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)
```

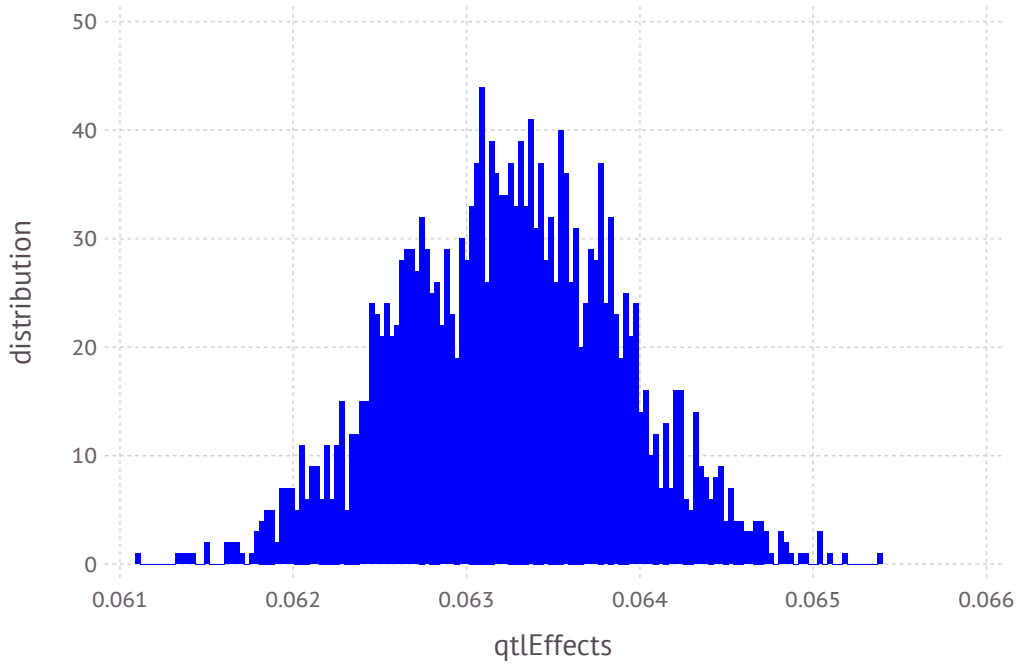
In [7]:
```
qtlEffects
```

Out[7]:
```
2000-element Array{Float64,1}:
 0.0633153
 0.0638437
 0.0638025
 0.0632424
 0.0636102
 0.0639411
 0.0638006
 0.0628264
 0.0616876
 0.0629352
 0.0621043
 0.0627512
 0.0624903
 ⋮
 0.0622024
 0.0638138
 0.062691
 0.0628327
 0.0627688
 0.0632083
 0.0629444
 0.0622376
 0.0627101
 0.0629937
 0.0633465
 0.0624975
```

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]: 0.06323994905242337

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

Out[11]: 4.1148079020099905e-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
```

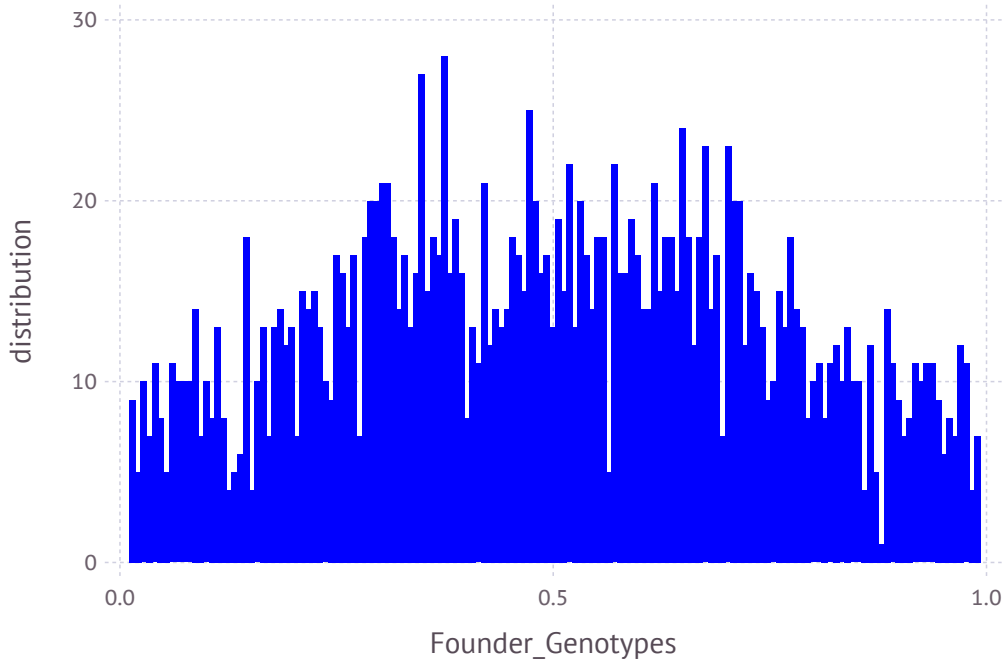
In [19]:
```
gSPSire = XSim.getOurGenotypes(popSP[1])
gSPDam = XSim.getOurGenotypes(popSP[2])
gSP = [gSPSire;gSPSire];
```

