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; 200 loci per chromosome = > 2000 Loci (500 QTL & 1500 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 = 0.1
numLoci   = 200
nQTL      = 50
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)
```

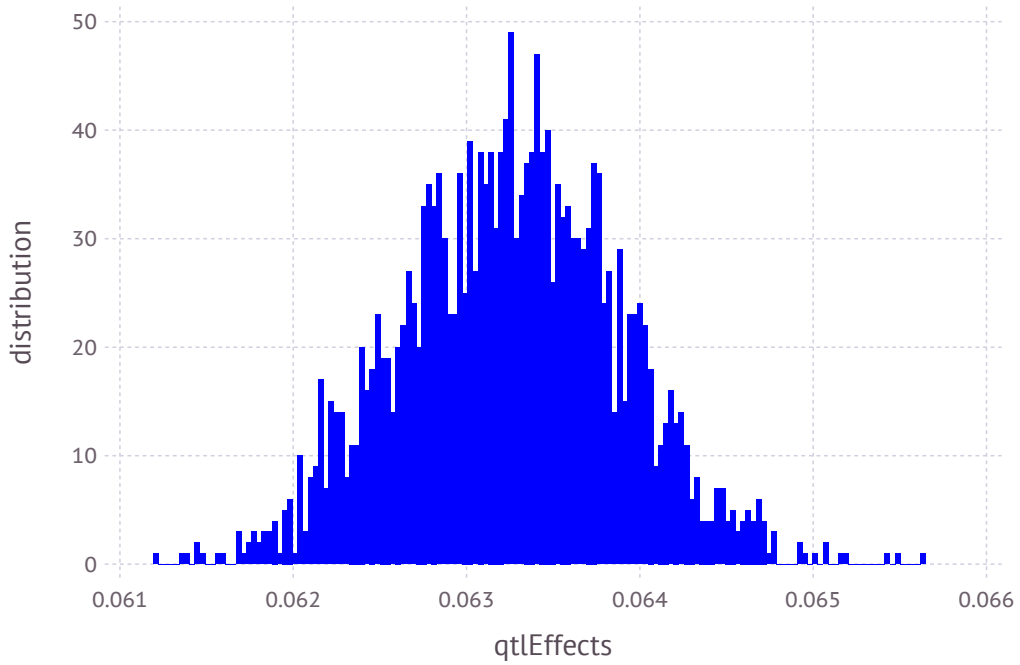
In [7]:
```
qtlEffects
```

Out[7]:
```
2000-element Array{Float64,1}:
 0.0638393
 0.0630802
 0.0644439
 0.0632514
 0.0636745
 0.0628089
 0.0624913
 0.0642954
 0.0638382
 0.0629657
 0.0637549
 0.062892
 0.0623429
 ⋮
 0.06284
 0.063602
 0.0631572
 0.0623926
 0.0637483
 0.0636036
 0.0630494
 0.0640627
 0.063561
 0.0618217
 0.0637116
 0.0617319
```

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]:
```
0.06325492439875419
```

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

Out[11]:
```
3.9610422411067817e-7
```

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/ch1to10.2000SNP.group1")
dams  = XSim.sampleFounders("/home/nicole/Jupyter/HERdata/data/Clean/newdata/ch1to10.2000SNP.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
```

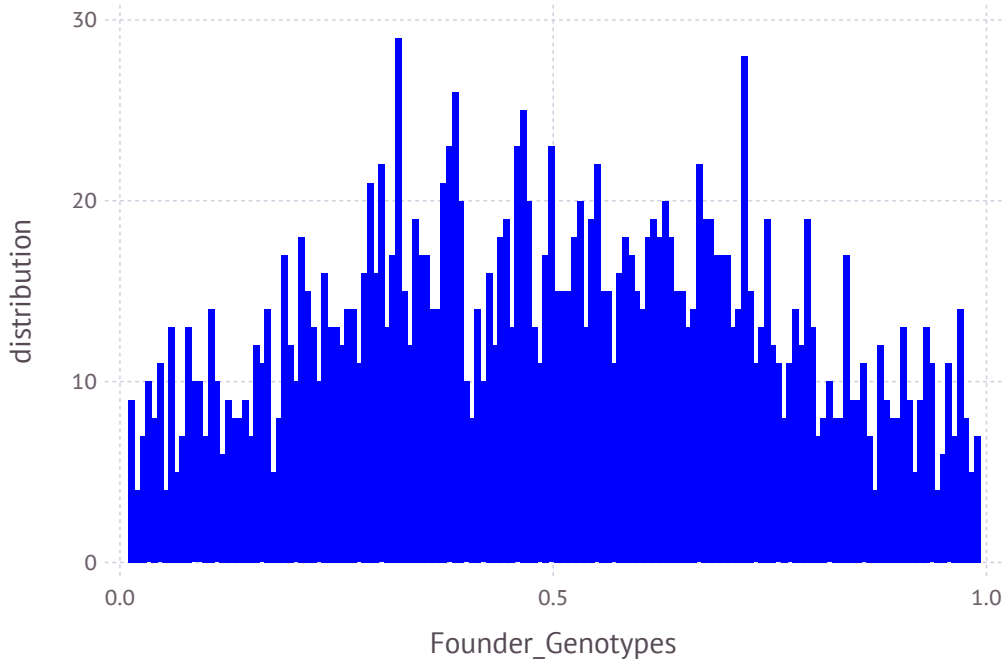
In [19]:
```
gSPSire = XSim.getOurGenotypes(popSP[1])
gSPDam = XSim.getOurGenotypes(popSP[2])
gSP = [gSPSire;gSPSire];
```

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

Out[20]:
```
1x2000 Array{Float64,2}:
 0.069125  0.847  0.28625  0.94925  …  0.2825  0.35425  0.44  0.272375
```

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);
```

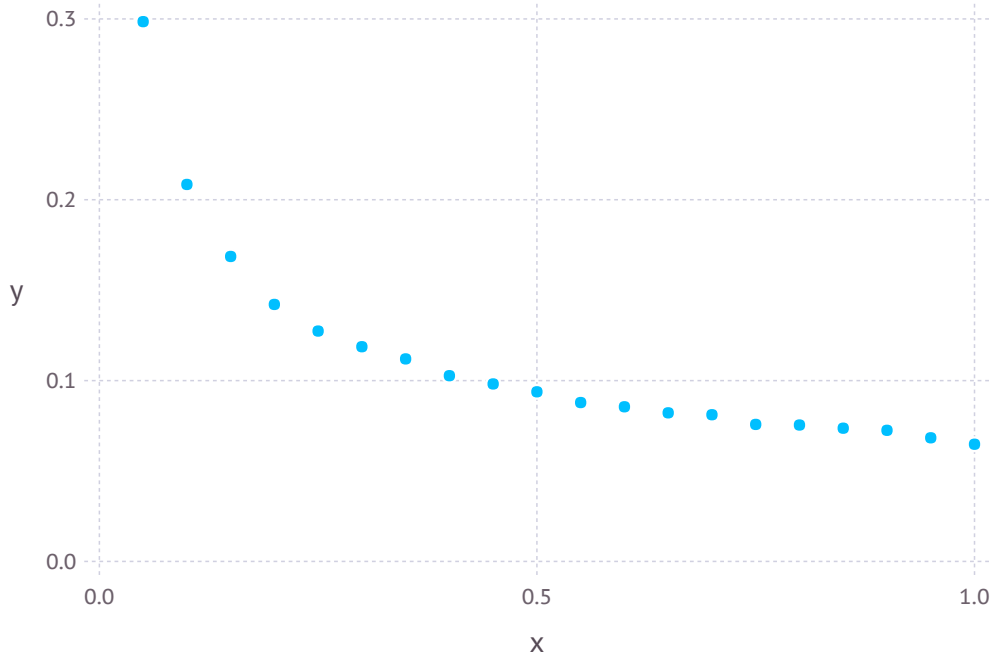
In [23]:
```
for i=1:(nRows-20)
LDMat[i,:] = corMat[i,(i+1):(i+20)].^2
end
```

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

Out[24]:
```
1x20 Array{Float64,2}:
 0.0648101  0.0683712  0.0725045  0.0736858  …  0.168632  0.208491  0.298477
```

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]:
```
32.69964480792459
```

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

Out[31]:
```
0.710401145453399
```

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

Out[32]:
```
0.710401145453399
```

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

Out[33]:
```
0.710401145453399
```

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]: 35.56154732036469

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

Out[36]: 35.571419359717225

