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)
```

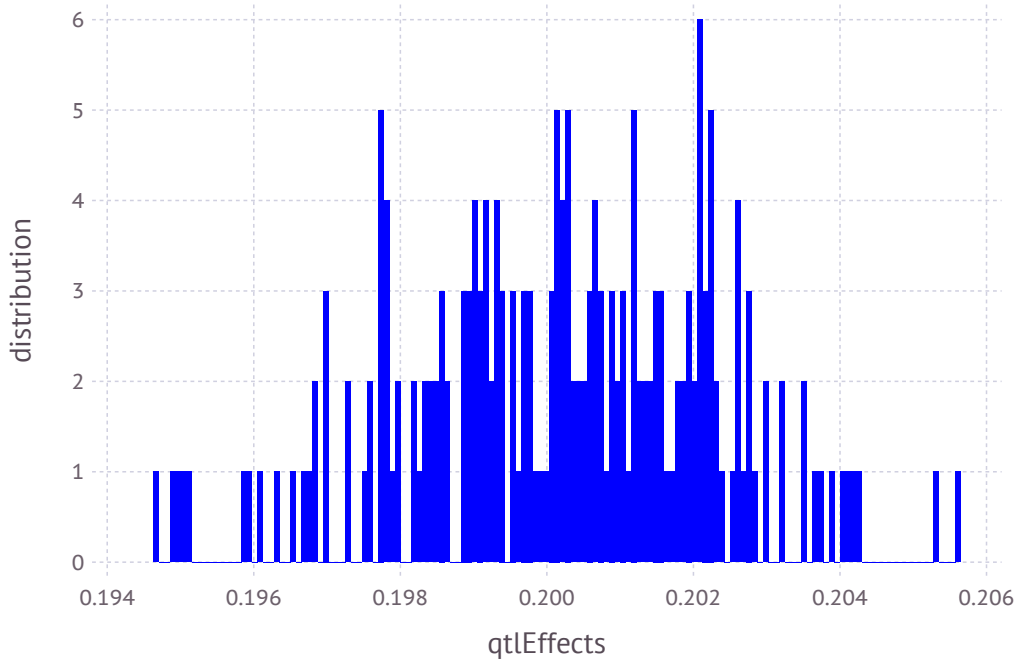
In [7]:
```
qtlEffects
```

Out[7]:
```
200-element Array{Float64,1}:
 0.197014
 0.197794
 0.197734
 0.200911
 0.198836
 0.200421
 0.198577
 0.200061
 0.201203
 0.197948
 0.197965
 0.204004
 0.200688
 ⋮
 0.200507
 0.199841
 0.199322
 0.200149
 0.202086
 0.197828
 0.20197
 0.196777
 0.202989
 0.199098
 0.202228
 0.198262
```

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]: 0.20017051652858472

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

Out[11]: 4.4568869131802544e-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
```

