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)
```

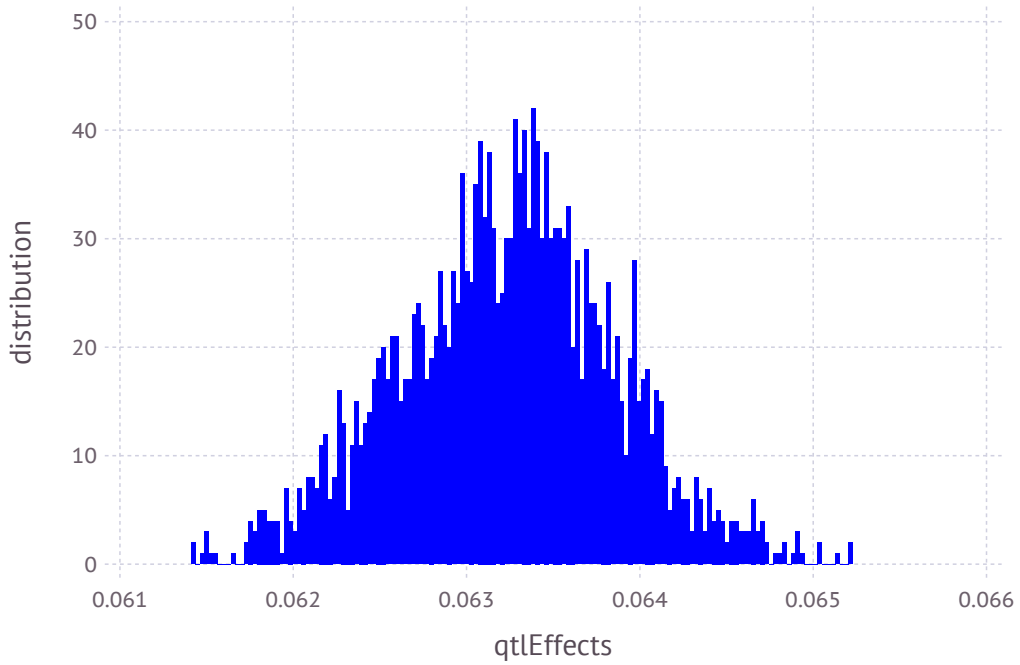
In [7]:
```
qtlEffects
```

Out[7]:
```
2000-element Array{Float64,1}:
 0.0632941
 0.0638879
 0.0630065
 0.0620426
 0.0626892
 0.0632167
 0.0622649
 0.0633408
 0.0638992
 0.0635867
 0.063167
 0.0637288
 0.0640845
 ⋮
 0.0634036
 0.0629021
 0.0633171
 0.062482
 0.0638333
 0.0635623
 0.0646385
 0.0627117
 0.0638845
 0.0639525
 0.0628769
 0.0637174
```

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]: 0.06323356741044617

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

Out[11]: 3.9143527341239495e-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
```

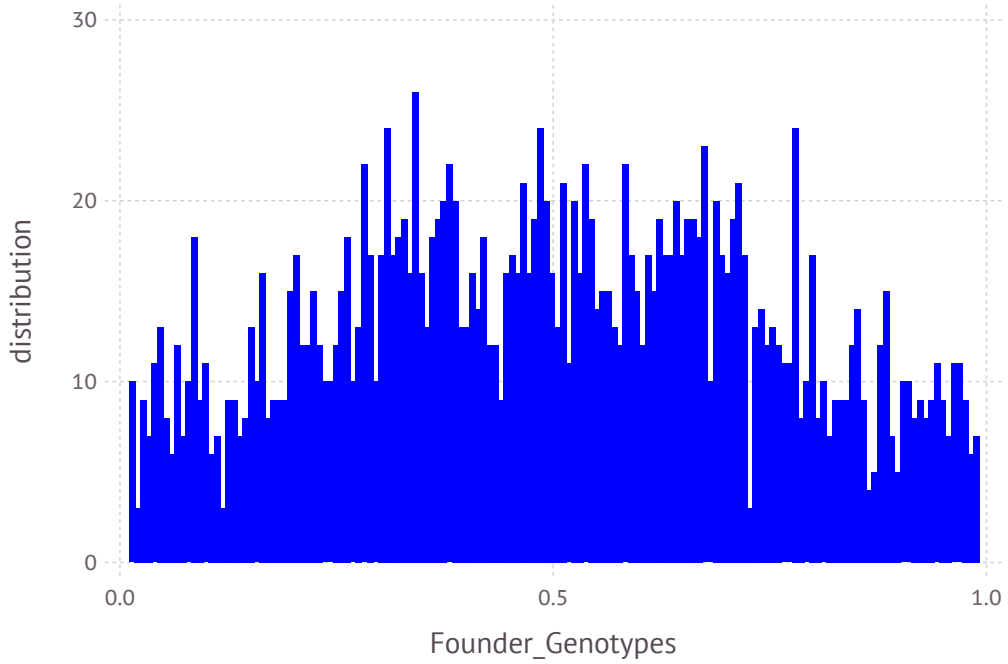
In [19]:
```
gSPSire = XSim.getOurGenotypes(popSP[1])
gSPDam = XSim.getOurGenotypes(popSP[2])
gSP = [gSPSire;gSPSire];
```