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

Out[37]: 0.6762096376358417

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

Out[38]: 0.6789998132508759

Pedigree: All animals

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

Out[39]:
```
48000x3 Array{Int64,2}:
 40722  35002  37868
 40723  34430  39990
 40724  36534  38747
 40725  33849  37069
 40726  34854  40363
 40727  33594  39995
 40728  35051  38389
 40729  33520  39383
 40730  32986  37272
 40731  35336  36738
 40732  34855  39347
 40733  35686  40691
 40734  34032  40395
     ⋮
 88710  75903  78270
 88711  76631  80589
 88712  75229  77856
 88713  73293  80100
 88714  76288  80686
 88715  75772  79242
 88716  74976  80676
 88717  75216  79648
 88718  75628  80546
 88719  76299  80371
 88720  76699  80677
 88721  75628  80420
```

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]: 2000

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

Out[45]: 48000x2001 Array{Int64,2}:

40722	0	2	1	2	1	0	0	0	2	0	…	2	1	2	2	0	1	1	1	1	1	0	0
40723	0	1	1	1	1	0	1	1	1	1		1	1	1	1	0	1	2	2	1	0	1	0
40724	0	2	0	2	2	0	0	0	2	0		1	1	1	1	0	0	0	0	0	2	0	0
40725	0	1	1	2	2	1	1	1	1	1		2	1	2	2	0	1	1	1	1	1	0	0
40726	0	2	0	2	2	0	1	1	2	0		2	2	2	2	1	2	2	2	1	0	1	1
40727	0	1	1	2	2	0	1	1	1	1	…	0	2	0	0	1	1	1	1	0	1	1	1
40728	0	1	1	1	1	0	1	1	1	1		2	2	2	2	0	2	2	2	2	0	0	0
40729	0	1	1	2	2	1	1	1	1	1		1	1	1	1	0	0	1	1	0	1	1	0
40730	0	2	0	2	2	0	0	0	2	0		0	2	0	1	0	1	1	1	1	1	1	1
40731	0	2	0	2	2	0	0	0	2	0		0	2	0	0	0	0	2	2	0	0	2	0
40732	0	2	0	2	2	0	0	0	2	0	…	2	0	2	2	0	0	0	0	0	2	0	0
40733	0	2	0	2	2	0	0	0	2	0		1	1	1	1	0	1	1	1	1	1	1	1
40734	0	2	1	2	1	0	0	0	2	0		1	2	1	1	0	2	2	2	2	0	1	1
⋮			⋮				⋮	⋱		⋮					⋮						⋮	
88710	0	2	1	2	1	0	0	0	2	0		1	1	1	1	0	0	1	1	0	1	0	0
88711	0	2	1	2	1	0	0	0	2	0		0	2	0	0	0	0	2	2	0	0	2	2
88712	0	0	2	2	2	1	2	2	0	2	…	1	1	1	0	0	0	1	1	0	1	1	0
88713	1	1	1	2	2	0	2	2	0	2		1	2	1	1	0	0	2	2	1	0	1	0
88714	2	2	0	2	2	0	2	2	0	2		1	1	1	1	0	0	1	1	0	1	1	0
88715	1	2	0	2	2	0	1	1	1	1		1	2	1	1	0	0	2	2	1	0	1	1
88716	0	1	1	2	2	1	1	1	1	1		2	0	2	2	0	0	0	0	0	2	0	0
88717	0	1	2	1	0	0	1	1	1	1	…	1	2	1	1	0	1	2	2	1	0	1	0
88718	1	2	1	2	1	0	0	0	2	0		1	2	1	1	0	1	2	2	1	0	1	0
88719	0	2	0	2	2	1	1	1	1	1		2	0	1	1	1	1	1	1	0	1	1	1
88720	0	1	2	2	1	0	1	1	1	1		2	1	2	2	0	0	1	1	1	1	0	0
88721	0	2	0	2	2	0	0	0	2	0		1	2	1	1	0	0	2	2	1	0	1	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]:
```
48000x2000 Array{Int64,2}:
 0  2  1  2  1  0  0  0  2  0  0  2  1  …  2  1  2  2  0  1  1  1  1  1  0  0
 0  1  1  1  1  0  1  1  1  1  1  0  1     1  1  1  1  0  1  2  2  1  0  1  0
 0  2  0  2  2  0  0  0  2  0  0  0  0     1  1  1  1  0  0  0  0  0  2  0  0
 0  1  1  2  2  1  1  1  1  1  1  1  2     2  1  2  2  0  1  1  1  1  1  0  0
 0  2  0  2  2  0  1  1  2  0  0  1  1     2  2  2  2  1  2  2  2  1  0  1  1
 0  1  1  2  2  0  1  1  1  1  1  0  0  …  0  2  0  0  1  1  1  1  0  1  1  1
 0  1  1  1  1  0  1  1  1  1  1  0  1     2  2  2  2  0  2  2  2  2  0  0  0
 0  1  1  2  2  1  1  1  1  1  1  1  1     1  1  1  1  0  0  1  1  0  1  1  0
 0  2  0  2  2  0  0  0  2  0  0  0  1     0  2  0  1  0  1  1  1  1  1  1  1
 0  2  0  2  2  0  0  0  2  0  0  1  1     0  2  0  0  0  0  2  2  0  0  2  0
 0  2  0  2  2  0  0  0  2  0  0  1  1  …  2  0  2  2  0  0  0  0  0  2  0  0
 0  2  0  2  2  0  0  0  2  0  0  1  1     1  1  1  1  0  1  1  1  1  1  1  1
 0  2  1  2  1  0  0  0  2  0  0  2  1     1  2  1  1  0  2  2  2  2  0  1  1
 ⋮              ⋮              ⋮        ⋱        ⋮              ⋮
 0  2  1  2  1  0  0  0  2  0  0  1  0     1  1  1  1  0  0  1  1  0  1  0  0
 0  2  1  2  1  0  0  0  2  0  0  1  0     0  2  0  0  0  0  2  2  0  0  2  2
 0  0  2  2  2  1  2  2  0  2  2  0  1  …  1  1  1  0  0  0  1  1  0  1  1  0
 1  1  1  2  2  0  2  2  0  2  2  0  1     1  2  1  1  0  0  2  2  1  0  1  0
 2  2  0  2  2  0  2  2  0  2  2  0  0     1  1  1  1  0  0  1  1  0  1  1  0
 1  2  0  2  2  0  1  1  1  1  1  0  0     1  2  1  1  0  0  2  2  1  0  1  1
 0  1  1  2  2  1  1  1  1  1  0  2  1     2  0  2  2  0  0  0  0  0  2  0  0
 0  1  2  1  0  0  1  1  1  1  1  1  1  …  1  2  1  1  0  1  2  2  1  0  1  0
 1  2  1  2  1  0  0  0  2  0  0  2  0     1  2  1  1  0  1  2  2  1  0  1  0
 0  2  0  2  2  1  1  1  1  1  1  0  1     2  0  1  1  1  1  1  1  0  1  1  1
 0  1  2  2  1  0  1  1  1  1  1  1  0     2  1  2  2  0  0  1  1  1  1  0  0
 0  2  0  2  2  0  0  0  2  0  0  0  0     1  2  1  1  0  0  2  2  1  0  1  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}:
 40901
 42982
 44428
 43496
 41814
 42172
 41735
 41479
 41373
 43926
 43404
 44349
 43141
    ⋮
 75903
 76631
 75229
 73293
 76288
 75772
 74976
 75216
 75628
 76299
