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)
```

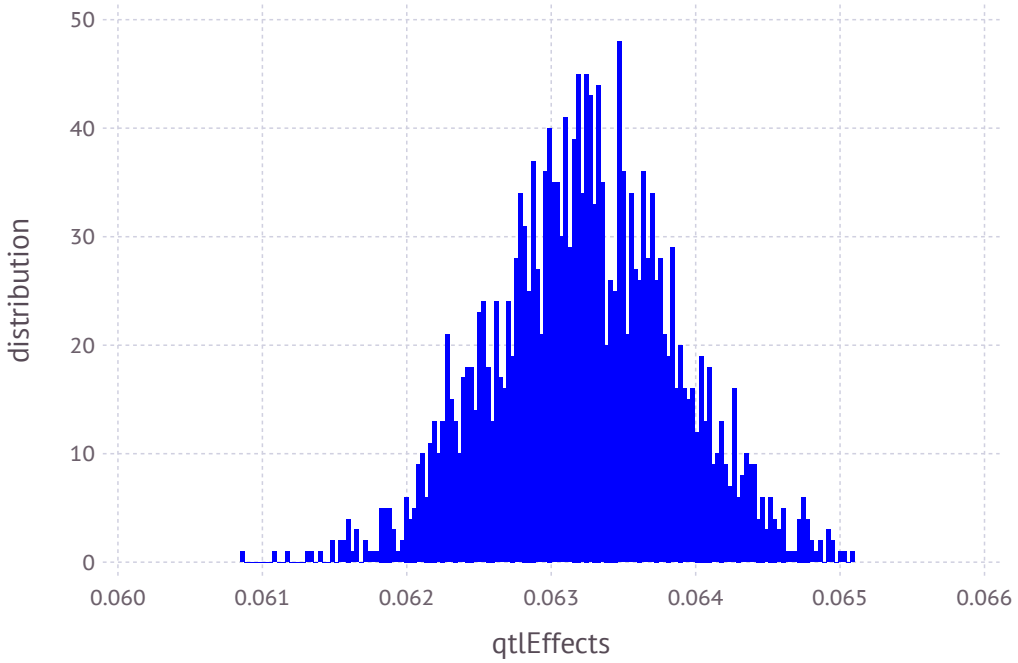
In [7]:
```
qtlEffects
```

Out[7]:
```
2000-element Array{Float64,1}:
 0.0641497
 0.0634227
 0.0638768
 0.0629071
 0.0636089
 0.0624924
 0.0632886
 0.0632091
 0.0635623
 0.0635774
 0.0632641
 0.0631666
 0.0631708
 ⋮
 0.064029
 0.0630726
 0.0632124
 0.0627917
 0.0635629
 0.0623243
 0.0635436
 0.0631643
 0.0639884
 0.0633233
 0.0622698
 0.0619774
```

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]: 0.06322466705519311

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

Out[11]: 4.088132779406068e-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
```

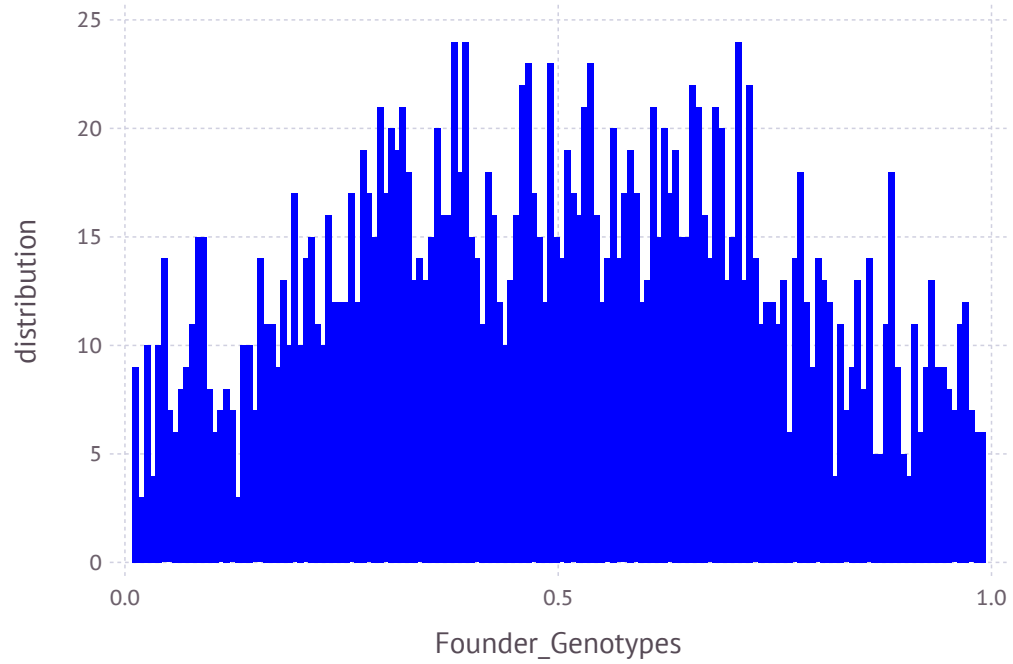
In [19]:
```
gSPSire = XSim.getOurGenotypes(popSP[1])
gSPDam = XSim.getOurGenotypes(popSP[2])
gSP = [gSPSire;gSPSire];
```