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

Out[20]:
```
1x2000 Array{Float64,2}:
 0.067125  0.839875  0.28575  0.946125  0.82  …  0.362875  0.45075  0.283125
```

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);
```

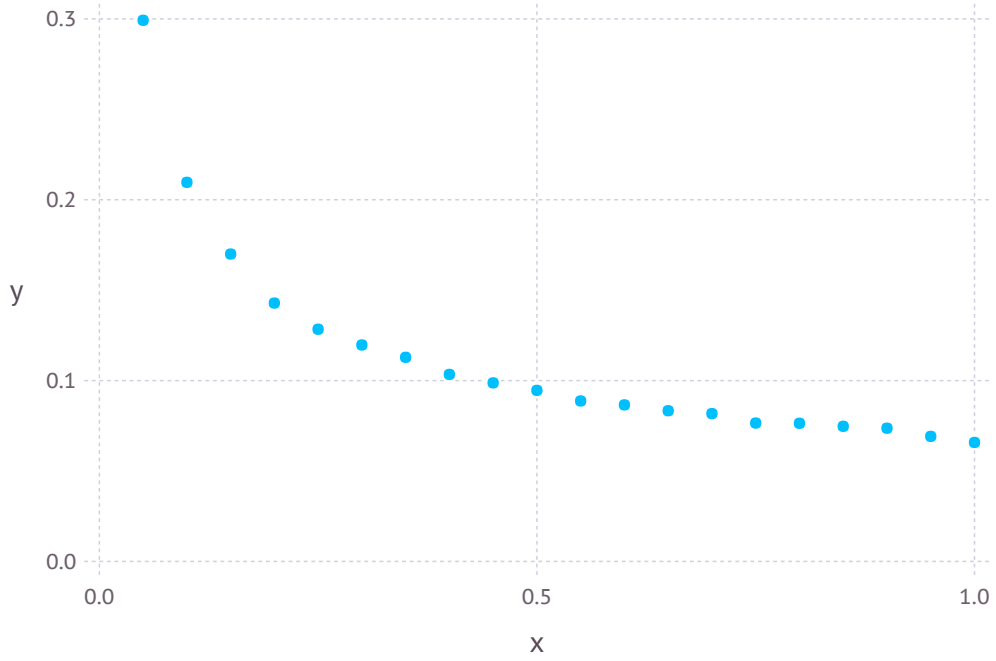
In [23]:
```
for i=1:(nRows-20)
LDMat[i,:] = corMat[i,(i+1):(i+20)].^2
end
```

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

Out[24]:
```
1x20 Array{Float64,2}:
 0.0658027  0.0691725  0.073695  0.0747299  …  0.169977  0.209651  0.299246
```

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]:
```
32.634861485164194
```

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

Out[31]:
```
0.8586708495985766
```

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

Out[32]:
```
0.8586708495985766
```

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

Out[33]:
```
0.8586708495985766
```

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]: 35.447511468804905

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

Out[36]: 35.43487780806261

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

Out[37]: 0.5707489120327502

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

Out[38]: 0.5849422038238666

Pedigree: All animals

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

Out[39]:
```
48000x3 Array{Int64,2}:
 40722  34668  40280
 40723  35319  39308
 40724  32754  38549
 40725  35720  37581
 40726  33213  38198
 40727  34775  40262
 40728  33323  39745
 40729  33854  37217
 40730  34313  37659
 40731  34135  40522
 40732  36428  38932
 40733  34241  37445
 40734  36164  36892
     ⋮
 88710  73029  78758
 88711  74971  78073
 88712  74294  79536
 88713  74155  79993
 88714  75558  80694
 88715  73764  80717
 88716  73354  78500
 88717  76255  80636
 88718  76204  78282
 88719  76255  80498
 88720  75445  78068
 88721  74440  79820
```

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	2	0	0	0	2	0	…	2	2	2	2	0	1	2	2	1	0	1	0
40723	0	2	2	2	0	0	0	0	2	0		2	1	2	1	0	1	2	2	1	0	1	1
40724	0	2	1	2	1	0	0	0	2	0		0	2	0	0	0	0	1	1	0	1	1	0
40725	0	2	1	2	1	0	0	0	2	0		2	2	2	2	1	2	2	2	1	0	1	1
40726	0	2	0	2	2	0	0	0	2	0		1	1	0	0	0	2	2	2	2	0	2	2
40727	0	2	1	2	1	0	1	1	1	1	…	0	2	0	0	0	0	2	2	0	0	2	0
40728	1	2	0	2	2	0	0	0	2	0		2	1	2	1	1	1	2	2	0	0	2	1
40729	0	1	2	2	1	1	1	1	1	1		2	1	1	1	1	2	2	2	1	0	1	1
40730	0	2	0	2	2	1	1	1	1	1		2	1	2	1	0	1	2	2	1	0	1	0
40731	1	2	0	2	2	0	0	0	2	0		2	1	1	1	1	0	1	1	0	1	1	1
40732	0	2	1	2	1	0	0	0	2	0	…	2	1	2	2	0	1	1	1	1	1	0	0
40733	0	2	1	2	1	0	0	0	2	0		2	2	2	2	0	1	2	2	0	0	2	2
40734	0	2	0	2	2	0	0	0	2	0		1	1	1	1	0	1	1	1	1	1	1	1
⋮			⋮				⋮	⋱		⋮					⋮					⋮		
88710	0	2	0	2	2	0	0	0	2	0		2	2	2	2	0	2	2	2	2	0	0	0