 76699
 75628
```

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

Out[53]:
```
1000-element Array{Int64,1}:
 40901
 42982
 44428
 43496
 41814
 42172
 41735
 41479
 41373
 43926
 43404
 44349
 43141
     ⋮
 76299
 75296
 75602
 73574
 76451
 73550
 76167
 76037
 74937
 76136
 75938
 76513
```

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}:
 40901
 42982
 44428
 43496
 41814
 42172
 41735
 41479
 41373
 43926
 43404
 44349
 43141
     ⋮
 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]: (9000,2001)

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

Out[62]: 9000

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

Out[63]: 2001

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  33.343  32.98
 40723  33.892  33.171
 40724  32.266  33.176
 40725  31.585  32.292
 40726  35.976  33.943
 40727  33.835  34.327
 40728  35.419  33.5
 40729  34.064  33.111
 40730  31.26   32.85
 40731  30.267  30.394
 40732  32.44   32.278
 40733  32.001  31.853
 40734  32.689  32.606
     ⋮
 88710  33.983  36.472
 88711  37.546  36.851
 88712  36.321  36.653
 88713  34.902  35.392
 88714  36.269  36.212
 88715  36.725  35.958
 88716  36.006  35.712
 88717  35.759  35.646
 88718  34.429  36.727
 88719  34.94   34.429
 88720  38.333  36.099
 88721  36.438  35.585
```

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]:
```
500-element Array{Int64,1}:
    3
    9
   20
   28
   29
   30
   32
   35
   40
   41
   50
   51
   65
    ⋮
 1957
 1958
 1959
 1961
 1962
 1963
 1973
 1977
 1978
 1982
 1988
 1993
```

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

Out[83]:
```
1500-element Array{Int64,1}:
    1
    2
    4
    5
    6
    7
    8
   10
   11
   12
   13
   14
   15
    ⋮
 1987
 1989
 1990
 1991
 1992
 1994
 1995
 1996
 1997
 1998
 1999
 2000
```

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]: 9000x2000 Array{Int64,2}:

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

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

Out[86]:
```
9000x500 Array{Int64,2}:
 0  2  2  1  1  1  2  1  1  0  0  0  1  …  2  2  0  1  2  0  2  1  2  1  0  1
 1  1  0  2  2  2  2  0  0  0  1  0  1     1  1  2  0  1  0  2  1  2  0  1  0
 1  0  1  2  2  1  2  1  1  1  0  2  1     1  1  2  1  1  1  2  2  2  1  1  0
