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 = 0.0
# 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
Va = nQTL*numChr*0.5              # Va= nQTL*2pq*mean(alpha)^2; mean(alpha) = 0
qtlEffects= rand(Normal(0, sqrt(1/Va)),numLoci*numChr)  # Let alpha mu = 0.0
XSim.init(numChr,numLoci,chrLength,geneFreq,mapPos,qtlMarker,qtlEffects,mutRate)
```

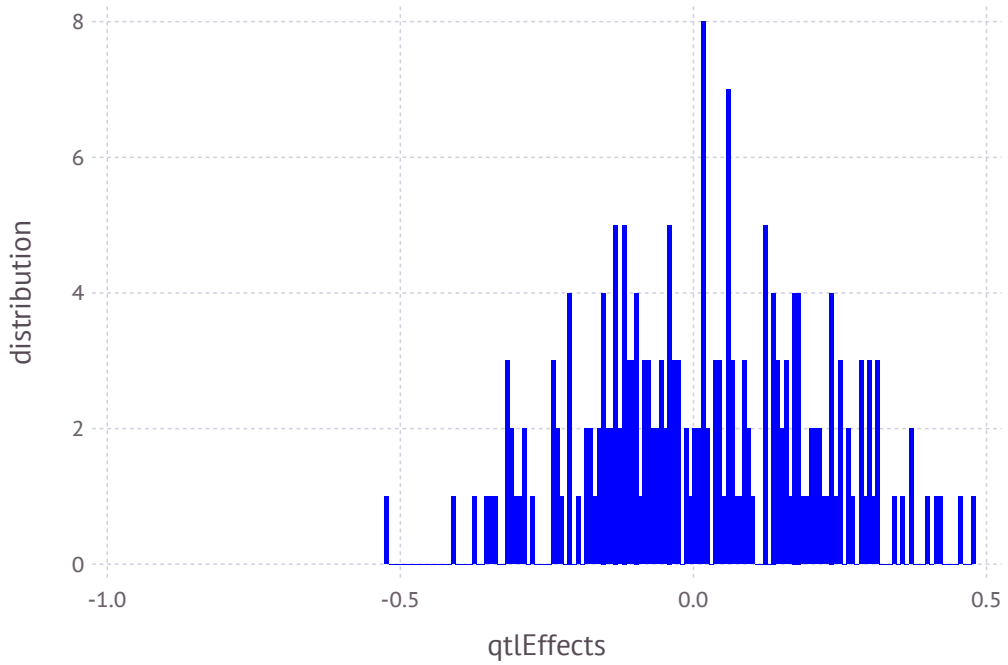
In [7]:
```
qtlEffects
```

Out[7]:
```
200-element Array{Float64,1}:
  0.287521
  0.201761
 -0.0281609
 -0.159658
  0.299098
 -0.0592318
 -0.111594
  0.182854
  0.0631532
  0.148249
  0.315287
  0.0660627
 -0.134084
  ⋮
  0.233941
  0.0176892
  0.370153
  0.23142
 -0.0692326
  0.136517
 -0.338492
 -0.0538571
  0.481246
  0.176676
 -0.227584
 -0.238883
```

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]: 0.01629809781818762

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

Out[11]: 0.03727546132404205

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
```

