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; 20 loci per chromosome = > 200 Loci (50 QTL & 150 Markers)
```

In [2]:
```
include("/home/nicole/Jupyter/XSimSel.jl")
```

Out[2]: XSim

In [3]:
```
using DataFrames
```

In [4]:
```
using Distributions
```

In [5]:
```
using(Gadfly)
```

Initialize XSim

In [6]:
```
numChr    = 10
chrLength = 0.01
numLoci   = 20
nQTL      = 5
mutRate   = 0.0
locusInt  = chrLength/numLoci
mapPos    = collect(locusInt/2:locusInt:chrLength)
geneFreq  = fill(0.5,numLoci)
QTL = sample(1:numLoci,nQTL,replace=false)
qtlMarker = fill(false,numLoci)
qtlMarker[QTL] = true
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)
```

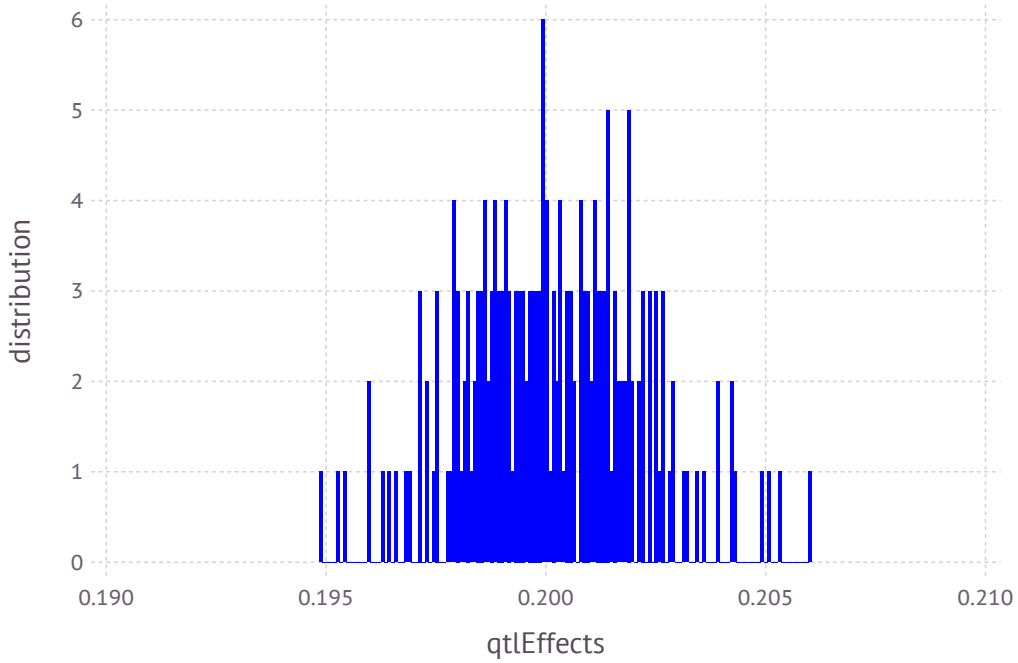
In [7]:
```
qtlEffects
```

Out[7]:
```
200-element Array{Float64,1}:
 0.201592
 0.199169
 0.201169
 0.198267
 0.199125
 0.200029
 0.197843
 0.200393
 0.198708
 0.201205
 0.200348
 0.199317
 0.20101
 ⋮
 0.201394
 0.199096
 0.194844
 0.203212
 0.202215
 0.195442
 0.196265
 0.197105
 0.196421
 0.198
 0.202648
 0.199715
```

In [8]:
```
writedlm("qtlEffects",qtlEffects)
```

In [9]:
```
plot(x=qtlEffects, Geom.histogram, Guide.XLabel("qtlEffects"), Guide.YLabel("distribution"), Theme(default_color=colorant"blue"))
```

Out[9]:



In [10]:
```
mean(qtlEffects)
```

Out[10]: 0.2001244981959468

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

Out[11]: 3.977405534690025e-6

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 generati
on
Phe = "Phe.txt"                 # phenotype file with all animals in G1 to G4
QTL = "QTL.txt"                 # QTL with with all progeny in G5 and all sires in each generation
Mar = "Mar.txt"                 # Marker with with all progeny in G5 and all sires in each generation
GenNF = "GenNF.txt"             # remove fixed genes from genotype file
QTLNF = "QTLNF.txt"             # remove fixed genes from QTL file
MarNF = "MarNF.txt"             # remove fixed genes from Marker file
;
```

```
posOFF1S: 8001; posOFF1E: 16000
posOFF2S: 16001; posOFF2E: 24000
posOFF3S: 24001; posOFF3E: 32000
posOFF4S: 32001; posOFF4E: 40000
posOFF5S: 40001; posOFF5E: 48000
```

In [14]:
```
fileName = "SS"
pedText = fileName * ".ped"
genText = fileName * ".gen"
pheText = fileName * ".phe"
;
```