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

Out[20]:
```
1x2000 Array{Float64,2}:
 0.059375  0.85125  0.28675  0.949875  …  0.2775  0.3515  0.452125  0.278375
```

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);
```

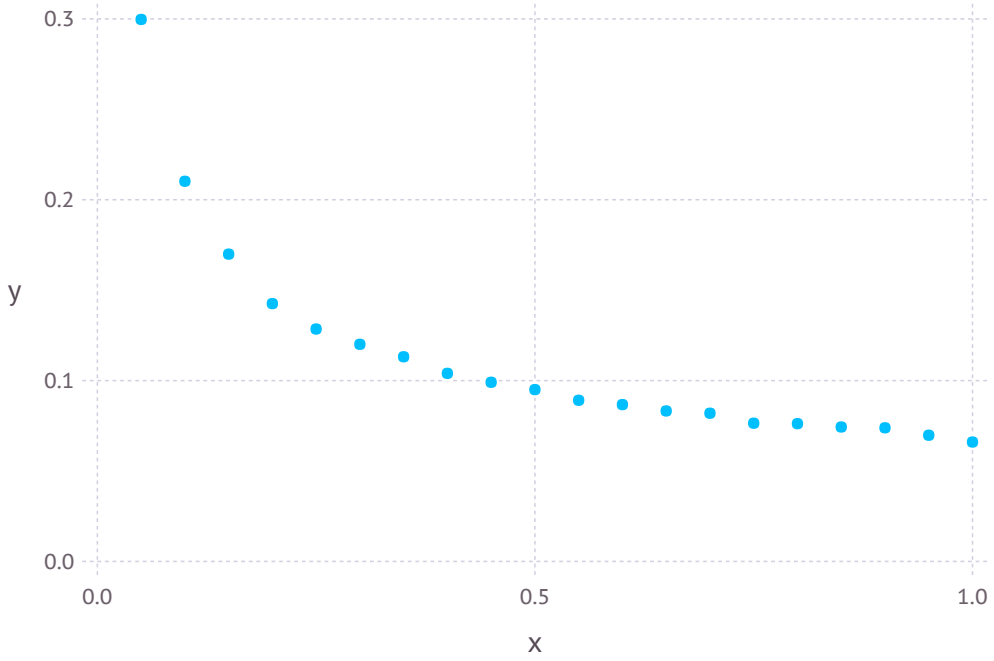
In [23]:
```
for i=1:(nRows-20)
LDMat[i,:] = corMat[i,(i+1):(i+20)].^2
end
```

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

Out[24]:
```
1x20 Array{Float64,2}:
 0.0660435  0.0698133  0.0739004  0.0743482  …  0.169952  0.210213  0.299718
```

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]:
```
31.99300847190478
```

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

Out[31]:
```
0.704445800009331
```

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

Out[32]:
```
0.704445800009331
```

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

Out[33]:
```
0.704445800009331
```

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]: 34.77460995086077

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

Out[36]: 34.762658853794605

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

Out[37]: 0.5869327326119108

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

Out[38]: 0.5678305222913852

Pedigree: All animals

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

Out[39]:
```
48000x3 Array{Int64,2}:
 40722  35036  40429
 40723  34707  40288
 40724  34275  39129
 40725  33719  40236
 40726  34776  40592
 40727  34567  39583
 40728  35609  37227
 40729  36713  39765
 40730  36562  37007
 40731  33280  38440
 40732  35153  38680
 40733  33248  40642
 40734  33586  38324
     ⋮
 88710  76058  78830
 88711  74439  78502
 88712  75897  80479
 88713  76669  78974
 88714  75988  79733
 88715  74538  78326
 88716  75220  80665
 88717  73141  77864
 88718  75366  78280
 88719  75695  80643
 88720  76587  78128
 88721  75863  78116
```

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	1	2	0	2	2	0	1	1	1	1	…	2	0	2	1	0	0	1	1	0	1	1	0
40723	0	1	1	2	2	1	1	1	1	1		1	2	1	1	0	2	2	2	2	0	0	0
40724	1	2	0	2	2	0	1	1	1	1		2	0	2	2	0	0	0	0	0	2	0	0
40725	0	2	0	2	2	0	0	0	2	0		0	2	0	0	1	2	2	2	1	0	2	2
40726	0	2	0	2	2	0	1	1	2	0		1	1	1	1	0	0	1	1	0	1	1	0
40727	0	2	1	2	1	0	0	0	2	0	…	2	0	2	2	0	0	1	1	0	1	1	0
40728	0	1	1	2	2	2	2	2	0	2		2	0	1	1	0	1	1	1	1	1	1	1
40729	0	2	0	2	2	0	0	0	2	0		1	2	1	1	0	1	2	2	1	0	1	1
40730	0	2	1	2	1	0	0	0	2	0		1	2	1	1	1	2	2	2	1	0	1	1
40731	0	2	0	2	2	0	2	2	1	1		1	1	0	0	1	2	2	2	1	0	2	2
40732	0	2	0	2	2	0	0	0	2	0	…	2	1	2	2	1	1	2	2	0	0	1	1
40733	0	2	1	2	1	0	0	0	2	0		1	1	1	1	0	0	2	2	0	0	2	1
40734	0	2	0	2	2	0	1	1	1	1		1	1	1	1	1	1	1	1	0	1	1	1
⋮			⋮				⋮		⋱		⋮					⋮					⋮		
88710	0	2	0	2	2	0	0	0	2	0		2	0	2	2	1	1	1	1	0	1	1	0
88711	0	1	1	2	2	1	1	1	1	1		2	0	2	2	0	0	0	0	0	2	0	0
