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)
```

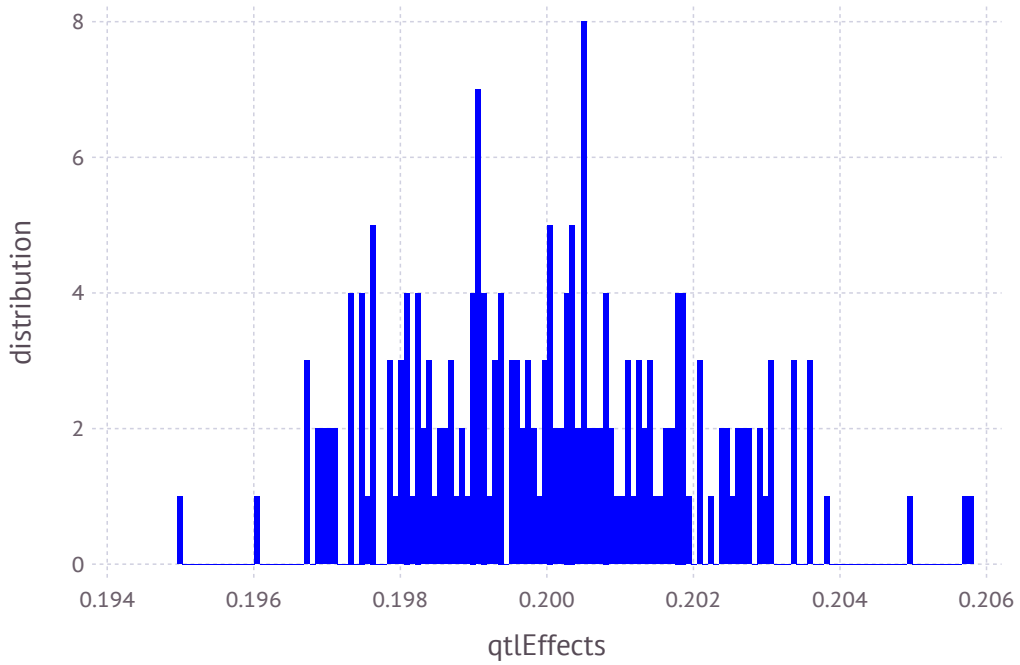
In [7]:
```
qtlEffects
```

Out[7]:
```
200-element Array{Float64,1}:
 0.200046
 0.199728
 0.201192
 0.199129
 0.203615
 0.200374
 0.20002
 0.201853
 0.20007
 0.201394
 0.199295
 0.197918
 0.198711
 ⋮
 0.197314
 0.200308
 0.202927
 0.201776
 0.198087
 0.199026
 0.199517
 0.201636
 0.198231
 0.198344
 0.196947
 0.200561
```

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]:
```
0.19999778925035375
```

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

Out[11]:
```
3.756160595758531e-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 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/ch1to10.200SNP.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
```

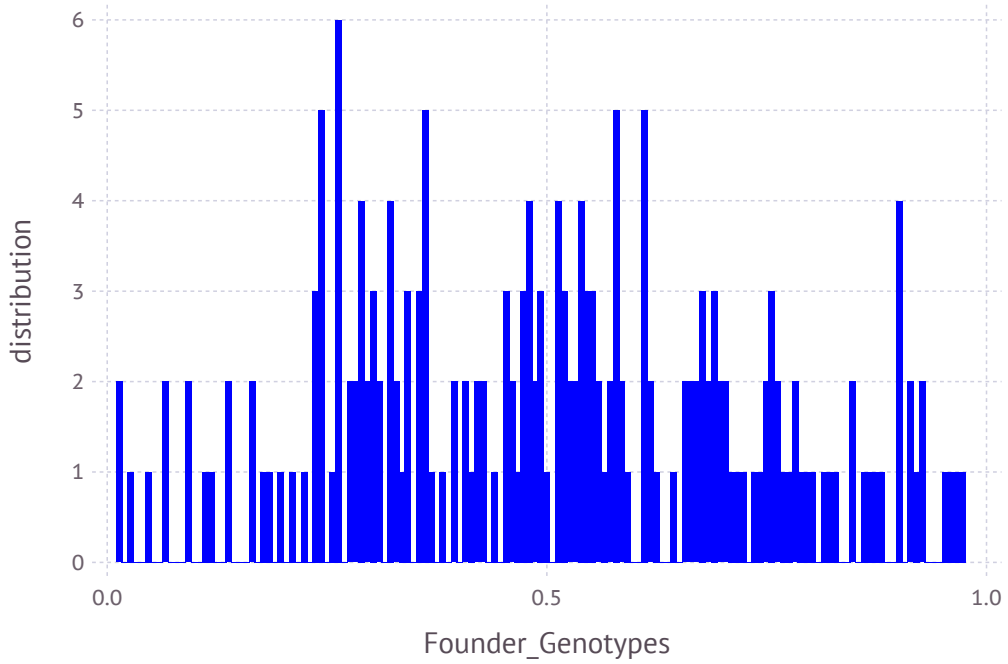
In [19]:
```
gSPSire = XSim.getOurGenotypes(popSP[1])
gSPDam = XSim.getOurGenotypes(popSP[2])
gSP = [gSPSire;gSPSire];
```

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

Out[20]:
```
1x200 Array{Float64,2}:
 0.063  0.849875  0.281  0.952125  …  0.3625  0.39625  0.898875  0.54375
```

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);
```

