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)
```

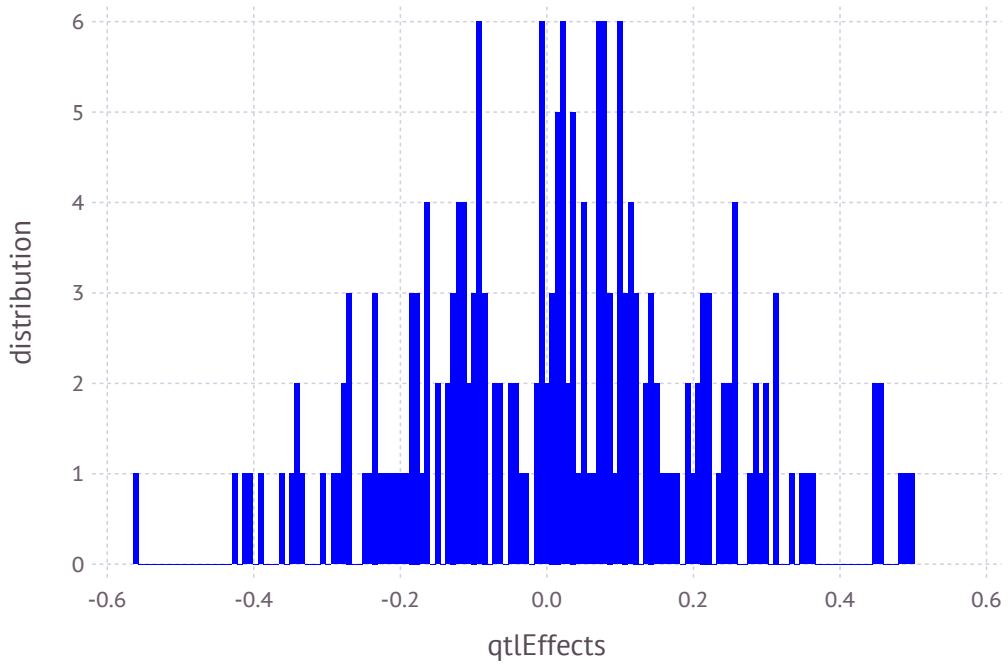
In [7]:
```
qtlEffects
```

Out[7]:
```
200-element Array{Float64,1}:
 -0.123544
 -0.148267
  0.165508
 -0.231476
  0.0270459
 -0.0839026
  0.0191086
 -0.114896
  0.146253
 -0.0154805
 -0.233182
 -0.275321
 -0.164591
  ⋮
  0.258544
 -0.219256
  0.196034
  0.203201
 -0.0816451
 -0.033189
 -0.00399408
  0.252406
  0.116275
 -0.0916927
 -0.26713
 -0.0092383
```

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]: 0.01544861095337777

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

Out[11]: 0.03963004744888262

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
```

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
```

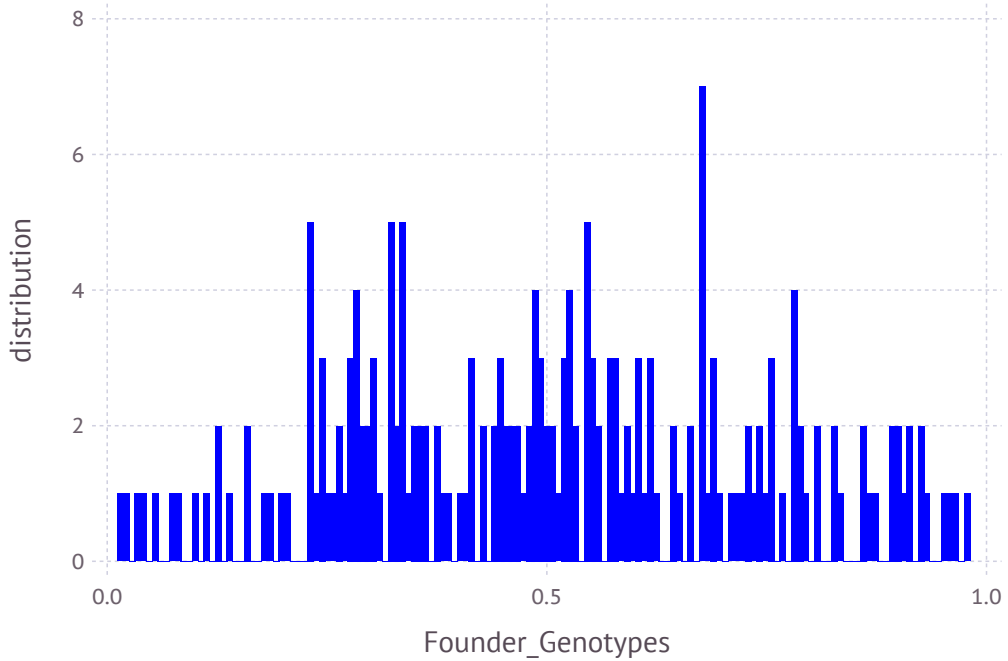
In [19]:
```
gSPSire = XSim.getOurGenotypes(popSP[1])
gSPDam = XSim.getOurGenotypes(popSP[2])
gSP = [gSPSire;gSPSire];
```

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

Out[20]:
```
1x200 Array{Float64,2}:
 0.054625  0.8325  0.285125  0.95075  …  0.364  0.384375  0.90125  0.557375
```

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);
```