88711	0	2	0	2	2	0	0	0	2	0		1	1	1	1	1	1	1	1	0	1	1	1
88712	0	1	1	2	2	0	1	1	1	1	…	2	0	2	2	0	0	0	0	0	2	0	0
88713	0	2	0	2	2	0	0	0	2	0		0	2	0	0	1	1	1	1	1	2	1	0
88714	0	2	0	2	2	0	1	1	2	0		0	2	0	0	0	1	1	1	1	1	1	1
88715	0	1	1	2	2	1	1	1	1	1		1	1	1	1	0	0	1	1	0	1	1	0
88716	0	1	1	2	2	1	2	2	0	2		1	2	1	1	0	1	1	1	1	1	0	0
88717	0	1	1	2	2	1	1	1	1	1	…	2	1	1	1	1	1	1	1	0	1	1	1
88718	0	2	0	2	2	0	0	0	2	0		1	1	0	0	0	1	2	2	1	0	2	1
88719	0	2	0	2	2	0	0	0	2	0		1	1	1	1	1	1	1	1	0	1	1	1
88720	0	2	0	2	2	0	0	0	2	0		2	0	2	1	0	0	1	1	0	1	1	0
88721	0	2	1	2	1	0	0	0	2	0		1	2	1	1	0	1	1	1	1	1	0	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  2  0  0  0  2  0  0  1  1  …  2  2  2  2  0  1  2  2  1  0  1  0
 0  2  2  2  0  0  0  0  2  0  0  2  0     2  1  2  1  0  1  2  2  1  0  1  1
 0  2  1  2  1  0  0  0  2  0  0  1  0     0  2  0  0  0  0  1  1  0  1  1  0
 0  2  1  2  1  0  0  0  2  0  0  2  1     2  2  2  2  1  2  2  2  1  0  1  1
 0  2  0  2  2  0  0  0  2  0  0  0  0     1  1  0  0  0  2  2  2  2  0  2  2
 0  2  1  2  1  0  1  1  1  1  1  1  0  …  0  2  0  0  0  0  2  2  0  0  2  0
 1  2  0  2  2  0  0  0  2  0  0  2  1     2  1  2  1  1  1  2  2  0  0  2  1
 0  1  2  2  1  1  1  1  1  1  1  1  1     2  1  1  1  1  2  2  2  1  0  1  1
 0  2  0  2  2  1  1  1  1  1  1  0  1     2  1  2  1  0  1  2  2  1  0  1  0
 1  2  0  2  2  0  0  0  2  0  0  1  0     2  1  1  1  1  0  1  1  0  1  1  1
 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     2  2  2  2  0  1  2  2  0  0  2  2
 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  0  2  2  0  0  0  2  0  0  1  1     2  2  2  2  0  2  2  2  2  0  0  0
 0  2  0  2  2  0  0  0  2  0  0  1  1     1  1  1  1  1  1  1  1  0  1  1  1
 0  1  1  2  2  0  1  1  1  1  1  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     0  2  0  0  1  1  1  1  1  2  1  0
 0  2  0  2  2  0  1  1  2  0  0  2  2     0  2  0  0  0  1  1  1  1  1  1  1
 0  1  1  2  2  1  1  1  1  1  1  1  2     1  1  1  1  0  0  1  1  0  1  1  0
 0  1  1  2  2  1  2  2  0  2  2  0  1     1  2  1  1  0  1  1  1  1  1  0  0
 0  1  1  2  2  1  1  1  1  1  1  1  2  …  2  1  1  1  1  1  1  1  0  1  1  1
 0  2  0  2  2  0  0  0  2  0  0  2  2     1  1  0  0  0  1  2  2  1  0  2  1
 0  2  0  2  2  0  0  0  2  0  0  1  1     1  1  1  1  1  1  1  1  0  1  1  1
 0  2  0  2  2  0  0  0  2  0  0  1  1     2  0  2  1  0  0  1  1  0  1  1  0
 0  2  1  2  1  0  0  0  2  0  0  2  2     1  2  1  1  0  1  1  1  1  1  0  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}:
 44401
 41861
 44436
 41625
 40823
 41530
 42256
 43867
 41597
 41861
 41950
 43644
 44531
     ⋮
 73029
 74971
 74294
 74155
 75558
 73764
 73354
 76255
 76204
 76255
 75445
 74440
```