In [15]:
```
;rm $pedText $genText $pheText
```

```
rm: cannot remove 'SS.ped': No such file or directory
rm: cannot remove 'SS.gen': No such file or directory
rm: cannot remove 'SS.phe': No such file or directory
```

Sampling Founders

In [16]:
```
sires = XSim.sampleFounders("/home/nicole/Jupyter/HERdata/data/Clean/newdata/ch1to10.200SNP.group1")
dams  = XSim.sampleFounders("/home/nicole/Jupyter/HERdata/data/Clean/newdata/ch1to10.200SNP.group2");
```

```
Sampling 360 animals into base population.
Sampling 361 animals into base population.
```

Base population

In [17]:
```
baseSires, baseDams, baseGen = XSim.sampleRan(popSizeBase, nGenBase, sires, dams, gen=gen);
```

```
Generation     1: sampling  4000 males and  4000 females
Generation     2: sampling  4000 males and  4000 females
Generation     3: sampling  4000 males and  4000 females
Generation     4: sampling  4000 males and  4000 females
Generation     5: sampling  4000 males and  4000 females
```

Sample animals for sire and dam candidates

In [18]:
```
popSP = XSim.sampleRan(popSizeSP, 1, baseSires, baseDams, gen=baseGen,fileName=fileName);
```

```
Generation     6: sampling  4000 males and  4000 females
```

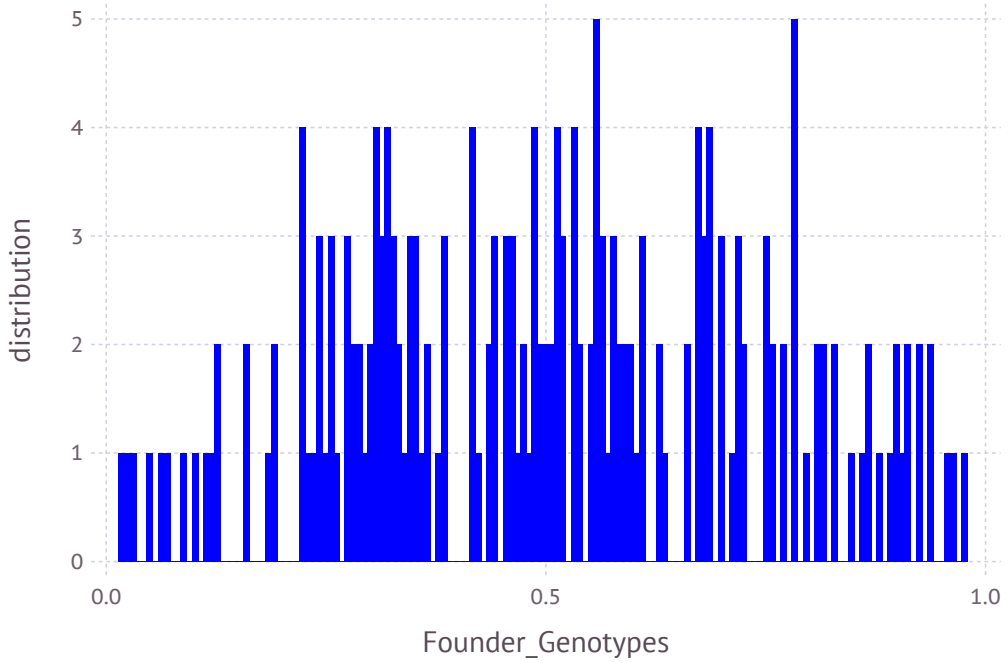
In [19]:
```
gSPSire = XSim.getOurGenotypes(popSP[1])
gSPDam = XSim.getOurGenotypes(popSP[2])
gSP = [gSPSire;gSPSire];
```

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

Out[20]:
```
1x200 Array{Float64,2}:
 0.060875  0.82825  0.301875  0.938625  …  0.376875  0.892875  0.557375
```

In [21]:
```
plot(x=FCM, Geom.histogram, Guide.XLabel("Founder_Genotypes"), Guide.YLabel("distribution"), Theme(de
fault_color=colorant"blue"))
```

Out[21]:



In [22]:
```
V=var(gSP,1)
Mark=gSP[:,V.>0]
corMat=cor(Mark)
nRows =size(corMat,1)
LDMat =zeros(nRows-1,20);
```