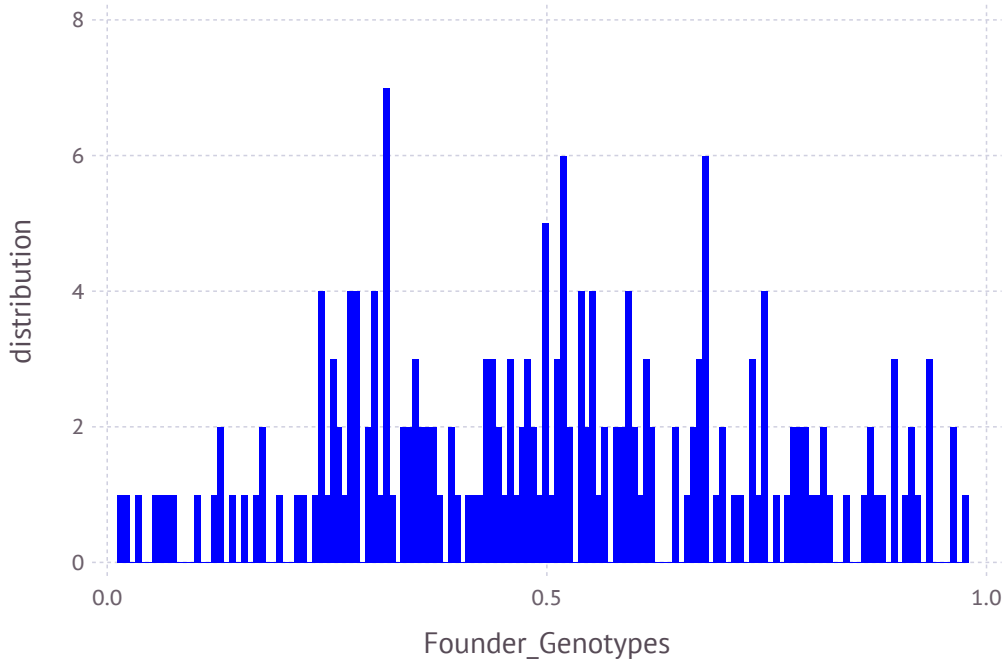
In [19]:
```
gSPSire = XSim.getOurGenotypes(popSP[1])
gSPDam = XSim.getOurGenotypes(popSP[2])
gSP = [gSPSire;gSPSire];
```

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

Out[20]:
```
1x200 Array{Float64,2}:
 0.069875  0.839375  0.28225  0.93575  …  0.369625  0.892625  0.545375
```

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);
```

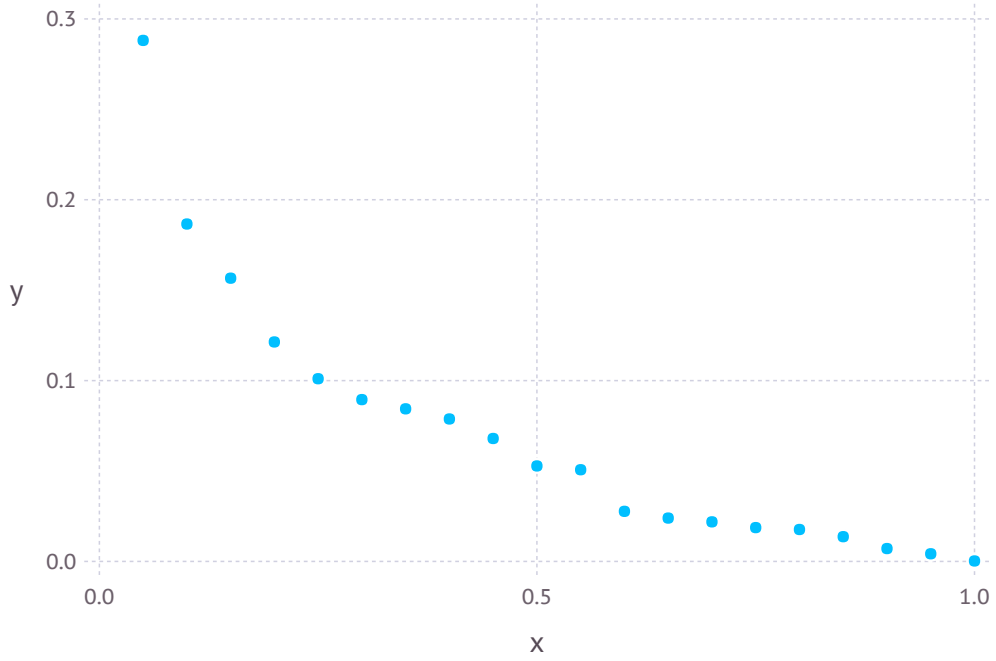
In [23]:
```
for i=1:(nRows-20)
LDMat[i,:] = corMat[i,(i+1):(i+20)].^2
end
```

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

Out[24]:
```
1x20 Array{Float64,2}:
 0.000251171  0.00422733  0.0071456  …  0.156625  0.186572  0.288111
```

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]:
```
3.855261774847346
```

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

Out[31]:
```
0.46069767412711804
```

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

Out[32]:
```
0.46069767412711804
```

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

Out[33]:
```
0.46069767412711804
```

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]: 5.91316729647545

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

Out[36]: 5.910756159405064

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

Out[37]: 0.3051799381034903

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

Out[38]: 0.3083534612294081

Pedigree: All animals

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

Out[39]:
```
48000x3 Array{Int64,2}:
 40722  33991  39218
 40723  32845  36845
 40724  35262  39152
 40725  35263  38409
 40726  33819  39962
 40727  36093  38088
 40728  33428  37854
 40729  33995  39412
 40730  33764  38418
 40731  35447  37581
 40732  34374  37751
 40733  36660  37478
 40734  35706  39709
     ⋮
 88710  74065  80014
 88711  74595  80607
 88712  74393  79615
 88713  74847  80144
 88714  74825  80236
 88715  76298  77052
 88716  73917  80684
 88717  74183  79966
 88718  75104  79637
 88719  74254  80115
 88720  75513  76826
 88721  75637  79775
```

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

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}:
 41937
 41581
 43364
 40797
 43846
 42464
 41866
 40923
 42607
 43013
 40889
 40889
 43828
    ⋮
 74065
 74595
 74393
 74847
 74825
 76298
 73917
 74183
 75104
 74254
 75513
 75637
```