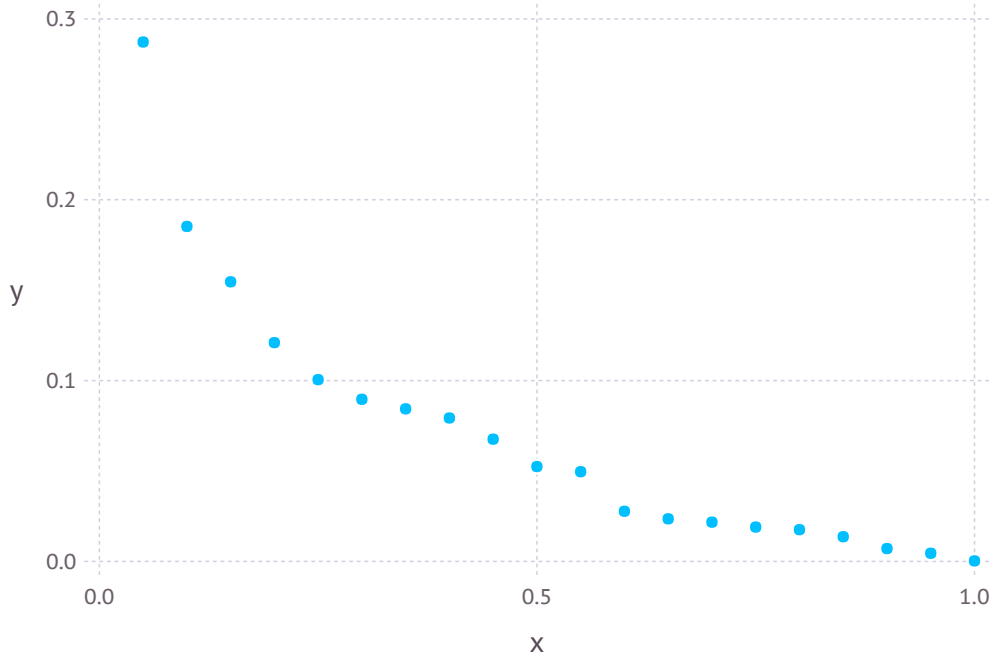
In [23]:
```
for i=1:(nRows-20)
LDMat[i,:] = corMat[i,(i+1):(i+20)].^2
end
```

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

Out[24]:
```
1x20 Array{Float64,2}:
 0.000306128  0.00455929  0.0071286  …  0.121  0.154653  0.185273  0.28725
```

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]:
```
9.951135313022682
```

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

Out[31]:
```
0.6271698888104651
```

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

Out[32]:
```
0.6271698888104651
```

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

Out[33]:
```
0.6271698888104651
```

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]: 12.3187204148105

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

Out[36]: 12.300432763818902

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

Out[37]: 0.36995896578219856

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

Out[38]: 0.3854324423079384

Pedigree: All animals

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

Out[39]:
```
48000x3 Array{Int64,2}:
 40722  33619  39179
 40723  33153  38935
 40724  33437  39896
 40725  34580  36746
 40726  32817  37807
 40727  36113  38674
 40728  33042  38310
 40729  35244  40247
 40730  35440  37167
 40731  33329  39199
 40732  33752  40220
 40733  36087  38216
 40734  35659  40283
     ⋮
 88710  74528  77739
 88711  76647  78689
 88712  75592  80381
 88713  75591  79340
 88714  74676  79685
 88715  75607  80479
 88716  74824  80075
 88717  75631  78507
 88718  74312  80603
 88719  73624  77765
 88720  76483  79682
 88721  75037  80212
```