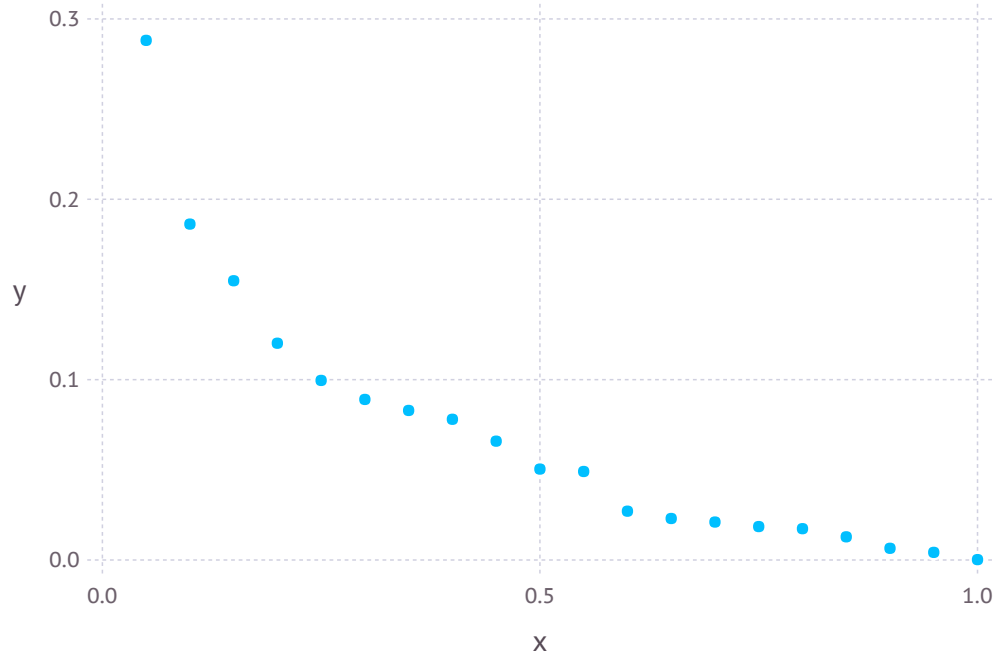
In [23]:
```
for i=1:(nRows-20)
LDMat[i,:] = corMat[i,(i+1):(i+20)].^2
end
```

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

Out[24]:
```
1x20 Array{Float64,2}:
 0.000247322  0.00425004  0.00650701  …  0.154831  0.186245  0.288139
```

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]:
```
9.845331693845425
```