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

Out[53]:
```
1000-element Array{Int64,1}:
 44401
 41861
 44436
 41625
 40823
 41530
 42256
 43867
 41597
 41950
 43644
 44531
 44038
     ⋮
 76563
 74720
 75049
 76023
 76464
 74244
 74856
 74897
 76631
 74976
 75161
 76492
```

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}:
 44401
 41861
 44436
 41625
 40823
 41530
 42256
 43867
 41597
 41950
 43644
 44531
 44038
     ⋮
 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  31.411  31.395
 40723  31.469  33.294
 40724  33.391  33.223
 40725  30.876  32.154
 40726  30.848  31.829
 40727  32.012  32.412
 40728  32.622  32.468
 40729  31.57   31.381
 40730  30.399  31.832
 40731  32.932  33.425
 40732  33.153  32.345
 40733  32.584  32.278
 40734  32.178  31.966
     ⋮
 88710  37.341  36.151
 88711  36.06   35.894
 88712  36.192  36.837
 88713  35.352  35.902
 88714  35.584  35.197
 88715  34.355  35.76
 88716  34.354  36.078
 88717  37.792  37.276
 88718  36.378  35.509
 88719  36.662  35.633
 88720  35.027  34.679
 88721  36.047  36.399
```

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}:
    4
    8
    9
   33
   34
   37
   40
   50
   52
   57
   74
   76
   78
    ⋮
 1946
 1950
 1952
 1964
 1965
 1968
 1970
 1976
 1988
 1996
 1998
 2000
```

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

Out[83]:
```
1500-element Array{Int64,1}:
    1
    2
    3
    5
    6
    7
   10
   11
   12
   13
   14
   15
   16
    ⋮
 1985
 1986
 1987
 1989
 1990
 1991
 1992
 1993
 1994
 1995
 1997
 1999
```