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

Out[53]:
```
1000-element Array{Int64,1}:
 41937
 41581
 43364
 40797
 43846
 42464
 41866
 40923
 42607
 43013
 40889
 43828
 43738
     ⋮
 75638
 75273
 75605
 73557
 75844
 76419
 75496
 74656
 74983
 73436
 75379
 73961
```

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}:
 41937
 41581
 43364
 40797
 43846
 42464
 41866
 40923
 42607
 43013
 40889
 43828
 43738
     ⋮
 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  3.142  2.844
 40723  4.046  3.619
 40724  4.226  4.092
 40725  3.785  3.344
 40726  3.231  3.221
 40727  4.539  3.871
 40728  4.082  3.936
 40729  2.563  3.246
 40730  5.07   3.886
 40731  3.672  3.435
 40732  4.677  4.368
 40733  5.385  3.998
 40734  2.904  3.442
     ⋮
 88710  5.809  5.668
 88711  7.472  6.66
 88712  5.372  5.742
 88713  5.776  6.283
 88714  7.934  6.063
 88715  6.701  5.839
 88716  6.447  6.354
 88717  3.927  5.673
 88718  5.822  5.951
 88719  7.082  6.402
 88720  6.133  6.83
 88721  7.831  6.913
```

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}:
   6
   9
  11
  17
  19
  26
  29
  31
  37
  39
  46
  49
  51
   ⋮
 157
 159
 166
 169
 171
 177
 179
 186
 189
 191
 197
 199
```