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	1	2	1	0	0	0	2	0	…	1	1	2	1	1	1	1	1	1	1	2	1
40723	0	2	1	2	1	0	0	0	2	0		1	0	1	2	0	2	1	0	0	1	0	2
40724	0	2	0	2	2	1	1	1	1	1		2	2	2	0	2	2	0	2	2	0	2	0
40725	0	1	1	2	2	1	1	1	1	1		2	1	1	1	1	2	0	1	1	0	2	1
40726	0	2	0	2	2	0	0	0	2	0		2	1	0	2	2	2	0	0	0	1	2	1
40727	1	2	0	2	2	0	1	1	1	1	…	1	1	2	1	1	2	0	1	1	0	1	1
40728	0	2	1	2	1	0	0	0	2	0		2	2	2	0	2	2	0	2	2	0	2	0
40729	0	2	2	2	0	0	0	0	2	0		2	1	1	2	0	1	0	0	0	0	0	2
40730	0	1	1	1	1	0	1	1	1	1		2	2	1	1	2	2	0	1	1	0	2	1
40731	0	1	1	1	1	0	1	1	1	1		2	2	1	2	2	2	0	0	0	0	2	1
40732	0	2	2	2	0	1	1	1	1	1	…	1	0	1	2	0	1	1	0	0	1	2	2
40733	0	2	0	2	2	0	0	0	2	0		0	0	2	2	0	0	2	0	0	2	2	2
40734	1	1	1	2	2	1	2	2	0	2		1	0	1	2	0	1	1	0	0	1	2	2
⋮				⋮				⋮	⋱			⋮					⋮					⋮	
88710	1	1	1	2	2	0	2	2	0	2		2	1	1	1	2	2	0	1	1	1	2	0
88711	1	1	1	1	2	1	2	2	0	2		2	1	1	2	2	2	0	0	0	1	2	0
88712	0	0	2	1	1	0	2	2	0	2	…	2	2	2	1	2	2	0	1	1	1	2	0
88713	1	1	1	1	1	0	2	2	0	2		2	1	2	1	1	1	0	1	1	0	2	1
88714	1	2	0	2	2	0	2	2	0	2		1	2	2	1	1	2	0	1	1	1	1	1
88715	1	1	1	2	2	0	2	2	0	2		2	2	2	0	2	2	0	2	2	0	2	0
88716	0	1	1	2	2	0	2	2	1	1		2	2	2	0	2	2	0	2	2	0	2	0
88717	0	0	2	2	2	0	2	2	0	2	…	2	1	1	1	2	2	0	1	1	1	2	0
88718	0	2	0	2	2	2	2	2	0	2		2	2	2	0	2	2	1	1	1	1	2	1
88719	0	0	2	2	2	1	2	2	0	2		2	2	2	1	2	2	0	1	1	1	2	0
88720	1	2	0	2	2	0	1	1	1	1		0	0	2	2	0	0	2	0	0	2	2	2
88721	0	0	2	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  1  2  1  0  0  0  2  0  0  1  0  …  1  1  2  1  1  1  1  1  1  1  2  1
 0  2  1  2  1  0  0  0  2  0  0  2  1     1  0  1  2  0  2  1  0  0  1  0  2
 0  2  0  2  2  1  1  1  1  1  1  0  1     2  2  2  0  2  2  0  2  2  0  2  0
 0  1  1  2  2  1  1  1  1  1  1  1  2     2  1  1  1  1  2  0  1  1  0  2  1
 0  2  0  2  2  0  0  0  2  0  0  1  1     2  1  0  2  2  2  0  0  0  1  2  1
 1  2  0  2  2  0  1  1  1  1  1  1  1  …  1  1  2  1  1  2  0  1  1  0  1  1
 0  2  1  2  1  0  0  0  2  0  0  2  1     2  2  2  0  2  2  0  2  2  0  2  0
 0  2  2  2  0  0  0  0  2  0  0  2  0     2  1  1  2  0  1  0  0  0  0  0  2
 0  1  1  1  1  0  1  1  1  1  1  1  2     2  2  1  1  2  2  0  1  1  0  2  1
 0  1  1  1  1  0  1  1  1  1  1  0  1     2  2  1  2  2  2  0  0  0  0  2  1
 0  2  2  2  0  1  1  1  1  1  1  1  0  …  1  0  1  2  0  1  1  0  0  1  2  2
 0  2  0  2  2  0  0  0  2  0  0  1  1     0  0  2  2  0  0  2  0  0  2  2  2