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

Out[20]:
```
1x2000 Array{Float64,2}:
 0.065125  0.828  0.29  0.941  0.82175  …  0.37175  0.435875  0.272625
```

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);
```

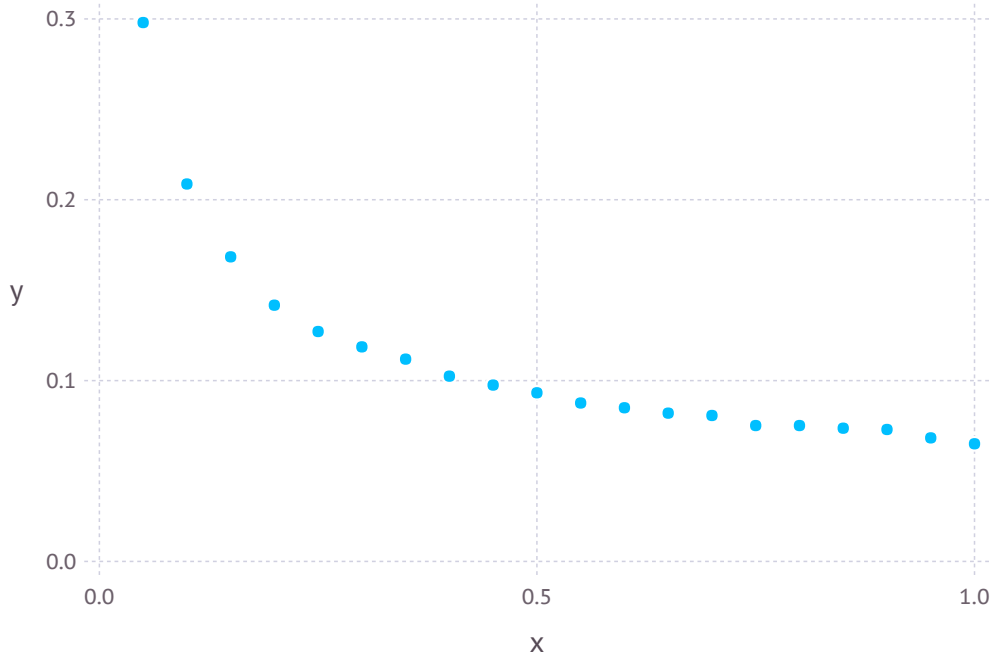
In [23]:
```
for i=1:(nRows-20)
LDMat[i,:] = corMat[i,(i+1):(i+20)].^2
end
```

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

Out[24]:
```
1x20 Array{Float64,2}:
 0.0650744  0.0683211  0.0729652  …  0.141692  0.168427  0.208727  0.298
```

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]:
```
31.488626174981874
```

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

Out[31]:
```
0.9101323492910246
```

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

Out[32]:
```
0.9101323492910246
```

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

Out[33]:
```
0.9101323492910246
```

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]: 34.617571309887126

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

Out[36]: 34.62796213326653

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

Out[37]: 0.6678938379009868

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

Out[38]: 0.6616568862975195

Pedigree: All animals

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

Out[39]:
```
48000x3 Array{Int64,2}:
 40722  36006  39864
 40723  35090  38736
 40724  33914  40047
 40725  33945  38565
 40726  36067  37719
 40727  33452  37767
 40728  34333  40599
 40729  33110  37379
 40730  35722  40678
 40731  36630  38686
 40732  33630  37477
 40733  33660  36764
 40734  35618  40105
     ⋮
 88710  73859  78702
 88711  74979  80339
 88712  76708  80256
 88713  75227  80426
 88714  75604  79429
 88715  75239  77978
 88716  75897  78248
 88717  75227  80528
 88718  76545  77627
 88719  76317  80700
 88720  75420  79792
 88721  73013  79596
```

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

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

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}:
 43737
 42637
 44510
 44484
 41553
 44142
 41553
 42526
 43976
 41119
 41328
 44704
 43527
     ⋮
 73859
 74979
 76708
 75227
 75604
 75239
 75897
 75227
 76545
 76317
 75420
 73013
```