88712	0	2	1	2	1	0	1	1	2	0	…	1	2	1	1	0	1	2	2	1	0	1	1
88713	0	1	1	2	2	2	2	2	0	2		0	1	0	0	0	1	1	1	1	1	1	1
88714	0	1	1	2	2	0	1	1	1	1		0	2	0	0	0	1	2	2	1	0	2	1
88715	0	1	1	2	2	2	2	2	0	2		1	1	1	1	1	1	1	1	0	1	1	1
88716	0	1	1	2	2	1	1	1	1	1		2	0	2	2	0	0	1	1	0	1	1	1
88717	0	1	2	2	1	0	1	1	1	1	…	2	2	2	2	0	2	2	2	2	0	0	0
88718	0	1	2	1	0	0	0	0	2	0		2	0	2	2	1	1	1	1	0	1	1	1
88719	0	1	2	2	1	1	1	1	1	1		2	0	2	2	0	0	0	1	0	1	1	0
88720	0	0	2	2	2	0	2	2	0	2		2	0	2	1	1	1	1	1	0	1	1	1
88721	0	1	2	2	1	0	1	1	1	1		1	1	0	1	1	1	1	1	0	1	1	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}:
 1  2  0  2  2  0  1  1  1  1  1  0  0  …  2  0  2  1  0  0  1  1  0  1  1  0
 0  1  1  2  2  1  1  1  1  1  1  1  2     1  2  1  1  0  2  2  2  2  0  0  0
 1  2  0  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  2  2  2  1  0  2  2
 0  2  0  2  2  0  1  1  2  0  0  1  1     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  2  0  0  1  1  0  1  1  0
 0  1  1  2  2  2  2  2  0  2  2  0  2     2  0  1  1  0  1  1  1  1  1  1  1
 0  2  0  2  2  0  0  0  2  0  0  1  1     1  2  1  1  0  1  2  2  1  0  1  1
 0  2  1  2  1  0  0  0  2  0  0  2  1     1  2  1  1  1  2  2  2  1  0  1  1
 0  2  0  2  2  0  2  2  1  1  1  1  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  …  2  1  2  2  1  1  2  2  0  0  1  1
 0  2  1  2  1  0  0  0  2  0  0  1  0     1  1  1  1  0  0  2  2  0  0  2  1
 0  2  0  2  2  0  1  1  1  1  1  0  1     1  1  1  1  1  1  1  1  0  1  1  1
 ⋮                 ⋮                 ⋮        ⋱        ⋮                 ⋮
 0  2  0  2  2  0  0  0  2  0  0  0  0     2  0  2  2  1  1  1  1  0  1  1  0
 0  1  1  2  2  1  1  1  1  1  1  1  2     2  0  2  2  0  0  0  0  0  2  0  0
 0  2  1  2  1  0  1  1  2  0  0  2  1  …  1  2  1  1  0  1  2  2  1  0  1  1
 0  1  1  2  2  2  2  2  0  2  2  0  1     0  1  0  0  0  1  1  1  1  1  1  1
 0  1  1  2  2  0  1  1  1  1  1  1  2     0  2  0  0  0  1  2  2  1  0  2  1
 0  1  1  2  2  2  2  2  0  2  2  0  2     1  1  1  1  1  1  1  1  0  1  1  1
 0  1  1  2  2  1  1  1  1  1  1  1  2     2  0  2  2  0  0  1  1  0  1  1  1
 0  1  2  2  1  0  1  1  1  1  1  1  0  …  2  2  2  2  0  2  2  2  2  0  0  0
 0  1  2  1  0  0  0  0  2  0  0  1  0     2  0  2  2  1  1  1  1  0  1  1  1
 0  1  2  2  1  1  1  1  1  1  1  0  1     2  0  2  2  0  0  0  1  0  1  1  0
 0  0  2  2  2  0  2  2  0  2  2  0  0     2  0  2  1  1  1  1  1  0  1  1  1
 0  1  2  2  1  0  1  1  1  1  1  1  0     1  1  0  1  1  1  1  1  0  1  1  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}:
 44413
 42888
 41734
 44637
 41310
 41851
 41456
 42731
 43790
 42401
 42613
 44047
 44345
     ⋮
 76058
 74439
 75897
 76669
 75988
 74538
 75220
 73141
 75366
 75695
 76587
 75863
```