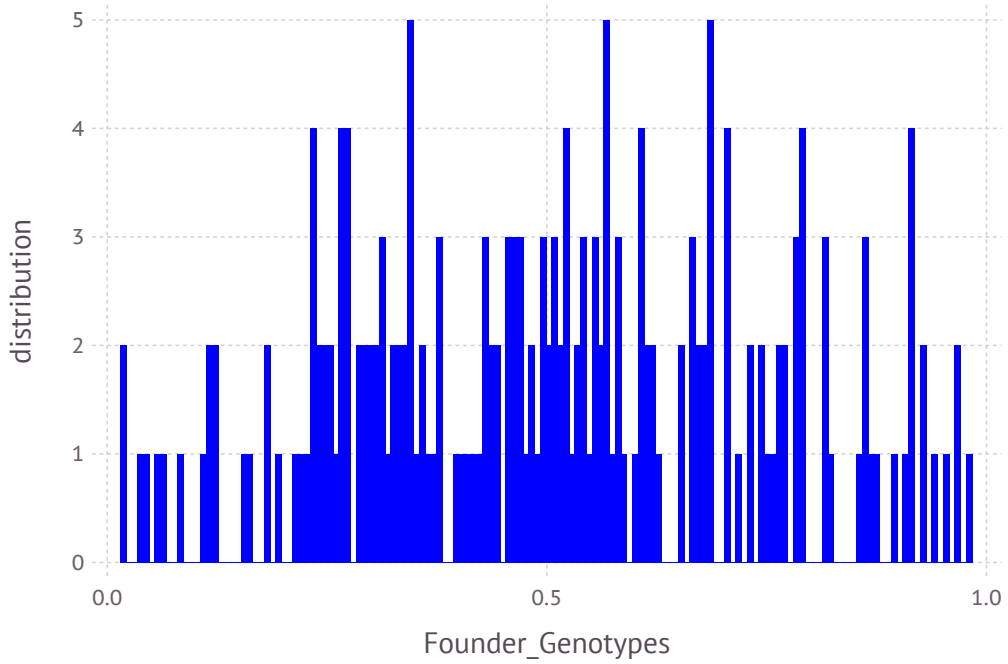
In [19]:
```
gSPSire = XSim.getOurGenotypes(popSP[1])
gSPDam = XSim.getOurGenotypes(popSP[2])
gSP = [gSPSire;gSPSire];
```

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

Out[20]:
```
1x200 Array{Float64,2}:
 0.060125  0.853125  0.2735  0.951375  0.82625  …  0.37075  0.912625  0.54275
```

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);
```

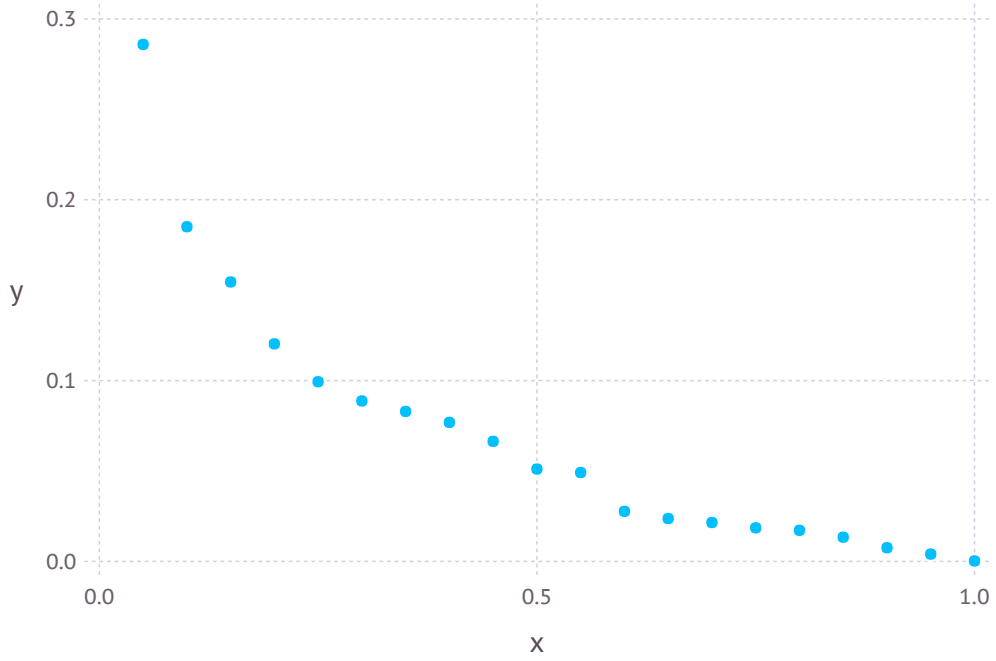
In [23]:
```
for i=1:(nRows-20)
LDMat[i,:] = corMat[i,(i+1):(i+20)].^2
end
```

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

Out[24]:
```
1x20 Array{Float64,2}:
 0.0002643  0.00408809  0.00757364  …  0.154556  0.185093  0.285889
```

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]:
```
9.342731491574312
```

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

Out[31]:
```
0.7141536641192705
```

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

Out[32]:
```
0.7141536641192705
```

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

Out[33]:
```
0.7141536641192705
```

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]: 11.931993180510451

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

Out[36]: 11.942388234848133

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

Out[37]: 0.4716679712374789

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

Out[38]: 0.481868517975879

Pedigree: All animals

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

Out[39]:
```
48000x3 Array{Int64,2}:
 40722  36032  39839
 40723  34230  38195
 40724  36143  39308
 40725  35791  37823
 40726  35212  37419
 40727  33545  36835
 40728  35391  36765
 40729  33201  40635
 40730  34712  40310
 40731  35549  37915
 40732  34408  37777
 40733  34250  37637
 40734  36149  40412
     ⋮
 88710  74777  79143
 88711  76693  79071
 88712  76657  79923
 88713  74817  79785
 88714  74020  77957
 88715  75328  77145
 88716  74492  80613
 88717  76030  80603
 88718  74140  78444
 88719  76573  78594
 88720  74154  78255
 88721  74688  77736
```

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

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

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}:
 43845
 40815
 43096
 40811
 40805
 44222
 40748
 41253
 42175
 44274
 41327
 43494
 42598
     ⋮
 74777
 76693
 76657
 74817
 74020
 75328
 74492
 76030
 74140
 76573
 74154
 74688
```