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

Out[83]:
```
150-element Array{Int64,1}:
   1
   2
   3
   4
   5
   7
   8
  10
  12
  13
  14
  15
  16
   ⋮
 184
 185
 187
 188
 190
 192
 193
 194
 195
 196
 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	1	1	2	2	0	2	2	1	1	1	1	1	…	1	1	2	1	1	1	1	1	1	2	1
0	1	1	2	2	0	0	0	2	0	0	1	1		2	1	2	2	1	2	0	0	0	2	1	1
0	2	0	2	2	0	0	0	2	0	0	0	0		2	0	2	2	2	1	1	0	0	1	2	2
0	2	0	2	2	2	2	2	0	2	2	0	2		1	0	2	2	0	0	1	0	0	2	1	2
0	1	2	1	0	0	1	1	1	1	1	1	1		2	2	2	1	2	2	0	1	1	1	2	1
0	1	1	2	2	0	1	1	1	1	1	1	1	…	1	1	1	2	0	1	2	0	0	2	1	2
0	2	0	2	2	0	1	1	2	0	0	2	2		2	2	2	0	2	2	0	2	2	0	2	0
0	1	1	2	2	0	2	2	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		1	1	2	1	1	1	1	1	1	1	2	1
0	2	1	2	1	0	0	0	2	0	0	1	0		2	1	2	1	2	2	0	2	2	0	2	0
0	2	0	2	2	1	1	1	1	1	1	0	1	…	2	1	1	1	1	2	0	1	1	0	2	1
0	2	1	2	1	0	1	1	1	1	1	1	0		2	2	2	0	2	2	0	2	2	0	2	0
0	2	0	2	2	0	0	0	2	0	0	1	1		2	0	1	2	1	2	0	1	1	0	2	1
⋮				⋮				⋮			⋱			⋮					⋮					
0	1	1	2	2	0	2	2	0	2	2	0	2		0	0	2	2	0	0	2	0	0	2	2	2
1	2	0	2	2	1	2	2	0	2	2	0	1		2	1	2	1	2	2	0	1	1	0	2	1
1	2	0	2	1	1	2	2	0	2	2	0	1	…	1	0	2	2	1	1	1	0	0	1	2	2
0	2	0	2	2	1	1	1	1	1	1	0	1		2	0	0	2	0	2	0	0	0	0	2	2
0	2	1	2	1	0	0	0	2	0	0	2	1		2	1	1	1	1	2	1	1	1	1	1	1
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	0	1	1	1	1	1	0	1		2	0	2	2	1	1	0	0	0	0	2	2
0	1	1	2	2	0	1	1	1	1	1	1	2	…	1	0	2	2	0	0	1	0	0	1	2	2
0	1	2	1	0	0	1	1	1	1	1	1	1		2	2	1	2	1	1	0	0	0	1	1	2
0	1	1	2	2	0	1	1	1	1	1	1	2		2	1	2	2	2	2	0	0	0	2	1	2
1	2	0	2	2	0	1	1	1	1	1	0	0		1	1	2	1	1	1	1	1	1	1	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

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

Out[86]:
```
9000x50 Array{Int64,2}:
 0  1  1  1  0  0  2  0  0  1  1  1  1  …  2  2  1  0  1  1  0  1  1  2  1  2
 0  2  0  1  0  1  2  1  0  1  1  1  1     1  1  0  1  1  1  1  2  2  2  0  1
 0  2  0  1  0  0  1  0  1  1  1  2  1     2  1  2  1  2  2  1  1  2  2  0  2
 2  0  2  1  0  2  2  2  1  1  0  1  1     2  1  0  1  0  0  0  1  1  2  0  1
 0  1  1  0  0  0  2  0  0  1  0  1  1     2  2  2  1  1  2  1  1  2  2  1  2
 0  1  1  1  0  1  1  0  2  0  1  1  1  …  2  1  1  1  1  2  1  1  1  1  0  1
 0  2  0  0  0  1  2  1  1  0  0  1  0     2  1  2  0  1  1  0  2  2  2  2  2
 0  1  1  1  0  0  1  0  1  0  1  2  2     2  2  1  1  0  2  1  2  2  2  2  2
 0  1  1  2  0  0  2  0  0  1  0  0  0     2  2  1  0  0  1  0  1  1  2  1  2
 0  2  0  1  0  0  2  1  1  1  0  2  1     2  1  1  1  1  2  1  2  2  2  2  2
 1  1  1  2  0  0  2  0  0  1  1  2  1  …  2  2  2  0  1  1  0  1  2  1  1  2
 0  1  1  1  0  0  2  0  2  1  1  1  1     2  2  2  0  2  0  0  2  2  2  2  2
 0  2  0  1  0  1  2  1  1  1  0  2  2     2  2  0  2  2  2  1  1  2  1  1  2
 ⋮              ⋮              ⋮           ⋱        ⋮              ⋮
 0  0  2  1  0  0  2  0  1  1  0  1  1     2  2  2  1  2  2  1  0  0  2  0  2
 1  0  2  1  1  1  2  1  1  1  2  2  2     2  2  1  1  1  2  2  1  2  2  1  2
 1  0  2  2  1  0  1  1  1  0  2  2  2  …  2  2  1  0  1  1  0  0  1  2  0  2
 1  1  1  1  0  1  2  1  0  2  0  2  2     2  2  1  2  1  1  1  0  2  0  0  2
 0  2  0  0  0  1  2  1  0  2  1  1  2     2  2  1  1  2  1  1  1  2  1  1  1
 0  2  0  0  0  0  1  0  1  0  2  2  2     2  2  1  1  2  2  1  1  1  2  1  2
 0  1  1  2  0  0  1  0  0  1  1  2  2     2  2  1  1  2  2  0  1  2  2  0  2
 0  1  1  1  1  1  2  1  0  1  1  1  2  …  1  1  2  0  1  1  1  0  1  2  0  2
 0  1  1  0  0  2  2  2  0  2  1  2  2     2  2  1  1  0  1  1  2  2  1  0  1
 0  1  1  1  0  0  2  0  2  1  1  1  2     2  2  1  0  2  2  0  2  2  2  0  1
 0  1  1  2  1  0  2  1  1  1  2  2  2     2  2  2  0  2  2  0  1  1  2  1  2
 1  0  2  2  1  1  2  0  2  1  2  2  2     2  2  1  1  2  2  1  2  2  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]: 0.6103182019893327