 1  1  1  2  2  1  2  2  0  2  2  0  2     1  0  1  2  0  1  1  0  0  1  2  2
 ⋮              ⋮              ⋮        ⋱     ⋮              ⋮
 1  1  1  2  2  0  2  2  0  2  2  0  1     2  1  1  1  2  2  0  1  1  1  2  0
 1  1  1  1  2  1  2  2  0  2  2  0  2     2  1  1  2  2  2  0  0  0  1  2  0
 0  0  2  1  1  0  2  2  0  2  2  1  1  …  2  2  2  1  2  2  0  1  1  1  2  0
 1  1  1  1  1  0  2  2  0  2  2  0  1     2  1  2  1  1  1  0  1  1  0  2  1
 1  2  0  2  2  0  2  2  0  2  2  0  2     1  2  2  1  1  2  0  1  1  1  1  1
 1  1  1  2  2  0  2  2  0  2  2  0  1     2  2  2  0  2  2  0  2  2  0  2  0
 0  1  1  2  2  0  2  2  1  1  1  1  2     2  2  2  0  2  2  0  2  2  0  2  0
 0  0  2  2  2  0  2  2  0  2  2  0  2  …  2  1  1  1  2  2  0  1  1  1  2  0
 0  2  0  2  2  2  2  2  0  2  2  0  2     2  2  2  0  2  2  1  1  1  1  2  1
 0  0  2  2  2  1  2  2  0  2  2  0  2     2  2  2  1  2  2  0  1  1  1  2  0
 1  2  0  2  2  0  1  1  1  1  1  1  1     0  0  2  2  0  0  2  0  0  2  2  2
 0  0  2  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}:
 42922
 42747
 40870
 44544
 41523
 42921
 42068
 44019
 40956
 44574
 41608
 41441
 40728
     ⋮
 74528
 76647
 75592
 75591
 74676
 75607
 74824
 75631
 74312
 73624
 76483
 75037
```

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

Out[53]:
```
1000-element Array{Int64,1}:
 42922
 42747
 40870
 44544
 41523
 42921
 42068
 44019
 40956
 44574
 41608
 41441
 40728
     ⋮
 72840
 76213
 75289
 75901
 75254
 76159
 76169
 75859
 75592
 73055
 74899
 76339
```

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}:
 42922
 42747
 40870
 44544
 41523
 42921
 42068
 44019
 40956
 44574
 41608
 41441
 40728
     ⋮
 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  10.126  10.091
 40723   8.036   7.881
 40724  11.502  11.085
 40725   9.146  10.091
 40726   8.645   9.491
 40727  10.091   9.892
 40728  12.707  11.109
 40729   7.167   8.275
 40730   7.23    8.874
 40731   8.795  10.087
 40732   8.692   8.878
 40733   8.341   9.086
 40734   9.841  10.289
     ⋮
 88710  12.569  13.506
 88711  12.466  13.115
 88712  12.975  13.117
 88713  12.281  12.518
 88714  12.597  12.908
 88715  12.162  12.909
 88716  12.104  12.304
 88717  13.509  12.905
 88718  13.996  13.721
 88719  14.312  13.112
 88720  13.085  12.911
 88721  11.972  13.309
```