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

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

Out[86]:
```
9000x500 Array{Int64,2}:
 2  1  1  0  2  2  0  1  2  2  2  2  0  …  2  0  0  0  0  2  0  1  2  2  0  1
 2  0  2  1  2  1  0  1  1  1  2  0  2     2  1  0  2  0  0  2  1  2  2  0  2
 2  0  2  1  1  1  0  2  1  2  2  2  2     2  0  0  2  1  0  2  2  1  1  1  0
 2  0  2  1  2  1  0  1  1  1  2  0  2     2  0  1  1  2  1  1  2  0  1  1  1
 2  2  0  2  1  0  0  0  2  1  2  1  2     1  1  1  2  0  1  1  1  0  2  0  1
 2  1  1  1  1  1  0  1  1  2  2  1  2  …  2  1  1  2  1  1  1  1  0  0  2  0
 2  1  1  2  0  0  0  1  2  2  1  2  2     2  1  0  1  1  1  1  1  1  0  2  0
 2  0  2  0  2  2  0  2  2  2  2  2  0     2  0  0  2  2  0  2  2  0  0  2  0
 1  0  2  1  2  1  0  1  1  1  2  0  2     2  1  0  1  1  1  1  2  0  1  1  0
 2  2  0  2  0  0  0  0  2  2  2  2  2     2  1  0  2  1  0  2  2  0  1  1  0
 2  1  1  1  2  1  0  1  1  1  2  1  1  …  2  1  0  2  1  0  2  2  1  0  2  0
 2  1  1  1  1  1  0  1  2  2  2  2  2     1  1  0  1  1  1  1  0  1  2  0  1
 2  2  0  2  0  0  2  0  2  0  2  2  2     1  1  0  2  1  2  0  1  0  2  0  0
 ⋮              ⋮              ⋮           ⋱        ⋮              ⋮
 2  0  2  1  2  1  0  1  1  1  2  0  2     0  2  0  2  0  2  0  0  0  2  0  0
 2  0  2  2  1  1  0  0  2  1  2  1  2     2  2  0  2  1  0  2  2  1  1  1  1
 2  1  1  1  2  1  0  1  2  1  2  2  1  …  2  0  0  2  2  0  2  2  0  0  2  0
 2  0  2  1  2  1  0  1  2  1  2  1  1     2  1  0  1  1  1  2  2  2  1  2  0
 2  1  2  2  1  0  1  0  2  0  2  1  2     2  0  0  1  1  0  2  2  2  1  1  1
 2  1  1  1  2  2  0  2  1  2  2  2  1     2  0  0  1  1  1  1  2  1  1  1  0
 2  2  0  1  2  2  0  2  0  2  2  1  2     2  1  1  2  0  1  1  1  1  1  1  0
 2  1  1  2  2  1  0  1  1  1  2  2  2  …  1  0  0  2  1  2  1  2  1  1  1  1
 2  0  2  2  2  0  0  0  2  1  2  1  2     2  0  1  1  0  1  1  2  1  2  0  1
 2  0  2  1  2  1  0  1  1  1  2  1  2     2  1  1  2  0  0  2  2  1  1  1  1
 2  0  2  1  2  1  0  1  2  1  2  1  2     1  0  0  2  1  1  1  2  0  1  1  0
 2  0  2  2  2  0  0  0  2  0  2  0  2     1  1  0  2  0  1  1  1  1  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]: 0.6450420591577035

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

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

Out[100]:
```
0.8044504375843706
```

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

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

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

Out[102]:
```
500-element Array{Float64,1}:
 0.0620426
 0.0633408
 0.0638992
 0.0631013
 0.0636222
 0.0631969
 0.0631378
 0.063467
 0.0635
 0.0625703
 0.0629946
 0.0640528
 0.0639932
 ⋮
 0.063495
 0.0636571
 0.0624298
 0.0637974
 0.0631864
 0.0634991
 0.0632231
 0.063797
 0.0621881
 0.0627117
 0.0639525
 0.0637174
```

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

Out[103]:
```
48000-element Array{Float64,1}:
 31.3747
 33.2848
 33.212
 32.1298
 31.8171
 32.3854
 32.4587
 31.3705
 31.8201
 33.403
 32.3258
 32.2583
 31.9531
  ⋮
 36.1271
 35.8669
 36.8304
 35.876
 35.16
 35.7337
 36.0469
 37.2541
 35.4892
 35.6089
 34.6745
 36.3721
```

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

Out[104]:
```
32.61801354587032
```

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

Out[105]:
```
33.27143596486958
```

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

Out[106]:
```
33.88428140388499
```

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

Out[107]:
```
34.42088059652976
```

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

Out[108]:
```
34.966702380152526
```

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

Out[109]:
```
35.42155520392388
```

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

Out[110]:
```
9000-element Array{Float64,1}:
 32.7665
 34.0116
 34.4091
 33.0282
 33.5225
 33.9157
 33.5955
 33.6778
 34.1801
 33.1438
 33.5274
 33.464
 33.7884
  ⋮
 36.1271
 35.8669
 36.8304
 35.876
 35.16
 35.7337
 36.0469
 37.2541
 35.4892
 35.6089
 34.6745
 36.3721
```

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

Out[111]: 35.36869825971409

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

Out[112]: 2.750684713843768

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

Out[113]: 33.910536687725056

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

Out[114]: 1.292523141854737

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

Out[115]: 34.465239395390846

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

Out[116]: 1.8472258495205267

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

Out[117]: 34.955332546889366

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

Out[118]: 2.337319001019047

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

Out[119]: 35.51619402972564

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

Out[120]: 2.8981804838553202

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

Out[121]: 35.88191087044751

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

Out[122]: 3.2638973245771936

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

Out[123]: 35.42155520392388

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

Out[124]: 2.803541658053561