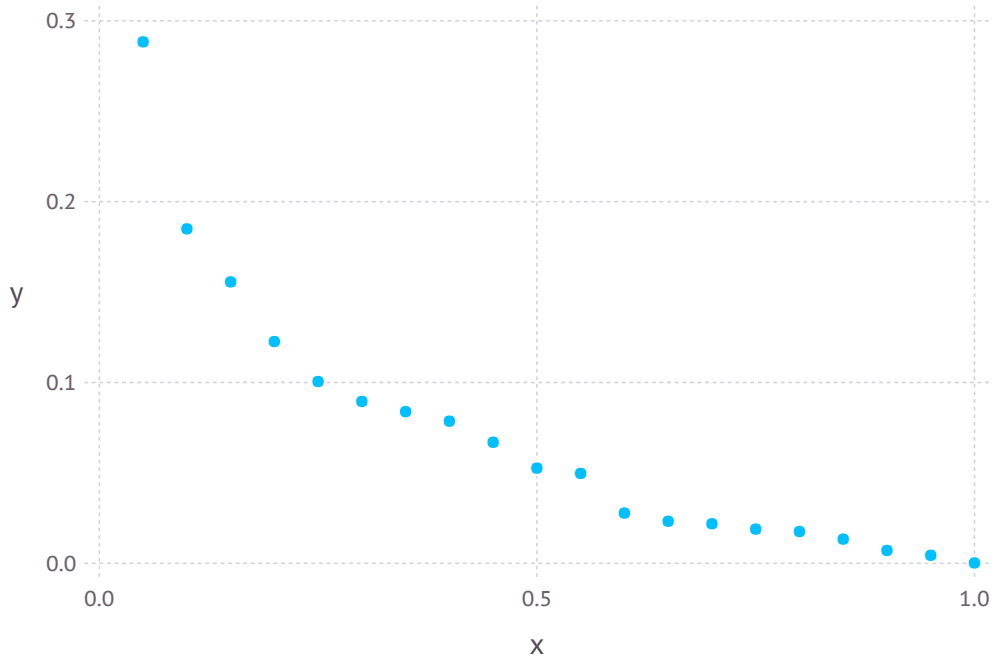
In [23]:
```
for i=1:(nRows-20)
LDMat[i,:] = corMat[i,(i+1):(i+20)].^2
end
```

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

Out[24]:
```
1x20 Array{Float64,2}:
 0.000210591  0.00445664  0.00713422  …  0.155634  0.184943  0.28837
```

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]:
```
0.013460613011598423
```

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

Out[31]:
```
0.32886771603235854
```

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

Out[32]:
```
0.32886771603235854
```

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

Out[33]:
```
0.32886771603235854
```

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]:
```
1.9297707741710366
```

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

Out[36]:
```
1.9310419859029515
```

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

Out[37]: 0.32037144729201394

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

Out[38]: 0.31386044203656377

Pedigree: All animals

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

Out[39]:
```
48000x3 Array{Int64,2}:
 40722  35167  37687
 40723  32951  39985
 40724  35585  40512
 40725  34261  36846
 40726  33785  39558
 40727  33043  38574
 40728  34039  37561
 40729  34263  38937
 40730  36320  38920
 40731  35700  36737
 40732  35227  39633
 40733  34332  38867
 40734  33835  36729
     ⋮
 88710  76380  80522
 88711  75868  79167
 88712  74773  77270
 88713  74089  80615
 88714  72936  80216
 88715  76016  80662
 88716  74220  80380
 88717  74773  80071
 88718  75642  80148
 88719  75575  80218
 88720  74063  79553
 88721  75760  80272
```