 1  0  1  2  2  1  2  0  0  1  0  1  1     0  1  1  2  0  2  1  1  2  0  0  0
 0  2  0  2  2  0  2  2  0  0  2  2  2     0  1  1  1  0  2  0  1  2  1  2  1
 1  2  2  0  0  0  1  1  0  0  1  1  1  …  0  1  1  1  0  2  1  0  1  0  2  1
 0  1  2  1  1  1  1  0  0  0  1  1  0     2  2  0  2  2  0  2  0  2  0  0  0
 2  1  1  1  1  1  1  0  1  1  0  1  2     1  1  1  2  2  0  2  1  2  1  1  0
 1  1  1  1  1  1  2  2  0  0  1  1  2     0  1  1  1  1  1  1  0  2  0  0  0
 1  0  1  1  2  1  2  1  0  0  1  1  0     0  0  2  0  2  1  1  1  2  1  1  0
 0  1  2  0  1  0  1  1  0  0  0  1  0  …  0  0  2  0  2  0  2  2  2  2  2  0
 0  2  1  1  1  0  2  2  0  0  2  1  2     0  2  1  1  0  2  1  1  2  0  0  0
 1  2  1  1  1  0  2  2  0  0  1  2  1     1  2  0  2  1  1  1  0  2  0  0  0
 ⋮              ⋮              ⋮        ⋱     ⋮              ⋮
 1  2  1  1  1  0  2  2  0  0  2  2  2     0  1  1  2  1  1  1  1  2  1  1  0
 1  2  1  1  1  0  2  2  0  0  2  2  2     2  2  0  2  1  1  2  2  2  2  2  0
 2  0  0  2  2  1  2  0  0  0  1  1  1  …  1  1  2  1  1  1  2  2  2  1  1  0
 1  0  0  2  2  1  2  0  0  0  1  1  1     1  1  1  1  2  0  2  1  2  1  1  0
 0  0  1  1  1  1  2  2  1  1  1  2  1     0  1  1  0  1  1  1  1  2  1  1  0
 0  1  1  1  1  1  2  1  0  0  1  2  2     1  1  1  2  1  1  2  0  2  0  1  0
 1  1  0  2  2  0  2  2  0  0  2  2  1     0  2  0  2  0  2  0  0  2  0  0  0
 2  1  1  1  1  1  2  1  0  0  2  2  2  …  1  1  1  1  1  0  2  1  2  0  1  0
 1  2  1  0  0  0  2  2  0  0  1  2  2     1  1  1  2  1  1  2  1  2  1  1  0
 0  1  1  1  2  0  2  2  0  0  1  2  1     0  1  1  1  0  1  1  0  2  0  0  1
 2  1  1  1  1  1  2  1  0  0  1  1  2     2  2  0  2  1  1  2  1  2  1  0  0
 0  2  1  1  1  1  2  1  0  0  1  1  1     1  1  1  2  1  1  2  1  2  1  1  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]: 0.6844524233928011

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

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

Out[100]:
```
0.8256325837635035
```

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

Out[101]:
```
48000x500 Array{Int64,2}:
 1  2  1  1  1  0  2  1  0  0  1  1  2  …  1  2  0  2  1  1  1  0  2  0  0  0
 1  1  0  2  2  1  2  1  0  0  2  1  1     1  1  2  2  0  2  2  2  2  1  1  0
 0  2  1  1  1  1  2  1  0  0  1  1  2     1  2  0  2  0  1  0  0  1  0  1  0