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

Out[53]:
```
1000-element Array{Int64,1}:
 44413
 42888
 41734
 44637
 41310
 41851
 41456
 42731
 43790
 42401
 42613
 44047
 44345
     ⋮
 76296
 76149
 75486
 74292
 75532
 76170
 74598
 75995
 76292
 76046
 75482
 74439
```

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}:
 44413
 42888
 41734
 44637
 41310
 41851
 41456
 42731
 43790
 42401
 42613
 44047
 44345
     ⋮
 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  32.044  30.787
 40723  32.39   33.29
 40724  31.757  30.901
 40725  34.061  32.104
 40726  31.745  32.117
 40727  29.611  29.893
 40728  31.62   32.604
 40729  32.043  31.856
 40730  30.078  30.147
 40731  31.856  33.307
 40732  30.082  32.488
 40733  33.426  32.785
 40734  32.757  32.108
     ⋮
 88710  35.288  35.055
 88711  34.672  34.309
 88712  34.944  35.007
 88713  34.711  34.703
 88714  35.264  36.127
 88715  35.483  36.65
 88716  36.83   35.887
 88717  34.298  34.866
 88718  35.525  35.509
 88719  36.481  35.574
 88720  35.548  36.188
 88721  36.277  34.953
```

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
    6
   11
   12
   19
   20
   23
   24
   30
   32
   37
   41
   42
    ⋮
 1940
 1949
 1951
 1955
 1962
 1963
 1970
 1981
 1986
 1990
 1992
 1999
```