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}:
   5
   8
  10
  11
  18
  25
  28
  30
  31
  38
  45
  48
  50
   ⋮
 151
 158
 165
 168
 170
 171
 178
 185
 188
 190
 191
 198
```

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

Out[83]:
```
150-element Array{Int64,1}:
   1
   2
   3
   4
   6
   7
   9
  12
  13
  14
  15
  16
  17
   ⋮
 184
 186
 187
 189
 192
 193
 194
 195
 196
 197
 199
 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	2	1	…	2	2	2	0	2	2	0	2	2	0	2	0
0	2	0	2	2	0	0	0	2	0	0	2	1		1	0	1	2	0	1	1	0	0	1	2	2
0	2	0	2	2	0	0	0	2	0	0	1	1		2	1	2	1	2	1	1	1	1	1	2	1
1	2	0	2	2	1	1	1	1	1	1	1	1		1	1	1	2	2	2	0	1	1	0	2	1
1	1	1	2	2	0	2	2	0	2	2	0	1		2	1	1	1	2	2	0	1	1	1	2	0
0	2	0	2	2	1	2	2	1	1	1	1	2	…	1	0	2	1	1	1	1	1	1	1	2	1
0	2	0	2	2	1	1	1	1	1	1	1	2		2	2	2	0	2	2	0	2	2	0	2	0
0	2	0	2	2	0	0	0	2	0	0	0	0		1	1	2	1	1	1	1	1	1	1	2	1
0	1	1	2	2	0	1	1	1	1	1	0	1		2	2	2	1	2	2	0	1	1	1	2	1
0	1	1	1	1	0	1	1	1	1	1	0	1		2	2	2	0	2	2	0	2	2	0	2	0
0	1	1	1	1	0	1	1	1	1	1	1	2	…	2	2	2	0	1	1	1	1	1	1	2	1
0	2	1	2	1	0	0	0	2	0	0	2	1		1	0	2	2	1	0	1	0	0	1	2	2
1	1	1	1	1	0	1	1	1	1	1	1	1		1	1	1	2	0	0	2	0	0	2	2	2
⋮				⋮					⋮				⋱			⋮					⋮				
1	1	1	2	2	0	2	2	0	2	2	0	1		2	1	1	1	2	2	0	1	1	1	2	0
1	1	1	1	2	1	2	2	0	2	2	0	2		2	1	1	2	2	2	0	0	0	1	2	0
0	0	2	1	1	0	2	2	0	2	2	1	1	…	2	2	2	1	2	2	0	1	1	1	2	0
1	1	1	1	1	0	2	2	0	2	2	0	1		2	1	2	1	1	1	0	1	1	0	2	1
1	2	0	2	2	0	2	2	0	2	2	0	2		1	2	2	1	1	2	0	1	1	1	1	1
1	1	1	2	2	0	2	2	0	2	2	0	1		2	2	2	0	2	2	0	2	2	0	2	0
0	1	1	2	2	0	2	2	1	1	1	1	2		2	2	2	0	2	2	0	2	2	0	2	0
0	0	2	2	2	0	2	2	0	2	2	0	2	…	2	1	1	1	2	2	0	1	1	1	2	0
0	2	0	2	2	2	2	2	0	2	2	0	2		2	2	2	0	2	2	1	1	1	1	2	1
0	0	2	2	2	1	2	2	0	2	2	0	2		2	2	2	1	2	2	0	1	1	1	2	0
1	2	0	2	2	0	1	1	1	1	1	1	1		0	0	2	2	0	0	2	0	0	2	2	2
0	0	2	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}:
 1  0  0  0  2  2  2  0  0  1  1  2  0  …  1  1  1  2  1  0  1  1  1  2  2  0
 2  0  0  0  1  1  1  0  0  2  1  2  0     2  0  0  2  2  0  2  2  0  0  1  1
 2  0  0  0  1  0  0  0  2  2  1  1  1     2  0  0  2  1  1  2  0  2  1  2  1
 2  1  1  1  1  2  1  0  0  2  1  1  1     1  1  0  2  2  1  1  2  0  1  1  0
 2  2  2  2  1  1  2  0  0  1  2  1  1     1  2  1  1  1  0  1  1  1  1  1  1
 2  2  1  1  1  2  0  0  1  2  2  0  2  …  1  0  0  2  2  1  1  1  1  0  2  1
 2  1  1  1  1  1  1  0  0  1  1  1  1     1  2  1  2  1  1  1  1  1  2  2  0
 2  0  0  0  0  2  0  0  1  1  2  0  2     2  0  1  2  2  1  0  1  1  1  2  1
 2  1  1  1  0  2  2  0  0  1  0  2  0     2  2  0  2  1  1  1  1  1  2  2  1
 1  1  1  1  0  1  1  0  0  1  1  0  2     1  1  0  2  2  2  0  0  2  2  2  0
 1  1  1  1  1  1  2  0  0  1  1  0  2  …  0  2  1  2  1  1  0  0  2  2  2  1
 1  0  0  0  2  0  0  0  0  2  2  0  2     2  1  0  1  0  2  2  1  0  0  2  1
 1  1  1  1  0  1  0  0  0  2  1  1  1     0  2  0  2  1  1  2  2  0  1  1  2
 ⋮              ⋮              ⋮        ⋱     ⋮              ⋮
 2  2  2  2  1  2  0  0  2  2  2  0  2     2  2  0  2  2  2  1  0  2  1  1  1
 2  2  2  2  2  1  0  0  2  2  1  0  2     2  1  1  2  2  0  1  1  1  1  1  1
 1  2  2  2  0  2  2  0  0  1  1  0  2  …  1  1  0  2  2  1  1  1  2  2  2  1
 1  2  2  2  0  2  0  0  0  2  2  0  2     1  1  1  2  2  2  1  1  1  1  2  0
 2  2  2  2  1  2  2  0  0  1  1  1  1     2  1  1  2  2  2  0  0  1  2  2  1
 2  2  2  2  0  2  1  0  0  2  1  0  2     2  1  0  2  2  0  1  0  2  2  2  0
 2  2  1  1  1  2  2  0  0  0  1  0  2     2  2  0  2  2  1  1  1  1  2  2  0
 2  2  2  2  0  1  1  0  0  2  2  0  2  …  2  1  0  2  2  2  1  1  1  1  1  1
 2  2  2  2  1  2  0  0  1  2  2  0  2     1  2  0  2  2  1  1  1  1  2  2  1
 2  2  2  2  0  1  1  0  0  2  2  0  2     2  2  0  2  2  0  2  0  2  2  2  1
 2  1  1  1  1  2  0  0  2  2  0  1  1     2  1  1  2  2  1  1  2  0  0  2  2
 2  2  2  2  1  2  0  0  0  2  2  0  2     2  2  0  2  1  1  1  0  2  2  2  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]: 0.6309193560372355

