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)
```

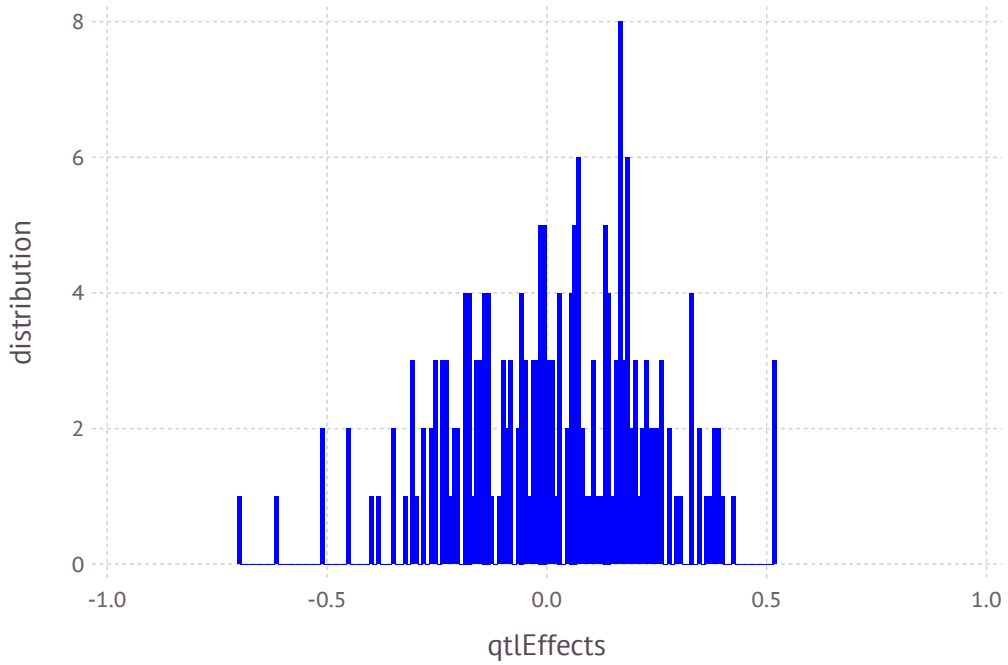
In [7]:
```
qtlEffects
```

Out[7]:
```
200-element Array{Float64,1}:
 -0.0144712
  0.350156
  0.377572
 -0.230113
  0.198044
  0.000490792
  0.0843271
 -0.12432
  0.350805
 -0.301174
  0.190045
 -0.166049
 -0.351479
  ⋮
  0.19564
  0.183409
 -0.0500917
 -0.143534
 -0.0224212
  0.0066607
  0.0705544
 -0.0554408
 -0.130899
 -0.450462
  0.032694
 -0.0286977
```

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]: 0.017378962036454206

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

Out[11]: 0.04795885961480579

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
```

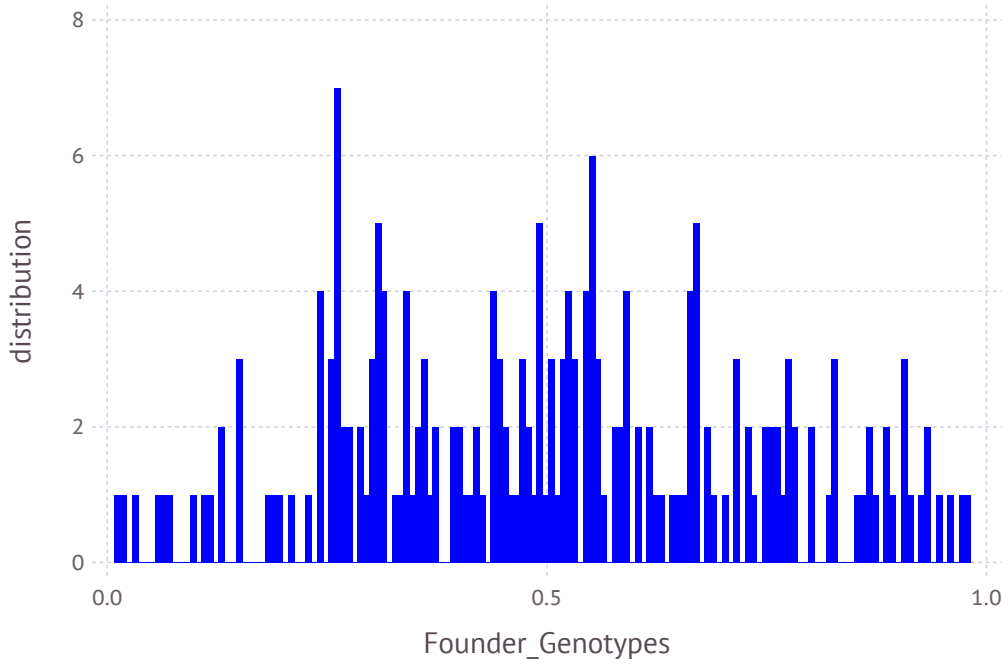
In [19]:
```
gSPSire = XSim.getOurGenotypes(popSP[1])
gSPDam = XSim.getOurGenotypes(popSP[2])
gSP = [gSPSire;gSPSire];
```