 1  1  1  1  1  0  2  1  0  0  1  0  2     1  2  0  2  1  1  1  0  2  0  0  0
 0  2  2  1  1  1  2  1  1  0  0  0  1     2  2  0  1  2  0  2  1  2  1  0  1
 1  1  1  1  1  1  2  1  1  1  1  2  2  …  1  2  0  1  1  1  1  0  1  0  1  1
 1  1  0  2  2  2  2  0  0  0  1  0  1     1  1  2  0  1  0  2  1  2  0  1  0
 1  1  1  1  1  0  1  1  0  0  1  1  1     0  1  1  2  0  2  1  1  2  1  1  0
 0  2  1  1  1  0  2  2  0  0  2  1  2     1  1  1  1  1  1  2  1  2  1  2  0
 0  2  1  1  1  1  1  0  1  0  0  0  1     0  2  0  2  0  2  2  2  2  2  2  0
 0  2  1  2  2  0  2  2  0  0  2  2  2  …  0  2  0  2  0  2  0  0  2  0  0  0
 0  2  1  1  1  0  1  1  0  0  1  1  1     0  1  1  2  1  1  1  1  2  1  1  0
 1  2  2  0  0  0  1  1  0  0  1  1  1     1  2  1  2  2  0  2  1  2  1  1  0
 ⋮              ⋮              ⋮          ⋱        ⋮              ⋮
 1  2  1  1  1  0  2  2  0  0  2  2  2     0  1  1  2  1  1  1  1  2  1  1  0
 1  2  1  1  1  0  2  2  0  0  2  2  2     2  2  0  2  1  1  2  2  2  2  2  0
 2  0  0  2  2  1  2  0  0  0  1  1  1  …  1  1  2  1  1  1  2  2  2  1  1  0
 1  0  0  2  2  1  2  0  0  0  1  1  1     1  1  1  1  2  0  2  1  2  1  1  0
 0  0  1  1  1  1  2  2  1  1  1  2  1     0  1  1  0  1  1  1  1  2  1  1  0
 0  1  1  1  1  1  2  1  0  0  1  2  2     1  1  1  2  1  1  2  0  2  0  1  0
 1  1  0  2  2  0  2  2  0  0  2  2  1     0  2  0  2  0  2  0  0  2  0  0  0
 2  1  1  1  1  1  2  1  0  0  2  2  2  …  1  1  1  1  1  0  2  1  2  0  1  0
 1  2  1  0  0  0  2  2  0  0  1  2  2     1  1  1  2  1  1  2  1  2  1  1  0
 0  1  1  1  2  0  2  2  0  0  1  2  1     0  1  1  1  0  1  1  0  2  0  0  1
 2  1  1  1  1  1  2  1  0  0  1  1  2     2  2  0  2  1  1  2  1  2  1  0  0
 0  2  1  1  1  1  2  1  0  0  1  1  1     1  1  1  2  1  1  2  1  2  1  1  0
```

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

Out[102]:
```
500-element Array{Float64,1}:
 0.0644439
 0.0638382
 0.0637036
 0.0631948
 0.0629696
 0.0636714
 0.063029
 0.0635087
 0.06334
 0.0625102
 0.0621025
 0.0628374
 0.0626338
 ⋮
 0.0620948
 0.0634836
 0.0630525
 0.062127
 0.0642011
 0.0654771
 0.0617998
 0.0628596
 0.063144
 0.0633209
 0.0627911
 0.0637483
```

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

Out[103]:
```
48000-element Array{Float64,1}:
 32.945
 33.1275
 33.128
 32.2462
 33.9102
 34.2769
 33.4585
 33.0716
 32.816
 30.3668
 32.2619
 31.815
 32.5634
  ⋮
 36.4254
 36.8021
 36.606
 35.3419
 36.1676
 35.8982
 35.6687
 35.587
 36.6617
 34.392
 36.0441
 35.5349
```

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

Out[104]:
```
32.65977150486079
```

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

Out[105]:
```
33.19412346026536
```

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

Out[106]:
```
33.792576533325324
```

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

Out[107]:
```
34.367362175510486
```

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

Out[108]:
```
34.94972281537506
```

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

Out[109]:
```
35.518947163499355
```

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

Out[110]:
```
9000-element Array{Float64,1}:
 33.9102
 33.4585
 34.585
 34.0926
 33.327
 33.4569
 33.4439
 33.6359
 32.9584
 33.4452
 34.2721
 33.5239
 33.0824
  ⋮
 36.4254
 36.8021
 36.606
 35.3419
 36.1676
 35.8982
 35.6687
 35.587
 36.6617
 34.392
 36.0441
 35.5349
```

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

Out[111]: 35.45365087290199

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

Out[112]: 2.793879368041196

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

Out[113]: 33.73129073673009

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

Out[114]: 1.0715192318692957

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

Out[115]: 34.39682891660554

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

Out[116]: 1.7370574117447504

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

Out[117]: 34.94479566204535

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

Out[118]: 2.2850241571845586

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

Out[119]: 35.52016035953361

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

Out[120]: 2.86038885467282

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

Out[121]: 36.06332706570097

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

Out[122]: 3.403555560840175

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

Out[123]: 35.518947163499355

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

Out[124]: 2.859175658638563