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

Out[53]:
```
1000-element Array{Int64,1}:
 43845
 40815
 43096
 40811
 40805
 44222
 40748
 41253
 42175
 44274
 41327
 43494
 42598
     ⋮
 75496
 72752
 74751
 74720
 74810
 74467
 76368
 73336
 74902
 76433
 75808
 73653
```

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}:
 43845
 40815
 43096
 40811
 40805
 44222
 40748
 41253
 42175
 44274
 41327
 43494
 42598
     ⋮
 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  10.135   8.56
 40723  10.483   9.959
 40724  10.433   9.36
 40725   9.069   9.173
 40726   7.739   8.965
 40727   9.598   9.178
 40728  10.408   9.972
 40729   9.651   8.96
 40730   7.0     8.759
 40731  10.043   8.775
 40732  10.523  11.368
 40733   8.925   9.563
 40734   6.668   8.766
    ⋮
 88710  10.912  12.353
 88711  12.395  12.756
 88712  12.938  12.76
 88713  12.758  13.151
 88714  12.793  12.952
 88715  13.526  12.159
 88716  13.431  13.557
 88717  11.908  12.758
 88718  13.239  12.754
 88719  14.253  14.146
 88720  12.327  12.153
 88721  13.362  12.161
```

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}:
   3
   5
   6
   9
  10
  23
  25
  26
  29
  30
  43
  45
  46
   ⋮
 149
 150
 163
 165
 166
 169
 170
 183
 185
 186
 189
 190
```