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

Out[20]:
```
1x200 Array{Float64,2}:
 0.067375  0.851  0.27  0.9465  0.826625  …  0.39025  0.896625  0.549875
```

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);
```

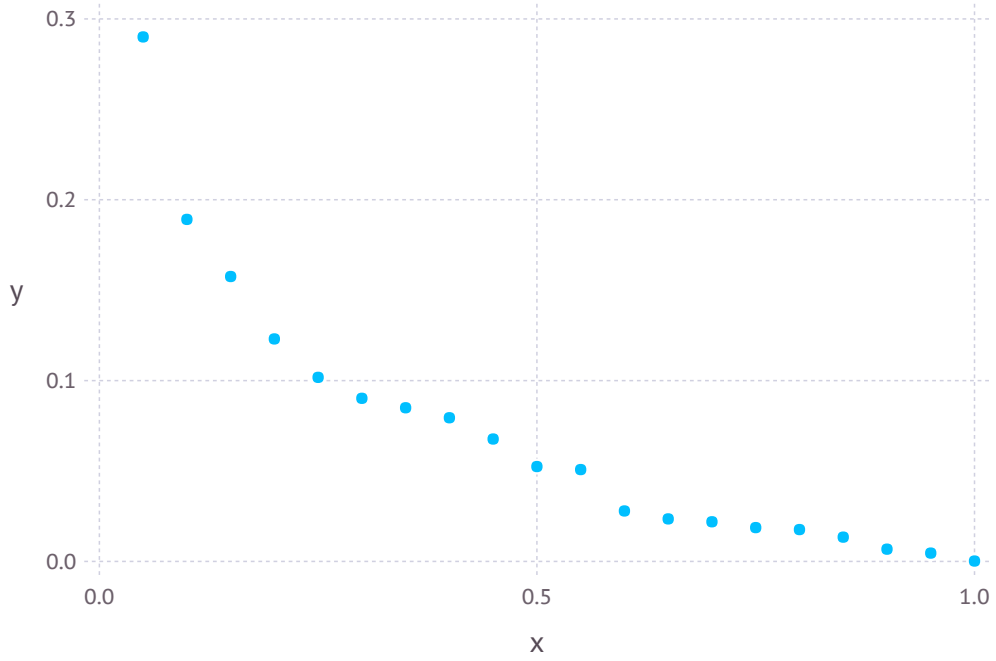
In [23]:
```
for i=1:(nRows-20)
LDMat[i,:] = corMat[i,(i+1):(i+20)].^2
end
```

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

Out[24]:
```
1x20 Array{Float64,2}:
 0.000248527  0.00460797  0.00679246  …  0.157572  0.189158  0.290056
```

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]:
```
3.299662540179286
```

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

Out[31]:
```
0.2625708028692303
```

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

Out[32]:
```
0.2625708028692303
```

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

Out[33]:
```
0.2625708028692303
```

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]: 4.741221742032125

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

Out[36]: 4.739492316774075

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

Out[37]: 0.16128297744111733

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

Out[38]: 0.16805464558517075

Pedigree: All animals

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

Out[39]:
```
48000x3 Array{Int64,2}:
 40722  34405  38443
 40723  32883  39353
 40724  35525  39973
 40725  33844  37333
 40726  36067  40117
 40727  34351  39208
 40728  34211  37679
 40729  36342  37483
 40730  33640  37944
 40731  36358  39567
 40732  33852  38796
 40733  36706  37004
 40734  33721  39187
     ⋮
 88710  75649  77469
 88711  73874  79042
 88712  75985  77859
 88713  74353  80271
 88714  75114  77818
 88715  73043  80176
 88716  74291  80031
 88717  76397  80287
 88718  75730  78140
 88719  76136  80273
 88720  76647  78382
 88721  73028  79119
```

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	2	2	1	1	1	1	1	…	2	1	2	1	2	1	1	1	1	0	1	1
40723	0	1	1	2	2	0	1	1	1	1		2	1	1	1	1	2	0	1	1	0	2	1
40724	0	2	1	2	1	0	0	0	2	0		2	2	2	0	2	2	0	2	2	0	2	0
40725	0	2	0	2	2	0	0	0	2	0		2	1	1	1	1	2	0	1	1	0	2	1
40726	1	1	1	2	2	2	2	2	0	2		2	1	2	1	1	1	0	1	1	0	2	1
40727	0	2	0	2	2	0	0	0	2	0	…	2	1	1	1	2	2	0	1	1	1	2	0
40728	0	2	0	2	2	0	0	0	2	0		2	2	2	0	2	2	0	2	2	0	2	0
40729	1	1	1	1	1	0	1	1	1	1		2	1	2	1	1	1	0	1	1	0	2	1
40730	0	2	0	2	2	1	1	1	1	1		1	0	2	1	1	1	1	1	1	1	2	1
40731	1	2	0	2	2	0	1	1	2	0		2	1	1	2	2	2	0	0	0	2	2	0
40732	0	1	1	2	2	0	1	1	1	1	…	2	1	2	1	2	2	0	1	1	0	2	1
40733	1	2	0	2	2	0	1	1	1	1		1	0	1	1	2	2	0	1	1	1	2	0
40734	0	2	0	2	2	0	0	0	2	0		1	1	2	2	1	1	1	0	0	1	2	2
⋮			⋮				⋮	⋱		⋮					⋮						⋮	
88710	0	1	1	2	2	1	2	2	0	2		1	0	1	2	0	1	1	0	0	1	2	2
88711	1	2	0	2	2	1	2	2	0	2		2	2	1	1	1	1	1	1	1	1	2	1