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

Out[31]:
```
0.6648162972786723
```

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

Out[32]:
```
0.6648162972786723
```

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

Out[33]:
```
0.6648162972786723
```

In [34]:
```
popRMP = XSim.sampleSelNoLap(nSires, nDams, nGener,popSP[1],popSP[2],gen=popSP[3],fileName=fileName,d
irection=1.0);
```

```
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]:
```
12.316896785468929
```

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

Out[36]:
```
12.325382459605136
```

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

Out[37]:
```
0.4438285548989076
```

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

Out[38]:
```
0.4529728024227603
```

Pedigree: All animals

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

Out[39]:
```
48000x3 Array{Int64,2}:
 40722  33512  38812
 40723  35708  37485
 40724  36651  38489
 40725  33327  38821
 40726  34222  39557
 40727  34240  38702
 40728  35215  37240
 40729  35031  39203
 40730  32841  38718
 40731  33594  39879
 40732  35430  38554
 40733  35673  40206
 40734  33118  37173
     ⋮
 88710  76005  78188
 88711  76300  80186
 88712  72969  80576
 88713  75444  77454
 88714  75654  79771
 88715  74836  79662
 88716  74624  78985
 88717  76214  79556
 88718  75606  80703
 88719  75780  79126
 88720  75974  80087
 88721  74078  78956
```

In [40]:
```
PEDstream = open(PedAll, "w")
```

Out[40]:
```
IOStream(<file PedAll.txt>)
```

In [41]:
```
for i in 1:size(PED,1)
    @printf(PEDstream, "%19d%19d%19d \n", PED[i,1], PED[i,2], PED[i,3])
end
```

In [42]:
```
close(PEDstream)
```

Create matrix of ``genotype" covariates for all animals

In [43]:
```
nObs = countlines(pedText)
```

Out[43]: 48000

In [44]:
```
nMarker = numChr*numLoci
```

Out[44]: 200

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

Out[45]:
48000x201 Array{Int64,2}:

40722	0	2	0	2	2	0	1	1	2	0	…	1	1	2	1	1	1	1	1	1	1	2	1
40723	0	2	0	2	2	0	0	0	2	0		2	2	2	0	2	2	0	2	2	0	2	0
40724	1	2	0	2	2	0	0	0	2	0		2	2	2	0	2	2	0	2	2	0	2	0
40725	0	1	1	1	1	0	1	1	1	1		2	2	2	1	2	2	0	1	1	1	2	0
40726	0	2	0	2	2	0	0	0	2	0		1	1	2	1	1	1	1	1	1	1	2	1
40727	0	2	0	2	2	0	0	0	2	0	…	2	1	1	2	0	1	2	0	0	2	2	2
40728	0	2	0	2	2	0	0	0	2	0		2	1	1	1	1	2	0	2	2	0	2	0
40729	0	2	1	2	1	1	1	1	1	1		1	1	2	1	1	1	1	1	1	1	2	1
40730	0	2	1	2	1	1	1	1	1	1		1	0	1	2	0	1	1	0	0	0	2	2
40731	0	2	0	2	2	0	0	0	2	0		1	0	1	2	0	1	2	0	0	2	1	2
40732	0	1	1	1	1	0	1	1	1	1	…	2	2	2	1	2	2	0	1	1	0	2	0
40733	1	2	0	2	2	0	0	0	2	0		1	1	2	1	1	1	1	1	1	1	2	1
40734	0	1	1	2	2	1	1	1	1	1		0	0	2	2	0	0	2	0	0	2	2	2
⋮			⋮				⋮	⋱		⋮					⋮					⋮		
88710	0	2	1	2	1	0	1	2	1	1		2	1	1	2	2	2	0	0	0	0	2	2
88711	1	2	0	2	2	1	2	2	0	2		2	2	2	0	2	2	0	2	2	0	2	0
88712	1	2	0	2	2	1	2	2	0	2	…	2	2	2	2	2	2	0	0	0	2	2	1
88713	1	2	0	2	2	1	2	2	0	2		2	2	2	0	2	2	0	2	2	0	2	0
88714	1	1	1	2	2	0	2	2	0	2		1	1	2	1	1	1	1	1	1	1	2	1
88715	0	2	0	2	2	2	2	2	0	2		2	2	2	0	2	2	0	2	2	0	2	0
88716	0	1	1	1	1	1	2	2	0	2		2	2	2	0	2	2	0	2	2	0	2	0
88717	1	2	0	2	2	2	2	2	0	2	…	2	1	2	2	2	1	1	0	0	2	2	2
88718	1	1	1	2	1	1	2	2	0	2		2	2	2	0	2	2	0	2	2	0	2	0
88719	1	1	1	2	2	1	2	2	0	2		2	1	1	1	0	1	1	0	0	1	2	2
88720	0	1	2	2	1	1	1	1	1	1		2	2	2	0	2	2	0	2	2	0	2	0
88721	1	1	1	2	2	0	2	2	0	2		2	2	2	0	2	2	0	2	2	0	2	0

In [46]:
```
allID = GT[:,1];
```

In [47]:
```
GTM = GT[:,2:end];
```