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

Out[83]:
```
150-element Array{Int64,1}:
   1
   2
   4
   7
   8
  11
  12
  13
  14
  15
  16
  17
  18
   ⋮
 187
 188
 191
 192
 193
 194
 195
 196
 197
 198
 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	0	2	2	1	1	1	1	1	1	1	2	…	2	2	1	1	2	2	0	1	1	0	2	1
0	1	1	2	2	1	1	1	1	1	1	1	1		2	1	1	1	1	2	0	1	1	0	2	1
0	2	1	2	1	0	0	0	2	0	0	1	0		2	2	2	1	2	2	0	1	1	1	2	1
0	2	1	2	1	0	0	0	2	0	0	2	1		2	1	0	2	0	1	0	0	0	0	1	2
0	2	2	2	0	0	0	0	2	0	0	2	0		1	1	2	1	1	1	1	1	1	1	2	1
0	2	0	2	2	1	2	2	1	1	1	1	2	…	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	1	2	2	1	1	1	0	0	1	2	1
0	1	1	2	2	0	1	1	1	1	1	0	0		2	1	1	2	1	2	0	0	0	1	2	2
0	2	0	2	2	0	0	0	2	0	0	0	1		2	1	1	1	1	2	0	1	1	0	2	1
0	2	0	2	2	1	1	1	1	1	1	0	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
0	1	1	2	1	1	1	1	1	1	1	0	0		2	0	1	2	0	1	0	0	0	1	1	2
0	0	2	1	1	0	2	2	0	2	2	0	1		2	1	1	1	2	2	0	2	2	0	2	0
⋮				⋮					⋮				⋱			⋮					⋮				
0	0	2	2	2	2	2	2	0	2	2	0	1		2	2	1	1	1	2	1	1	1	1	1	1
0	1	1	2	2	1	1	1	1	1	1	0	1		2	2	2	0	2	2	0	2	2	0	2	0
0	0	2	1	1	0	1	1	1	1	1	1	2	…	2	2	2	1	2	2	0	1	1	1	2	1
0	2	0	2	2	0	0	0	2	0	0	0	0		2	2	1	1	2	2	0	1	1	0	2	1
0	1	1	2	2	1	1	1	1	1	1	1	2		2	2	2	0	2	2	0	2	2	0	2	0
0	1	1	2	2	2	2	2	0	2	2	0	2		2	2	2	0	2	2	0	2	2	0	2	0
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
0	1	1	2	2	0	1	1	1	1	1	0	0	…	2	2	2	0	2	2	0	2	2	0	2	0
0	0	2	2	2	1	1	1	1	1	1	1	2		2	2	2	1	2	2	0	1	1	1	2	1
0	0	2	2	2	2	2	2	0	2	2	0	2		2	2	2	1	2	2	0	1	1	1	2	1
0	0	2	2	2	1	2	2	0	2	2	0	2		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	2	2	2	2	0	0	0	2	1	2

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

Out[86]:
```
9000x50 Array{Int64,2}:
 0  2  1  1  1  1  2  0  2  0  0  2  2  …  1  1  1  1  1  1  2  2  1  1  2  2
 1  2  1  1  1  2  0  0  2  0  0  2  1     1  1  0  0  1  2  2  2  2  0  2  1
 1  1  0  2  0  0  2  0  2  0  0  0  2     1  1  0  0  1  1  2  2  1  2  2  2
 1  1  0  2  0  2  1  1  1  0  0  0  2     1  1  1  1  1  0  2  2  2  1  2  1
 2  0  0  2  0  1  2  0  2  0  0  1  1     0  2  2  1  1  1  2  1  1  1  1  1
 0  2  1  1  1  0  2  1  2  0  1  0  1  …  1  1  1  1  1  1  2  1  1  1  1  1
 1  1  1  1  1  2  1  1  2  0  0  0  2     2  1  0  1  1  1  1  1  2  1  1  1
 1  2  0  1  1  2  0  0  1  0  0  1  1     1  1  0  0  2  0  2  2  2  1  2  1
 0  2  0  2  0  2  0  0  0  0  1  0  2     0  2  0  0  1  1  2  2  1  1  2  1
 0  2  1  1  1  1  2  1  2  0  1  1  1     2  0  0  0  1  1  1  1  1  1  1  1
 0  2  0  2  0  2  0  0  1  0  0  1  2  …  2  1  0  0  2  0  1  2  0  2  2  2
 1  1  1  1  1  2  1  0  1  0  0  2  2     2  0  0  0  1  1  2  2  2  1  2  0
 2  1  0  0  2  1  1  1  1  0  1  1  1     0  2  0  0  1  2  2  2  1  1  2  1
 ⋮                 ⋮                 ⋮        ⋱        ⋮                 ⋮
 2  2  2  0  2  2  2  0  2  0  2  0  0     2  2  1  1  1  2  2  2  1  2  2  2
 1  2  1  1  1  2  2  2  2  0  0  2  2     0  2  0  0  2  0  2  2  0  2  2  2
 2  1  0  1  1  2  1  1  2  0  0  2  2  …  2  1  0  0  1  2  2  2  1  2  2  2
 0  2  0  2  0  1  2  1  2  0  0  1  2     1  2  1  2  0  2  1  2  1  1  2  2
 1  2  1  1  1  1  1  0  2  0  1  1  2     2  2  1  0  2  0  2  2  0  2  2  2
 1  2  2  0  2  2  1  0  2  0  0  2  2     2  1  1  0  1  1  1  2  1  1  2  2
 2  1  1  1  1  2  2  2  2  0  0  0  2     2  2  0  0  2  1  1  2  0  2  2  2
 1  2  0  1  1  1  2  1  2  0  0  1  1  …  1  2  0  0  1  1  1  2  0  2  2  2
 2  2  1  1  1  2  2  0  2  0  0  1  2     1  2  0  0  2  0  2  2  1  2  2  2
 2  2  2  0  2  2  2  2  2  0  0  2  2     1  2  1  1  1  0  2  2  1  2  2  2
 2  2  1  0  2  2  2  1  2  0  1  0  2     2  0  0  0  2  0  2  1  1  1  1  1
 0  2  0  2  0  1  2  1  2  0  1  1  2     2  1  0  1  1  1  1  2  1  2  2  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]: 0.6401306518501786