88712	0	2	0	2	2	0	1	1	2	0	…	2	2	2	0	2	2	0	2	2	0	2	1
88713	0	1	1	2	2	1	1	1	1	1		2	1	1	1	1	2	0	1	1	0	2	1
88714	0	2	0	2	2	1	1	1	1	1		2	1	1	1	1	2	0	1	1	0	2	1
88715	0	2	0	2	2	0	1	1	2	0		2	2	1	1	2	2	0	1	1	1	2	1
88716	0	1	1	2	2	0	1	1	1	1		1	1	2	1	1	1	1	1	1	1	2	1
88717	0	2	0	2	2	1	2	2	1	1	…	2	2	2	0	2	2	0	2	2	0	2	0
88718	0	1	1	2	2	1	1	1	1	1		2	1	1	1	1	2	0	1	1	0	2	1
88719	0	1	1	2	2	0	1	1	1	1		2	0	0	2	0	1	1	0	0	1	2	2
88720	0	2	0	2	2	0	2	2	2	0		2	0	2	2	1	0	1	0	0	1	2	2
88721	0	2	0	2	2	1	1	1	1	1		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  2  2  1  1  1  1  1  1  1  2  …  2  1  2  1  2  1  1  1  1  0  1  1
 0  1  1  2  2  0  1  1  1  1  1  1  1     2  1  1  1  1  2  0  1  1  0  2  1
 0  2  1  2  1  0  0  0  2  0  0  1  0     2  2  2  0  2  2  0  2  2  0  2  0
 0  2  0  2  2  0  0  0  2  0  0  0  0     2  1  1  1  1  2  0  1  1  0  2  1
 1  1  1  2  2  2  2  2  0  2  2  0  2     2  1  2  1  1  1  0  1  1  0  2  1
 0  2  0  2  2  0  0  0  2  0  0  2  2  …  2  1  1  1  2  2  0  1  1  1  2  0
 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
 1  1  1  1  1  0  1  1  1  1  1  0  0     2  1  2  1  1  1  0  1  1  0  2  1
 0  2  0  2  2  1  1  1  1  1  1  1  2     1  0  2  1  1  1  1  1  1  1  2  1
 1  2  0  2  2  0  1  1  2  0  0  1  1     2  1  1  2  2  2  0  0  0  2  2  0
 0  1  1  2  2  0  1  1  1  1  1  1  1  …  2  1  2  1  2  2  0  1  1  0  2  1
 1  2  0  2  2  0  1  1  1  1  1  0  0     1  0  1  1  2  2  0  1  1  1  2  0
 0  2  0  2  2  0  0  0  2  0  0  1  1     1  1  2  2  1  1  1  0  0  1  2  2
 ⋮              ⋮              ⋮        ⋱        ⋮              ⋮
 0  1  1  2  2  1  2  2  0  2  2  0  1     1  0  1  2  0  1  1  0  0  1  2  2
 1  2  0  2  2  1  2  2  0  2  2  0  0     2  2  1  1  1  1  1  1  1  1  2  1
 0  2  0  2  2  0  1  1  2  0  0  2  2  …  2  2  2  0  2  2  0  2  2  0  2  1
 0  1  1  2  2  1  1  1  1  1  1  1  2     2  1  1  1  1  2  0  1  1  0  2  1
 0  2  0  2  2  1  1  1  1  1  1  1  2     2  1  1  1  1  2  0  1  1  0  2  1
 0  2  0  2  2  0  1  1  2  0  0  2  2     2  2  1  1  2  2  0  1  1  1  2  1
 0  1  1  2  2  0  1  1  1  1  1  1  1     1  1  2  1  1  1  1  1  1  1  2  1
 0  2  0  2  2  1  2  2  1  1  0  1  2  …  2  2  2  0  2  2  0  2  2  0  2  0
 0  1  1  2  2  1  1  1  1  1  1  1  2     2  1  1  1  1  2  0  1  1  0  2  1
 0  1  1  2  2  0  1  1  1  1  1  1  1     2  0  0  2  0  1  1  0  0  1  2  2
 0  2  0  2  2  0  2  2  2  0  0  2  2     2  0  2  2  1  0  1  0  0  1  2  2
 0  2  0  2  2  1  1  1  1  1  1  1  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}:
 43333
 44209
 43827
 41920
 41880
 40728
 44592
 44495
 41096
 43424
 40756
 40935
 40945
     ⋮
 75649
 73874
 75985
 74353
 75114
 73043
 74291
 76397
 75730
 76136
 76647
 73028
```