Create marker file for all animals

In [48]:
```
M = GTM     # maker file for Julia
```

Out[48]:
```
48000x200 Array{Int64,2}:
 0  2  0  2  2  0  1  1  2  0  0  1  1  …  1  1  2  1  1  1  1  1  1  1  2  1
 0  2  0  2  2  0  0  0  2  0  0  1  1     2  2  2  0  2  2  0  2  2  0  2  0
 1  2  0  2  2  0  0  0  2  0  0  2  1     2  2  2  0  2  2  0  2  2  0  2  0
 0  1  1  1  1  0  1  1  1  1  1  0  1     2  2  2  1  2  2  0  1  1  1  2  0
 0  2  0  2  2  0  0  0  2  0  0  1  1     1  1  2  1  1  1  1  1  1  1  2  1
 0  2  0  2  2  0  0  0  2  0  0  2  2  …  2  1  1  2  0  1  2  0  0  2  2  2
 0  2  0  2  2  0  0  0  2  0  0  1  1     2  1  1  1  1  2  0  2  2  0  2  0
 0  2  1  2  1  1  1  1  1  1  1  1  1     1  1  2  1  1  1  1  1  1  1  2  1
 0  2  1  2  1  1  1  1  1  1  1  1  1     1  0  1  2  0  1  1  0  0  0  2  2
 0  2  0  2  2  0  0  0  2  0  0  1  1     1  0  1  2  0  1  2  0  0  2  1  2
 0  1  1  1  1  0  1  1  1  1  1  1  2  …  2  2  2  1  2  2  0  1  1  0  2  0
 1  2  0  2  2  0  0  0  2  0  0  2  1     1  1  2  1  1  1  1  1  1  1  2  1
 0  1  1  2  2  1  1  1  1  1  1  0  1     0  0  2  2  0  0  2  0  0  2  2  2
 ⋮                 ⋮                 ⋮        ⋱        ⋮                 ⋮
 0  2  1  2  1  0  1  2  1  1  1  1  2     2  1  1  2  2  2  0  0  0  0  2  2
 1  2  0  2  2  1  2  2  0  2  2  0  1     2  2  2  0  2  2  0  2  2  0  2  0
 1  2  0  2  2  1  2  2  0  2  2  0  1  …  2  2  2  2  2  2  0  0  0  2  2  1
 1  2  0  2  2  1  2  2  0  2  2  0  1     2  2  2  0  2  2  0  2  2  0  2  0
 1  1  1  2  2  0  2  2  0  2  2  0  1     1  1  2  1  1  1  1  1  1  1  2  1
 0  2  0  2  2  2  2  2  0  2  2  0  2     2  2  2  0  2  2  0  2  2  0  2  0
 0  1  1  1  1  1  2  2  0  2  2  0  2     2  2  2  0  2  2  0  2  2  0  2  0
 1  2  0  2  2  2  2  2  0  2  1  0  1  …  2  1  2  2  2  1  1  0  0  2  2  2
 1  1  1  2  1  1  2  2  0  2  2  0  0     2  2  2  0  2  2  0  2  2  0  2  0
 1  1  1  2  2  1  2  2  0  2  2  0  1     2  1  1  1  0  1  1  0  0  1  2  2
 0  1  2  2  1  1  1  1  1  1  1  1  1     2  2  2  0  2  2  0  2  2  0  2  0
 1  1  1  2  2  0  2  2  0  2  2  0  0     2  2  2  0  2  2  0  2  2  0  2  0
```

In [49]:
```
Mstream = open(GenAll, "w")
```

Out[49]:
```
IOStream(<file GenAll.txt>)
```

In [50]:
```
for i in 1:size(M,1)
    @printf(Mstream, "%19d", allID[i])
    for j in 1:size(M,2)
        @printf(Mstream, "%3d", M[i,j])
    end
    @printf(Mstream, "\n")
end
```

In [51]:
```
close(Mstream)
```

Genotypes - all sires and all offsprings in G5

In [52]:
```
AllSire = PED[posOFF1S:posOFF5E,2]
```

Out[52]:
```
40000-element Array{Int64,1}:
 43806
 43379
 42439
 41253
 43930
 43145
 42577
 41973
 41893
 42214
 43394
 43901
 42993
    ⋮
 76005
 76300
 72969
 75444
 75654
 74836
 74624
 76214
 75606
 75780
 75974
 74078
```

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

Out[53]:
```
1000-element Array{Int64,1}:
 43806
 43379
 42439
 41253
 43930
 43145
 42577
 41973
 41893
 42214
 43394
 43901
 42993
    ⋮
 73706
 75637
 76134
 76121
 76192
 75241
 74960
 75962
 75443
 74878
 75113
 74796
```

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}:
 43806
 43379
 42439
 41253
 43930
 43145
 42577
 41973
 41893
 42214
 43394
 43901
 42993
     ⋮
 88710
 88711
 88712
 88713
 88714
 88715
 88716
 88717
 88718
 88719
 88720
 88721
```