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

Out[53]:
```
1000-element Array{Int64,1}:
 43737
 42637
 44510
 44484
 41553
 44142
 42526
 43976
 41119
 41328
 44704
 43527
 41198
     ⋮
 75369
 73128
 74439
 73844
 76708
 76538
 74896
 73354
 76469
 75862
 74378
 73800
```

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}:
 43737
 42637
 44510
 44484
 41553
 44142
 42526
 43976
 41119
 41328
 44704
 43527
 41198
     ⋮
 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  29.66   29.632
 40723  32.76   32.471
 40724  32.642  32.412
 40725  29.45   30.707
 40726  30.159  30.322
 40727  30.866  30.577
 40728  29.365  29.815
 40729  32.227  33.183
 40730  33.083  32.736
 40731  31.996  32.021
 40732  30.448  31.889
 40733  31.607  31.009
 40734  31.764  32.417
     ⋮
 88710  36.058  35.073
 88711  34.214  35.774
 88712  35.835  35.885
 88713  35.855  36.216
 88714  34.887  35.648
 88715  36.653  35.331
 88716  34.833  35.904
 88717  35.988  35.193
 88718  33.96   33.747
 88719  36.001  36.027
 88720  35.338  35.331
 88721  36.461  36.401
```

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}:
    2
    4
    5
    9
   11
   12
   14
   15
   17
   29
   37
   38
   41
    ⋮
 1966
 1969
 1971
 1973
 1977
 1978
 1982
 1984
 1986
 1991
 1992
 1999
```