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

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

Out[100]:
```
0.7991413710180063
```

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

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

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

Out[102]:
```
50-element Array{Float64,1}:
 0.197734
 0.198836
 0.200421
 0.201203
 0.197948
 0.196835
 0.197325
 0.199046
 0.201176
 0.200382
 0.199418
 0.202619
 0.199292
 ⋮
 0.199543
 0.19978
 0.195125
 0.201082
 0.201518
 0.200307
 0.200322
 0.195884
 0.201889
 0.198998
 0.200507
 0.199841
```

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

Out[103]:
```
48000-element Array{Float64,1}:
  8.58113
  9.97021
  9.37979
  9.19273
  8.98211
  9.1892
  9.99077
  8.97195
  8.77234
  8.79382
 11.3823
  9.56385
  8.77911
  ⋮
 12.3772
 12.7951
 12.7802
 13.1917
 12.9869
 12.1786
 13.5907
 12.8013
 12.7792
 14.1841
 12.1821
 12.1959
```

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

Out[104]:
```
9.361243555158502
```

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

Out[105]:
```
9.926092870769022
```

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

Out[106]:
```
10.482917747505285
```

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

Out[107]:
```
10.990079760365203
```

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

Out[108]:
```
11.499969752701984
```

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

Out[109]:
```
11.962620886192642
```

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

Out[110]:
```
9000-element Array{Float64,1}:
 11.5843
 10.5924
 10.1829
 10.3712
 10.1778
 10.589
 11.3838
 10.3879
 10.7886
  9.98042
 10.3757
 10.1759
  9.78035
  ⋮
 12.3772
 12.7951
 12.7802
 13.1917
 12.9869
 12.1786
 13.5907
 12.8013
 12.7792
 14.1841
 12.1821
 12.1959
```

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

Out[111]: 11.910910721236837

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

Out[112]: 2.5496671660783345

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

Out[113]: 10.460665241587613

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

Out[114]: 1.0994216864291104

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

Out[115]: 11.045829316385328

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

Out[116]: 1.6845857612268258

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

Out[117]: 11.528243278796516

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

Out[118]: 2.166999723638014

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

Out[119]: 12.030290944237912

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

Out[120]: 2.6690473890794095

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

Out[121]: 12.421118226944497

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

Out[122]: 3.0598746717859946

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

Out[123]: 11.962620886192642

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

Out[124]: 2.60137733103414