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

Out[53]:
```
1000-element Array{Int64,1}:
 43333
 44209
 43827
 41920
 41880
 40728
 44592
 44495
 41096
 43424
 40756
 40935
 40945
     ⋮
 73296
 74617
 76227
 75014
 73184
 74579
 74397
 74283
 74622
 76385
 76667
 74825
```

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}:
 43333
 44209
 43827
 41920
 41880
 40728
 44592
 44495
 41096
 43424
 40756
 40935
 40945
     ⋮
 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  2.439  3.415
 40723  3.896  3.78
 40724  5.68   3.503
 40725  3.308  3.331
 40726  2.042  3.454
 40727  3.235  4.085
 40728  6.358  3.927
 40729  4.372  3.359
 40730  2.843  3.2
 40731  2.817  2.302
 40732  3.254  4.756
 40733  3.056  3.547
 40734  3.779  3.068
     ⋮
 88710  4.976  4.662
 88711  4.656  4.615
 88712  5.657  5.108
 88713  5.095  5.018
 88714  4.494  4.835
 88715  4.653  5.042
 88716  4.991  4.481
 88717  4.948  5.416
 88718  5.955  5.221
 88719  5.377  5.248
 88720  4.815  6.103
 88721  4.705  5.521
```

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}:
   6
   7
  15
  16
  20
  26
  27
  35
  36
  40
  46
  47
  55
   ⋮
 156
 160
 166
 167
 175
 176
 180
 186
 187
 195
 196
 200
```

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

Out[83]:
```
150-element Array{Int64,1}:
   1
   2
   3
   4
   5
   8
   9
  10
  11
  12
  13
  14
  17
   ⋮
 184
 185
 188
 189
 190
 191
 192
 193
 194
 197
 198
 199
```