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

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

Out[100]:
```
0.7911635202763999
```

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

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

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

Out[102]:
```
50-element Array{Float64,1}:
 0.203615
 0.201853
 0.201394
 0.199295
 0.202489
 0.200493
 0.201478
 0.20051
 0.197517
 0.194953
 0.19858
 0.20181
 0.196891
 ⋮
 0.199483
 0.201302
 0.198966
 0.202607
 0.197116
 0.199659
 0.200694
 0.202972
 0.197984
 0.200308
 0.202927
 0.198344
```

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

Out[103]:
```
48000-element Array{Float64,1}:
  9.99183
  7.78454
 10.9964
 10.0016
  9.38369
  9.79509
 11.0056
  8.19445
  8.80487
  9.99076
  8.79529
  9.00639
 10.2162
  ⋮
 13.3786
 12.9915
 12.9904
 12.3949
 12.7994
 12.7892
 12.2095
 12.7793
 13.5951
 12.9836
 12.7877
 13.1979
```

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

Out[104]:
```
9.854777916223295
```

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

Out[105]:
```
10.331195990596312
```

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

Out[106]:
```
10.814103489249966
```

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

Out[107]:
```
11.303601939705059
```

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

Out[108]:
```
11.758701859305718
```

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

Out[109]:
```
12.194188858376759
```

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

Out[110]:
```
9000-element Array{Float64,1}:
 11.0056
 10.9957
 10.581
 10.8104
 10.8103
 11.2008
 11.6103
 10.3914
 11.8166
 10.7929
 10.7918
 10.3885
 10.1933
  ⋮
 13.3786
 12.9915
 12.9904
 12.3949
 12.7994
 12.7892
 12.2095
 12.7793
 13.5951
 12.9836
 12.7877
 13.1979
```

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

Out[111]: 12.145734145478155

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

Out[112]: 2.29095622925486

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

Out[113]: 10.8251326329286

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

Out[114]: 0.9703547167053053

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

Out[115]: 11.333643036985027

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

Out[116]: 1.478865120761732

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

Out[117]: 11.805152263299313

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

Out[118]: 1.9503743470760178

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

Out[119]: 12.206754654978127

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

Out[120]: 2.351976738754832

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

Out[121]: 12.61979962325567

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

Out[122]: 2.765021707032375

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

Out[123]: 12.194188858376759

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

Out[124]: 2.3394109421534637