In [56]:
```
SOFF5ID= DataFrame()
SOFF5ID[:ID] = SireOFF5ID;
```

In [57]:
```
typeof(SOFF5ID)
```

Out[57]:
```
DataFrames.DataFrame
```

In [58]:
```
MT = readtable(GenAll,separator=' ',header=false);
```

In [59]:
```
rename!(MT,:x1,:ID);
```

In [60]:
```
GSOFF5 = join(SOFF5ID, MT, on = :ID, kind = :inner);
```

In [61]:
```
size(GSOFF5)
```

Out[61]: (9000,201)

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

Out[62]: 9000

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

Out[63]: 201

In [64]:
```
GSOFF5stream = open(Gen, "w")
```

Out[64]: IOStream(<file Gen.txt>)

In [65]:
```
for i in 1:size(GSOFF5,1)
    for j in 1
        @printf(GSOFF5stream, "%19d", GSOFF5[i,j])
    end
    for k in 2:size(GSOFF5,2)
        @printf(GSOFF5stream, "%3d", GSOFF5[i,k])
    end
    @printf(GSOFF5stream, "\n")
end
```

In [66]:
```
close(GSOFF5stream)
```

Phenotypes - All animals

In [67]:
```
PBV = convert(Array,readtable(pheText,separator=' ',header=false))
```

Out[67]:
```
48000x3 Array{Real,2}:
 40722   9.724  10.445
 40723   8.5    10.252
 40724  11.066  11.056
 40725   9.975  10.254
 40726  10.522  10.047
 40727  10.321   8.643
 40728   7.819   9.04
 40729   8.469   9.249
 40730   6.035   8.244
 40731   9.863   8.845
 40732   9.684  10.657
 40733  10.068  10.045
 40734  10.482   9.249
     ⋮
 88710  11.656  12.255
 88711  13.579  13.663
 88712  13.336  13.872
 88713  13.793  13.669
 88714  13.874  13.872
 88715  13.078  13.263
 88716  12.739  13.07
 88717  15.804  14.069
 88718  13.535  13.061
 88719  12.513  12.665
 88720  13.022  12.263
 88721  12.797  12.868
```

In [68]:
```
PBVstream = open(PheAll, "w")
```

Out[68]:
```
IOStream(<file PheAll.txt>)
```

In [69]:
```
for i in 1:size(PBV,1)
    @printf(PBVstream, "%19d %10.3f %10.3f \n", PBV[i,1], PBV[i,2], PBV[i,3])
end
```