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

Out[83]:
```
1500-element Array{Int64,1}:
    1
    3
    6
    7
    8
   10
   13
   16
   18
   19
   20
   21
   22
    ⋮
 1985
 1987
 1988
 1989
 1990
 1993
 1994
 1995
 1996
 1997
 1998
 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	1	1	2	2	1	0	0	2	0	0	1	1	…	1	2	1	1	1	1	2	2	0	0	2	1
0	1	2	2	1	1	1	1	1	1	1	1	2		2	0	2	1	1	1	1	1	1	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	1	2	2	1	0	1	1	1	1	1	1	1		2	1	1	1	0	2	2	2	2	0	1	1
1	1	1	2	2	0	1	1	1	1	1	1	1		1	2	0	0	1	0	2	2	0	0	2	1
0	1	1	1	1	0	1	1	1	1	1	0	1	…	1	1	0	0	1	2	2	2	1	0	2	2
0	2	0	2	2	0	0	0	2	0	0	0	0		1	2	1	1	0	1	2	2	1	0	1	0
0	2	0	2	2	0	0	0	2	0	0	2	2		2	1	2	2	0	1	1	1	1	1	0	0
0	0	2	1	1	0	1	1	1	1	1	0	0		2	1	2	2	0	1	1	1	1	1	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	0	2	2	2	2	2	2	0	2	2	0	2	…	2	1	2	2	0	1	1	1	1	1	0	0
0	2	0	2	2	0	0	0	2	0	0	1	1		1	1	1	1	0	0	1	1	0	1	1	1
0	1	1	2	2	0	1	1	1	1	1	0	1		1	2	1	1	1	2	2	2	1	0	1	1
⋮				⋮					⋮				⋱			⋮				⋮					
0	2	0	2	2	0	1	1	1	1	1	0	1		2	1	2	2	0	1	1	1	1	1	1	1
0	1	1	2	2	1	2	2	0	2	2	0	2		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	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	1	2	1	0	0	0	2	0	0	1	1		2	1	2	2	0	1	1	1	1	1	0	0
1	2	0	2	2	0	1	1	1	1	1	1	1		1	1	2	0	1	2	2	2	1	0	1	1
0	2	1	2	1	0	0	0	2	0	0	2	1		2	0	2	1	0	0	1	1	0	1	1	0
0	1	1	2	2	1	0	0	2	0	0	0	0	…	2	1	2	2	1	2	2	2	1	0	1	1
0	2	0	2	2	0	0	0	2	0	0	0	0		1	2	1	1	0	1	2	2	1	0	1	1
0	2	1	2	1	0	0	0	2	0	0	1	0		2	2	2	2	0	2	2	2	2	0	0	0
0	1	1	2	2	0	1	1	1	1	1	0	0		2	2	2	2	0	2	2	2	2	0	0	0
0	1	1	2	2	1	2	2	0	2	2	0	1		0	2	0	0	0	1	2	2	1	0	2	1

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

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

In [87]:
```
onlyMar = QTLMarker[:,MarkerPos];
```

In [88]:
```
QTLstream = open(QTL, "w")
Marstream = open(Mar, "w");
```

In [89]:
```
for i in 1:size(onlyID,1)
    @printf(QTLstream, "%19d", onlyID[i])
    for j in 1:size(onlyQTL,2)
        @printf(QTLstream, "%3d", onlyQTL[i,j])
    end
    @printf(QTLstream, "\n")
end
```

In [90]:
```
for i in 1:size(onlyID,1)
    @printf(Marstream, "%19d", onlyID[i])
    for j in 1:size(onlyMar,2)
        @printf(Marstream, "%3d", onlyMar[i,j])
    end
    @printf(Marstream, "\n")
end
```

In [91]:
```
close(QTLstream)
close(Marstream)
```

Remove Fixed Gene from the panel

In [92]:
```
VQM = var(QTLMarker,1)
QMnoFixed = QTLMarker[:,VQM .> 0]
VQ = var(onlyQTL,1)
QnoFixed = onlyQTL[:,VQ .> 0]
VM = var(onlyMar,1)
MnoFixed = onlyMar[:,VM .> 0];
```

In [93]:
```
GenNFstream = open(GenNF, "w")
QTLNFstream = open(QTLNF, "w")
MarNFstream = open(MarNF, "w");
```

In [94]:
```
for i in 1:size(onlyID,1)
    @printf(GenNFstream, "%19d", onlyID[i])
    for j in 1:size(QMnoFixed,2)
        @printf(GenNFstream, "%3d", QMnoFixed[i,j])
    end
    @printf(GenNFstream, "\n")
end
```

In [95]:
```
for i in 1:size(onlyID,1)
    @printf(QTLNFstream, "%19d", onlyID[i])
    for j in 1:size(QnoFixed,2)
        @printf(QTLNFstream, "%3d", QnoFixed[i,j])
    end
    @printf(QTLNFstream, "\n")
end
```

In [96]:
```
for i in 1:size(onlyID,1)
    @printf(MarNFstream, "%19d", onlyID[i])
    for j in 1:size(MnoFixed,2)
        @printf(MarNFstream, "%3d", MnoFixed[i,j])
    end
    @printf(MarNFstream, "\n")
end
```

In [97]:
```
close(GenNFstream)
close(QTLNFstream)
close(MarNFstream)
```

Check heritability

In [98]:
```
P = AllPBV[:,2]
BV = AllPBV[:,3];
```

In [99]:
```
VP = var(P)
VBV = var(BV)
H = VBV/VP
```

Out[99]: 0.6746579393283672

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

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

Out[100]:
```
0.8186102917346373
```

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

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

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

Out[102]:
```
500-element Array{Float64,1}:
 0.0638437
 0.0632424
 0.0636102
 0.0616876
 0.0621043
 0.0627512
 0.0619245
 0.0638742
 0.0629205
 0.0641572
 0.063732
 0.0650331
 0.0648231
 ⋮
 0.0633217
 0.0633257
 0.0629843
 0.0635207
 0.0638719
 0.0633207
 0.06377
 0.0632556
 0.0626605
 0.062691
 0.0628327
 0.0633465
```

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

Out[103]:
```
48000-element Array{Float64,1}:
 29.6098
 32.4822
 32.3966
 30.6963
 30.2992
 30.5634
 29.8204
 33.1687
 32.7286
 32.016
 31.8904
 31.0166
 32.3921
  ⋮
 35.071
 35.7515
 35.8755
 36.1739
 35.6088
 35.3087
 35.885
 35.1829
 33.7255
 36.0037
 35.3165
 36.387
```

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

Out[104]:
```
31.476549072494514
```

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

Out[105]:
```
32.18289055128568
```

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

Out[106]:
```
32.82708666091049
```

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

Out[107]:
```
33.41084190498907
```

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

Out[108]:
```
34.035393229846264
```

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

Out[109]:
```
34.607190822756756
```

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

Out[110]:
```
9000-element Array{Float64,1}:
 33.9289
 32.8859
 32.9822
 33.9843
 31.7859
 32.337
 34.9439
 32.8467
 31.9573
 34.2923
 32.9628
 32.4049
 34.4802
  ⋮
 35.071
 35.7515
 35.8755
 36.1739
 35.6088
 35.3087
 35.885
 35.1829
 33.7255
 36.0037
 35.3165
 36.387
```

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

Out[111]: 34.54385120749008

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

Out[112]: 3.0673021349955647

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

Out[113]: 32.903108112093065

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

Out[114]: 1.4265590395985512

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

Out[115]: 33.47034645694286

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

Out[116]: 1.9937973844483494

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

Out[117]: 33.98638553071028

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

Out[118]: 2.5098364582157657

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

Out[119]: 34.66206745474343

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

Out[120]: 3.1855183822489153

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

Out[121]: 35.16376387229378

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

Out[122]: 3.687214799799264

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

Out[123]: 34.607190822756756

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

Out[124]: 3.1306417502622423