0	1	0	0	1	1	2
1	2	0	2	2	0	1	1	1	1	1	1	1		2	2	2	0	2	2	0	2	2	0	2	0
0	2	1	2	1	0	0	0	2	0	0	2	1		2	1	1	1	2	2	0	1	1	1	1	1
0	2	1	2	1	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	0	0	2	0	0	1	1		1	0	1	2	0	1	1	0	0	1	2	2
0	2	1	2	1	0	0	0	2	0	0	1	0	…	1	1	2	1	1	1	1	0	0	2	1	2
0	1	1	2	2	0	1	1	1	1	1	1	2		1	0	2	2	1	1	1	0	0	2	1	2
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	1	2	1	0	0	0	2	0	0	2	1		2	1	0	2	1	1	1	0	0	1	2	2
0	2	0	2	2	0	0	0	2	0	0	2	2		2	2	2	0	2	2	0	1	1	1	2	1
0	2	2	2	0	0	0	0	2	0	0	2	0	…	2	2	1	1	2	2	0	1	1	0	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	1	1
0	0	2	1	1	0	1	1	1	1	1	0	0		2	1	2	1	2	1	0	1	1	0	2	1
0	2	1	2	1	0	0	0	2	0	0	2	1		2	2	2	1	1	1	1	1	1	1	2	1
0	1	2	2	1	0	1	1	1	1	1	1	0		2	0	0	2	2	2	1	1	1	1	2	1
0	2	1	2	1	0	0	0	2	0	0	2	1	…	2	0	0	2	1	2	1	0	0	2	1	1
0	2	1	2	1	0	0	0	2	0	0	1	0		2	1	1	2	1	2	0	0	0	1	1	2
1	2	0	2	2	0	2	2	1	1	1	1	1		2	2	0	2	2	2	0	0	0	0	2	2
0	2	0	2	2	0	0	0	2	0	0	2	2		2	2	2	0	2	2	0	2	2	0	2	0
0	2	1	2	1	0	0	0	2	0	0	1	1		2	2	2	0	2	2	0	2	2	0	2	0

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

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

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]: 0.6451491481517805

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

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

Out[100]:
```
0.8042283219853588
```

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

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

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

Out[102]:
```
50-element Array{Float64,1}:
 0.199901
 0.196268
 0.199847
 0.197537
 0.199608
 0.199444
 0.199561
 0.200178
 0.197364
 0.198707
 0.200904
 0.199417
 0.197738
 ⋮
 0.197246
 0.200466
 0.202135
 0.201666
 0.199711
 0.202511
 0.200288
 0.201258
 0.197697
 0.201008
 0.202202
 0.20091
```

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

Out[103]:
```
48000-element Array{Float64,1}:
  8.99949
 10.2057
  9.40921
 10.2112
 10.007
  9.38798
  8.59958
  9.98785
 10.2055
  9.5979
  8.60262
  9.79331
  9.20962
  ⋮
 12.6112
 13.4149
 13.4064
 13.0099
 13.7844
 13.406
 13.0017
 14.0108
 13.0029
 13.0158
 11.0058
 13.8067
```

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

Out[104]:
```
9.985862350527333
```

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

Out[105]:
```
10.52357783771815
```

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

Out[106]:
```
11.038589197645845
```

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

Out[107]:
```
11.614958504701098
```

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

Out[108]:
```
12.085686763133857
```

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

Out[109]:
```
12.550473961685555
```

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

Out[110]:
```
9000-element Array{Float64,1}:
 10.2062
  9.99854
 11.2017
 11.1976
 10.2064
 10.8158
 10.599
 10.8068
 10.4114
 10.4091
 10.4032
 11.803
 11.408
  ⋮
 12.6112
 13.4149
 13.4064
 13.0099
 13.7844
 13.406
 13.0017
 14.0108
 13.0029
 13.0158
 11.0058
 13.8067
```

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

Out[111]: 12.497763679201803

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

Out[112]: 2.5119013286744707

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

Out[113]: 11.077555366185056

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

Out[114]: 1.0916930156577234

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

Out[115]: 11.533429031609339

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

Out[116]: 1.5475666810820066

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

Out[117]: 12.185741725112921

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

Out[118]: 2.1998793745855885

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

Out[119]: 12.558449639539019

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

Out[120]: 2.572587289011686

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

Out[121]: 13.025231334212727

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

Out[122]: 3.039368983685394

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

Out[123]: 12.550473961685555

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

Out[124]: 2.564611611158222