In [70]:
```
close(PBVstream )
```

Phenotypes - all animnls from G0 to G4

In [71]:
```
OFFG0toG4 = PED[posOFF0S:posOFF4E,1]
```

Out[71]:
```
40000-element Array{Int64,1}:
 40722
 40723
 40724
 40725
 40726
 40727
 40728
 40729
 40730
 40731
 40732
 40733
 40734
     ⋮
 80710
 80711
 80712
 80713
 80714
 80715
 80716
 80717
 80718
 80719
 80720
 80721
```

In [72]:
```
OFFG0toG4ID= DataFrame()
OFFG0toG4ID[:ID] = OFFG0toG4;
```

In [73]:
```
typeof(OFFG0toG4ID)
```

Out[73]:
```
DataFrames.DataFrame
```

In [74]:
```
AllPBV= readtable(pheText,separator=' ',header=false);
```

In [75]:
```
rename!(AllPBV,:x1,:ID)
head(AllPBV);
```

In [76]:
```
OFFG0toG4PBV = join(OFFG0toG4ID, AllPBV, on = :ID, kind = :inner);
```

In [77]:
```
Row = size(OFFG0toG4PBV,1)
```

Out[77]:
```
40000
```

In [78]:
```
Col = size(OFFG0toG4PBV,2)
```

Out[78]:
```
3
```

In [79]:
```
Phestream = open(Phe, "w")
```

Out[79]:
```
IOStream(<file Phe.txt>)
```

In [80]:
```
for i in 1:size(OFFG0toG4PBV,1)
    @printf(Phestream, "%19d %10.3f %10.3f \n", OFFG0toG4PBV[i,1], OFFG0toG4PBV[i,2],
OFFG0toG4PBV[i,3])
end
```

In [81]:
```
close(Phestream)
```

Get files with QTL only or Markers only

QTL file and Markers file

In [82]:
```
QTLPos = XSim.common.G.qtl_index
```

Out[82]:
```
50-element Array{Int64,1}:
   1
   2
   8
  10
  19
  21
  22
  28
  30
  39
  41
  42
  48
   ⋮
 150
 159
 161
 162
 168
 170
 179
 181
 182
 188
 190
 199
```

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

Out[83]:
```
150-element Array{Int64,1}:
   3
   4
   5
   6
   7
   9
  11
  12
  13
  14
  15
  16
  17
   ⋮
 186
 187
 189
 191
 192
 193
 194
 195
 196
 197
 198
 200
```

Genotype codes: 0, 1, 2

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

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

Out[85]:
```
9000x200 Array{Int64,2}:
 0  2  1  2  1  0  0  0  2  0  0  1  0  …  2  2  2  1  1  1  1  1  1  1  2  1
 0  2  0  2  2  1  1  1  1  1  1  1  2     1  1  2  1  1  1  1  1  1  1  2  1
 1  2  0  2  2  0  1  1  1  1  1  1  1     2  1  1  1  1  2  0  1  1  0  2  1
 0  1  1  1  1  0  2  2  0  2  2  0  2     2  1  2  1  1  2  0  1  1  0  2  1
 0  2  1  2  1  0  0  0  2  0  0  2  1     2  2  1  1  1  1  1  1  1  1  2  1
 0  2  0  2  2  0  0  0  2  0  0  1  1  …  2  1  1  1  1  2  1  1  1  1  1  1
 0  2  0  2  2  2  2  2  0  2  1  0  1     2  1  2  0  2  2  0  2  2  0  2  0
 0  1  1  2  1  1  1  1  1  1  1  0  0     2  2  2  0  2  2  0  2  2  0  2  0
 1  1  1  1  1  0  1  1  1  1  1  1  1     2  2  1  2  2  2  0  0  0  1  2  2
 0  2  0  2  2  0  1  1  2  0  0  2  2     1  1  2  1  1  0  1  0  0  1  2  2
 0  2  1  2  1  0  0  0  2  0  0  1  0  …  2  2  2  0  2  2  0  2  2  0  2  0
 0  1  2  2  1  0  1  1  1  1  1  1  1     2  2  2  0  2  2  0  2  2  0  2  0
 0  1  1  2  1  2  2  2  0  2  2  0  1     2  1  1  1  2  2  0  1  1  1  2  0
 ⋮                 ⋮                 ⋮        ⋱        ⋮                 ⋮
 0  2  1  2  1  0  1  2  1  1  1  1  2     2  1  1  2  2  2  0  0  0  0  2  2
 1  2  0  2  2  1  2  2  0  2  2  0  1     2  2  2  0  2  2  0  2  2  0  2  0
 1  2  0  2  2  1  2  2  0  2  2  0  1  …  2  2  2  2  2  2  0  0  0  2  2  1
 1  2  0  2  2  1  2  2  0  2  2  0  1     2  2  2  0  2  2  0  2  2  0  2  0
 1  1  1  2  2  0  2  2  0  2  2  0  1     1  1  2  1  1  1  1  1  1  1  2  1
 0  2  0  2  2  2  2  2  0  2  2  0  2     2  2  2  0  2  2  0  2  2  0  2  0
 0  1  1  1  1  1  2  2  0  2  2  0  2     2  2  2  0  2  2  0  2  2  0  2  0
 1  2  0  2  2  2  2  2  0  2  1  0  1  …  2  1  2  2  2  1  1  0  0  2  2  2
 1  1  1  2  1  1  2  2  0  2  2  0  0     2  2  2  0  2  2  0  2  2  0  2  0
 1  1  1  2  2  1  2  2  0  2  2  0  1     2  1  1  1  0  1  1  0  0  1  2  2
 0  1  2  2  1  1  1  1  1  1  1  1  1     2  2  2  0  2  2  0  2  2  0  2  0
 1  1  1  2  2  0  2  2  0  2  2  0  0     2  2  2  0  2  2  0  2  2  0  2  0
```