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

Out[83]:
```
1500-element Array{Int64,1}:
    1
    2
    3
    5
    7
    8
    9
   10
   13
   14
   15
   16
   17
    ⋮
 1985
 1987
 1988
 1989
 1991
 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	1	1	1	1	1	1	1	0	0	…	0	2	0	0	1	1	2	2	0	0	2	2
0	2	1	2	1	0	0	0	2	0	0	1	0		2	0	2	1	0	0	1	1	0	1	1	0
0	2	0	2	2	0	0	0	2	0	0	1	1		2	1	2	1	0	1	2	2	1	0	1	1
0	2	0	2	2	0	0	0	2	0	0	2	2		2	0	2	2	0	0	0	0	0	2	0	0
0	2	1	2	1	0	1	1	1	1	1	0	1		1	1	1	1	0	0	1	1	0	1	1	1
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
1	2	0	2	2	1	2	2	0	2	2	0	1		0	2	0	0	0	1	2	2	1	0	2	1
1	2	1	2	1	0	1	1	1	1	1	1	1		0	2	0	0	1	1	1	1	0	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
1	1	1	2	2	1	2	2	0	2	2	0	1		2	1	2	2	0	1	1	1	1	1	0	0
0	2	0	2	2	0	1	1	2	0	0	1	2	…	2	2	2	2	0	2	2	2	2	0	0	0
0	0	2	2	2	2	2	2	0	2	2	0	2		1	1	1	1	1	1	1	1	0	1	1	1
0	1	1	2	2	0	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	0	0	2	0	0	0	0		2	0	2	2	1	1	1	1	0	1	1	0
0	1	1	2	2	1	1	1	1	1	1	1	2		2	0	2	2	0	0	0	0	0	2	0	0
0	2	1	2	1	0	1	1	2	0	0	2	1	…	1	2	1	1	0	1	2	2	1	0	1	1
0	1	1	2	2	2	2	2	0	2	2	0	1		0	1	0	0	0	1	1	1	1	1	1	1
0	1	1	2	2	0	1	1	1	1	1	1	2		0	2	0	0	0	1	2	2	1	0	2	1
0	1	1	2	2	2	2	2	0	2	2	0	2		1	1	1	1	1	1	1	1	0	1	1	1
0	1	1	2	2	1	1	1	1	1	1	1	2		2	0	2	2	0	0	1	1	0	1	1	1
0	1	2	2	1	0	1	1	1	1	1	1	0	…	2	2	2	2	0	2	2	2	2	0	0	0
0	1	2	1	0	0	0	0	2	0	0	1	0		2	0	2	2	1	1	1	1	0	1	1	1
0	1	2	2	1	1	1	1	1	1	1	0	1		2	0	2	2	0	0	0	1	0	1	1	0
0	0	2	2	2	0	2	2	0	2	2	0	0		2	0	2	1	1	1	1	1	0	1	1	1
0	1	2	2	1	0	1	1	1	1	1	1	0		1	1	0	1	1	1	1	1	0	1	1	1

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

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

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]: 0.6670702329969379

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

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

Out[100]:
```
0.819017767821589
```

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

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

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

Out[102]:
```
500-element Array{Float64,1}:
 0.0629071
 0.0624924
 0.0632641
 0.0631666
 0.063601
 0.0632625
 0.0624318
 0.062871
 0.0643376
 0.0634743
 0.0615956
 0.0630879
 0.063156
 ⋮
 0.0636307
 0.0638227
 0.0637112
 0.0627497
 0.062045
 0.0633749
 0.0634981
 0.0632638
 0.0627278
 0.0630726
 0.0627917
 0.0622698
```

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

Out[103]:
```
48000-element Array{Float64,1}:
 30.7819
 33.3215
 30.9134
 32.1276
 32.1084
 29.9067
 32.6126
 31.8631
 30.1512
 33.3262
 32.4937
 32.8157
 32.109
  ⋮
 35.0946
 34.317
 35.0213
 34.704
 36.1639
 36.6905
 35.9087
 34.8971
 35.5226
 35.5963
 36.221
 34.9493
```

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

Out[104]:
```
32.00754039512472
```

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

Out[105]:
```
32.59314798146153
```

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

Out[106]:
```
33.197347379913175
```

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

Out[107]:
```
33.744743095109854
```

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

Out[108]:
```
34.26911803969463
```

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

Out[109]:
```
34.78690096354604
```

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

Out[110]:
```
9000-element Array{Float64,1}:
 33.8803
 31.8533
 33.3113
 33.5034
 33.3236
 33.5689
 34.46
 32.8625
 33.5294
 34.0215
 33.7004
 33.8821
 32.6777
  ⋮
 35.0946
 34.317
 35.0213
 34.704
 36.1639
 36.6905
 35.9087
 34.8971
 35.5226
 35.5963
 36.221
 34.9493
```

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

Out[111]: 34.72972236958669

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

Out[112]: 2.722181974461968

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

Out[113]: 33.16316537694624

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

Out[114]: 1.1556249818215178

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

Out[115]: 33.80191868925957

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

Out[116]: 1.7943782941348516

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

Out[117]: 34.29952939358596

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

Out[118]: 2.2919889984612425

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

Out[119]: 34.81282308693965

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

Out[120]: 2.8052826918149307

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

Out[121]: 35.28403154282826

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

Out[122]: 3.2764911477035383

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

Out[123]: 34.78690096354604

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

Out[124]: 2.7793605684213176