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

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

Out[100]:
```
0.7830100932508139
```

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

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

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

Out[102]:
```
50-element Array{Float64,1}:
 -0.0592318
  0.0631532
  0.315287
  0.0563676
  0.0187642
 -0.211274
  0.369278
  0.0223135
 -0.105935
 -0.171108
  0.00623903
  0.251304
  0.266821
  ⋮
  0.123606
 -0.311884
  0.0935625
 -0.115136
 -0.134758
 -0.080106
  0.0445622
 -0.300549
  0.233941
  0.370153
  0.481246
 -0.227584
```

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

Out[103]:
```
48000-element Array{Float64,1}:
 1.98651
 2.72539
 2.19725
 1.85986
 0.629412
 1.95454
 2.61625
 2.79664
 1.84616
 1.52849
 3.20419
 2.72216
 2.53494
 ⋮
 1.40592
 4.13338
 2.20194
 3.56055
 3.04714
 3.19468
 3.62425
 3.19721
 2.63294
 2.64364
 3.58085
 4.17978
```

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

Out[104]:
```
2.1226945849265495
```

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

Out[105]:
```
2.3657149822071277
```

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

Out[106]:
```
2.6437108208059827
```

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

```
Out[107]: 2.876205296605837
```

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

```
Out[108]: 3.1372280241212276
```

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

```
Out[109]: 3.4107426676725603
```

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

```
Out[110]: 9000-element Array{Float64,1}:
           2.60897
           1.45204
           1.67028
           1.80214
           3.33997
           1.89164
           3.72795
           3.14402
           1.14221
           4.37492
           4.24238
           2.56981
           3.13766
           ⋮
           1.40592
           4.13338
           2.20194
           3.56055
           3.04714
           3.19468
           3.62425
           3.19721
           2.63294
           2.64364
           3.58085
           4.17978
```

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

Out[111]: 3.3817831739703776

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

Out[112]: 1.2590885890438281

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

Out[113]: 2.628089593463476

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

Out[114]: 0.5053950085369263

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

Out[115]: 2.939573770994796

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

Out[116]: 0.8168791860682467

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

Out[117]: 3.1032124414611792

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

Out[118]: 0.9805178565346298

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

Out[119]: 3.3817959139209197

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

Out[120]: 1.2591013289943702

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

Out[121]: 3.6978644019242313

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

Out[122]: 1.5751698169976818

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

Out[123]: 3.4107426676725603

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

Out[124]: 1.2880480827460108