Genotype codes: 0, 1, 2

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

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

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

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

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

Out[86]:
```
9000x50 Array{Int64,2}:
 0  0  2  0  1  2  2  1  1  0  0  1  1  …  1  0  1  1  0  0  1  2  2  0  2  0
 0  0  0  2  2  0  0  1  1  1  0  0  2     0  2  2  1  1  0  1  2  2  0  2  0
 0  2  0  2  2  0  1  2  2  2  1  0  1     0  1  2  1  1  0  1  0  1  0  1  0
 0  1  1  1  0  1  2  0  2  1  1  1  0     1  1  0  0  0  0  1  1  1  0  0  2
 0  1  1  1  1  0  1  2  2  2  1  0  0     1  1  1  0  0  1  2  1  1  0  0  2
 1  0  1  1  1  0  1  2  1  2  1  1  1  …  1  1  1  0  0  1  1  1  1  0  0  2
 0  0  2  0  1  0  0  2  1  2  2  0  0     0  1  2  1  0  1  1  1  1  1  1  1
 1  2  0  2  2  0  1  0  2  1  1  0  1     0  0  2  0  1  1  2  1  1  0  1  1
 0  0  1  1  2  0  1  1  1  2  0  1  0     1  1  1  0  0  0  2  0  0  0  0  2
 0  0  1  1  0  0  1  1  2  1  1  1  2     1  1  2  0  1  1  1  1  1  0  0  2
 0  0  1  1  1  0  1  1  1  1  2  0  1  …  1  1  1  1  1  2  1  2  2  0  1  1
 0  0  1  1  2  1  1  1  2  0  1  1  0     2  0  1  1  0  0  1  2  2  0  2  0
 1  1  0  2  2  0  2  0  2  2  0  2  0     1  2  0  0  1  1  2  2  2  0  2  0
 ⋮              ⋮              ⋮          ⋱        ⋮              ⋮
 1  2  0  2  1  0  2  0  2  1  1  1  0     1  0  1  2  2  2  2  0  0  1  0  2
 1  2  1  2  2  0  2  1  2  2  0  2  0     0  1  2  1  0  0  2  2  1  1  1  1
 0  1  0  2  2  1  2  1  2  1  1  1  0  …  2  0  1  1  1  2  1  2  2  0  2  1
 1  1  1  1  2  0  2  0  2  2  1  1  1     0  1  2  0  1  1  2  1  1  0  1  1
 1  1  0  2  2  1  2  0  2  2  1  1  1     1  0  2  0  0  0  1  2  2  0  1  1
 0  1  0  2  2  0  0  0  2  0  0  2  0     1  1  1  0  1  2  1  2  2  0  1  1
 0  1  0  2  2  2  2  1  2  1  0  2  0     1  1  0  0  0  0  2  1  1  1  1  1
 1  2  0  2  2  0  2  1  2  2  1  0  1  …  0  2  1  2  1  0  2  2  2  0  2  0
 1  1  1  1  2  0  2  0  2  2  1  1  1     1  1  1  0  0  1  1  1  1  0  1  1
 0  1  0  2  2  0  2  1  1  2  0  1  1     0  2  0  1  1  1  2  0  1  1  0  2
 0  2  0  2  2  0  2  0  2  2  0  2  1     0  2  1  2  1  0  2  1  2  1  0  2
 1  1  0  2  2  0  1  1  2  2  1  1  0     1  1  0  2  1  1  2  0  2  0  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]: 0.5443636500530624

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

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

Out[100]: 0.7412096198509552

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

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

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

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

Out[102]:
```
50-element Array{Float64,1}:
  0.000490792
  0.0843271
 -0.0807867
  0.144925
  0.185057
  0.056694
 -0.175274
  0.118512
 -0.183091
 -0.131718
  0.174152
 -0.27686
  0.220511
  ⋮
  0.16582
 -0.174809
 -0.0174457
  0.259564
  0.162567
  0.0300632
  0.132084
 -0.178685
  0.261782
  0.0705544
 -0.0554408
 -0.0286977
```

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

Out[103]:
```
48000-element Array{Float64,1}:
 1.6286
 1.82977
 1.97805
 0.863383
 0.726938
 1.47529
 3.02457
 2.03787
 1.00802
 2.29651
 2.63479
 2.38198
 1.43541
 ⋮
 2.61762
 0.514167
 2.35715
 2.2206
 1.98622
 2.68154
 0.83349
 1.92848
 1.81172
 3.18109
 2.75249
 2.24085
```

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

Out[104]:
```
1.6485743982253853
```

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

Out[105]:
```
1.6916682407080206
```

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

Out[106]:
```
1.757569633940897
```

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

```
Out[107]: 1.8284389251308713
```

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

```
Out[108]: 1.954090004768203
```

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

```
Out[109]: 1.9944042238842945
```

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

```
Out[110]: 9000-element Array{Float64,1}:
           1.97805
           3.02457
           2.92569
           0.484238
           1.88912
           1.72484
           1.38004
           2.26103
           1.13942
           0.7419
           2.88057
           2.33973
           0.26338
           ⋮
           2.61762
           0.514167
           2.35715
           2.2206
           1.98622
           2.68154
           0.83349
           1.92848
           1.81172
           3.18109
           2.75249
           2.24085
```

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

Out[111]: 1.9855465693804768

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

Out[112]: 0.3369721711550915

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

Out[113]: 1.7389700192365802

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

Out[114]: 0.09039562101119492

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

Out[115]: 1.8037777963797847

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

Out[116]: 0.15520339815439943

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

Out[117]: 1.8964192990861637

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

Out[118]: 0.24784490086077837

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

Out[119]: 2.0976802038721045

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

Out[120]: 0.4491058056467192

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

Out[121]: 2.036579348175054

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

Out[122]: 0.3880049499496685

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

Out[123]: 1.9944042238842945

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

Out[124]: 0.3458298256589092