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

Out[86]:
```
9000x50 Array{Int64,2}:
 0  2  0  0  0  0  0  2  0  0  2  0  0  …  1  2  1  1  2  2  1  2  1  1  2  2
 0  2  1  1  0  2  1  0  0  0  1  1  1     1  1  1  0  2  2  0  2  0  1  1  2
 1  2  1  1  1  1  1  1  0  0  2  0  1     0  2  1  0  2  2  1  2  0  1  1  2
 0  1  2  2  0  1  0  0  1  1  2  0  2     1  0  1  0  2  2  0  2  1  1  1  2
 0  2  0  0  0  2  2  0  0  2  2  0  2     0  2  2  0  2  2  1  2  0  1  2  2
 0  2  0  0  0  2  2  0  0  1  2  0  2  …  0  2  2  0  2  2  1  1  1  1  1  1
 0  2  2  2  0  1  2  1  0  2  1  1  1     1  0  1  1  1  2  1  2  0  2  1  2
 0  1  1  1  0  2  1  0  0  1  0  2  0     2  1  2  0  2  2  0  2  0  2  2  2
 1  1  1  1  0  2  0  0  0  1  1  1  1     0  2  1  1  2  2  1  1  1  0  2  2
 0  2  1  0  0  1  1  1  0  1  0  2  0     1  1  1  1  1  1  1  2  0  1  1  2
 0  2  0  0  0  1  0  0  0  1  1  1  1  …  0  1  1  0  2  2  2  2  0  2  2  2
 0  1  1  1  0  1  1  1  0  1  2  0  1     1  2  0  0  2  2  2  2  0  2  2  2
 0  1  2  2  0  1  1  0  0  2  2  0  2     2  1  1  1  1  1  1  2  1  1  1  2
 ⋮              ⋮              ⋮          ⋱        ⋮              ⋮
 0  2  2  1  0  1  2  2  0  0  2  0  2     0  2  1  2  2  1  0  2  2  0  1  2
 1  2  2  2  1  1  1  1  1  2  0  2  1     2  2  1  1  2  2  1  2  0  2  2  2
 1  2  2  2  0  2  2  0  0  2  2  0  2  …  2  1  2  0  2  2  0  2  1  1  2  2
 1  2  2  2  1  2  2  1  0  2  1  1  0     2  2  1  0  2  2  2  2  0  2  2  2
 1  1  2  2  2  2  1  1  0  1  0  2  0     2  2  2  0  2  2  0  2  0  1  1  2
 0  2  2  2  0  2  1  0  0  2  0  2  0     1  2  1  1  2  2  1  2  0  2  2  2
 0  1  2  2  0  2  2  0  0  2  1  1  1     1  2  1  1  2  2  0  2  0  2  2  2
 1  2  2  2  0  2  2  1  0  2  0  2  0  …  1  2  1  1  2  2  2  2  1  1  1  2
 1  1  2  2  1  2  1  0  0  2  1  2  1     2  1  1  1  2  1  1  2  1  2  2  2
 1  1  2  2  1  2  1  0  0  1  0  2  1     1  2  2  1  2  2  0  2  0  1  1  2
 0  1  1  1  0  2  2  0  0  1  1  1  1     1  2  2  0  2  1  1  2  0  2  2  2
 1  1  2  2  0  2  1  1  0  0  0  2  0     1  2  1  0  2  2  2  2  0  2  2  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]: 0.6360031101069901

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

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

Out[100]: 0.7937071749206932

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

Out[101]:
```
48000x50 Array{Int64,2}:
 0  2  1  0  0  2  2  0  0  1  0  2  0  …  2  2  0  1  2  1  1  2  0  1  1  2
 0  2  0  0  0  1  1  1  0  1  1  1  0     1  1  2  0  2  2  1  2  0  2  2  2
 1  2  0  0  0  2  1  0  0  0  0  2  0     1  1  1  1  2  2  2  2  0  2  2  2
 0  1  1  1  0  1  2  1  0  0  1  1  1     2  2  2  0  2  2  1  2  0  2  2  2
 0  2  0  0  0  2  0  1  0  0  2  0  2     0  1  0  2  2  1  0  2  1  2  1  2
 0  2  0  0  0  1  0  0  0  0  0  2  0  …  0  1  1  0  1  2  1  2  0  1  1  2
 0  2  0  0  0  0  1  1  0  0  1  1  0     2  2  1  1  1  2  1  2  0  1  1  2
 0  2  1  1  0  1  0  0  0  0  1  1  0     2  2  1  0  2  2  1  2  0  1  1  2
 0  2  1  1  0  1  0  1  0  0  2  0  0     1  1  2  0  0  0  0  2  0  0  0  2
 0  2  0  0  0  1  1  0  0  1  2  1  0     2  1  1  0  2  2  1  1  1  0  0  1
 0  1  1  1  0  2  0  0  0  1  2  0  0  …  2  0  1  1  2  1  1  2  1  1  2  2
 1  2  0  0  0  2  1  0  0  1  1  1  1     1  1  1  0  2  1  0  2  0  1  1  2
 0  1  1  1  0  2  1  0  0  1  2  0  1     0  2  1  0  2  1  0  2  0  0  0  2
 ⋮                 ⋮                 ⋮        ⋱        ⋮                 ⋮
 0  2  2  1  0  1  2  2  0  0  2  0  2     0  2  1  2  2  1  0  2  2  0  1  2
 1  2  2  2  1  1  1  1  1  2  0  2  1     2  2  1  1  2  2  1  2  0  2  2  2
 1  2  2  2  0  2  2  0  0  2  2  0  2  …  2  1  2  0  2  2  0  2  1  1  2  2
 1  2  2  2  1  2  2  1  0  2  1  1  0     2  2  1  0  2  2  2  2  0  2  2  2
 1  1  2  2  2  2  1  1  0  1  0  2  0     2  2  2  0  2  2  0  2  0  1  1  2
 0  2  2  2  0  2  1  0  0  2  0  2  0     1  2  1  1  2  2  1  2  0  2  2  2
 0  1  2  2  0  2  2  0  0  2  1  1  1     1  2  1  1  2  2  0  2  0  2  2  2
 1  2  2  2  0  2  2  1  0  2  0  2  0  …  1  2  1  1  2  2  2  2  1  1  1  2
 1  1  2  2  1  2  1  0  0  2  1  2  1     2  1  1  1  2  1  1  2  1  2  2  2
 1  1  2  2  1  2  1  0  0  1  0  2  1     1  2  2  1  2  2  0  2  0  1  1  2
 0  1  1  1  0  2  2  0  0  1  1  1  1     1  2  2  0  2  1  1  2  0  2  2  2
 1  1  2  2  0  2  1  1  0  0  0  2  0     1  2  1  0  2  2  2  2  0  2  2  2
```

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

Out[102]:
```
50-element Array{Float64,1}:
 0.201592
 0.199169
 0.200393
 0.201205
 0.202108
 0.198219
 0.204301
 0.202673
 0.197961
 0.198171
 0.200886
 0.202485
 0.200326
 ⋮
 0.198885
 0.200808
 0.198815
 0.197516
 0.199043
 0.198956
 0.200168
 0.199049
 0.199736
 0.19835
 0.199096
 0.202648
```

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

Out[103]:
```
48000-element Array{Float64,1}:
 10.4111
 10.2002
 10.9977
 10.2091
  9.98639
  8.60257
  9.00496
  9.1981
  8.225
  8.80992
 10.5951
 10.0093
  9.20919
  ⋮
 12.2199
 13.6137
 13.806
 13.6169
 13.8184
 13.2086
 13.0043
 14.0122
 13.0159
 12.6108
 12.2095
 12.8111
```

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

Out[104]:
```
9.806113667002249
```

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

Out[105]:
```
10.362460649277244
```

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

Out[106]:
```
10.901082316193191
```

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

Out[107]:
```
11.37566048346864
```

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

Out[108]:
```
11.822493524739034
```

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

Out[109]:
```
12.270314789905953
```

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

Out[110]:
```
9000-element Array{Float64,1}:
 11.0048
 11.2148
 10.6124
 10.8006
 11.3998
 11.0052
 11.0157
 10.8075
 11.4014
 11.2093
 10.4078
 10.2009
 11.1982
  ⋮
 12.2199
 13.6137
 13.806
 13.6169
 13.8184
 13.2086
 13.0043
 14.0122
 13.0159
 12.6108
 12.2095
 12.8111
```

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

Out[111]: 12.222404674798467

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

Out[112]: 2.416291007796218

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

Out[113]: 10.916227649763425

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

Out[114]: 1.1101139827611757

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

Out[115]: 11.417158092936528

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

Out[116]: 1.6110444259342795

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

Out[117]: 11.863526912460465

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

Out[118]: 2.057413245458216

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

Out[119]: 12.290174018695925

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

Out[120]: 2.4840603516936763

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

Out[121]: 12.708532095836565

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

Out[122]: 2.9024184288343164

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

Out[123]: 12.270314789905953

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

Out[124]: 2.464201122903704