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	1	1	…	0	0	2	2	0	0	2	0	0	2	2	2
40723	0	1	1	2	2	0	1	1	1	1		2	1	1	1	2	2	0	1	1	1	2	0
40724	0	2	0	2	2	0	0	0	2	0		2	1	1	1	1	2	0	1	1	0	2	1
40725	0	2	0	2	2	0	0	0	2	0		1	1	2	1	1	1	1	1	1	1	2	1
40726	0	2	0	2	2	0	1	1	1	1		1	1	2	1	1	1	1	1	1	1	2	1
40727	0	2	0	2	2	1	1	1	1	1	…	2	2	1	1	2	2	0	1	1	0	2	1
40728	1	2	0	2	2	1	1	1	2	0		1	1	2	1	1	1	1	1	1	1	2	1
40729	2	2	0	2	2	0	1	1	1	1		1	0	2	2	1	0	2	0	0	2	2	2
40730	0	2	1	2	1	0	0	0	2	0		2	2	2	1	2	2	0	1	1	1	2	1
40731	0	1	1	2	2	1	2	2	0	2		2	1	1	2	1	2	1	1	1	0	1	2
40732	0	1	2	2	1	0	1	1	1	1	…	2	1	2	0	1	2	0	1	1	1	1	1
40733	0	2	2	2	0	0	0	0	2	0		2	1	2	1	1	1	0	1	1	0	2	1
40734	0	0	2	1	1	0	1	1	1	1		2	1	2	2	1	2	0	0	0	1	1	1
⋮				⋮					⋮	⋱		⋮				⋮						⋮	
88710	0	2	0	2	2	0	0	0	2	0		1	0	1	2	0	1	0	0	0	1	1	2
88711	0	2	2	2	1	0	0	0	2	0		2	1	1	2	1	2	0	0	0	1	2	2
88712	0	1	1	2	2	1	1	1	1	1	…	2	1	0	2	0	1	1	0	0	1	2	2
88713	0	2	1	2	1	0	0	0	2	0		2	1	0	2	1	1	1	0	0	1	2	2
88714	0	1	2	2	1	0	0	0	2	0		1	0	1	2	0	1	1	0	0	1	2	2
88715	0	2	0	2	2	0	0	0	2	0		2	0	0	2	1	1	1	0	0	0	2	2
88716	0	2	0	2	2	0	0	0	2	0		2	0	1	1	1	2	0	1	1	0	2	1
88717	0	1	1	2	2	1	1	1	1	1	…	2	1	0	2	0	1	1	0	0	1	2	2
88718	0	0	2	2	2	0	0	0	2	0		2	2	1	2	2	2	0	0	0	1	2	2
88719	0	1	1	2	2	1	0	0	2	0		2	1	1	2	1	1	0	0	0	0	2	2
88720	0	1	2	2	1	0	0	0	2	0		2	1	1	1	1	2	0	1	1	0	2	1
88721	0	2	0	2	2	0	0	0	2	0		2	1	0	2	1	2	0	0	0	0	2	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  0  2  2  0  1  1  1  1  1  0  1  …  0  0  2  2  0  0  2  0  0  2  2  2
 0  1  1  2  2  0  1  1  1  1  1  1  2     2  1  1  1  2  2  0  1  1  1  2  0
 0  2  0  2  2  0  0  0  2  0  0  0  0     2  1  1  1  1  2  0  1  1  0  2  1
 0  2  0  2  2  0  0  0  2  0  0  2  2     1  1  2  1  1  1  1  1  1  1  2  1
 0  2  0  2  2  0  1  1  1  1  1  1  2     1  1  2  1  1  1  1  1  1  1  2  1
 0  2  0  2  2  1  1  1  1  1  1  0  1  …  2  2  1  1  2  2  0  1  1  0  2  1
 1  2  0  2  2  1  1  1  2  0  0  1  0     1  1  2  1  1  1  1  1  1  1  2  1
 2  2  0  2  2  0  1  1  1  1  1  1  0     1  0  2  2  1  0  2  0  0  2  2  2
 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  1  1  2  2  1  2  2  0  2  2  0  1     2  1  1  2  1  2  1  1  1  0  1  2
 0  1  2  2  1  0  1  1  1  1  1  1  1  …  2  1  2  0  1  2  0  1  1  1  1  1
 0  2  2  2  0  0  0  0  2  0  0  2  0     2  1  2  1  1  1  0  1  1  0  2  1
 0  0  2  1  1  0  1  1  1  1  1  1  2     2  1  2  2  1  2  0  0  0  1  1  1
 ⋮              ⋮              ⋮        ⋱        ⋮              ⋮
 0  2  0  2  2  0  0  0  2  0  0  1  1     1  0  1  2  0  1  0  0  0  1  1  2
 0  2  2  2  1  0  0  0  2  0  0  1  0     2  1  1  2  1  2  0  0  0  1  2  2
 0  1  1  2  2  1  1  1  1  1  1  1  2  …  2  1  0  2  0  1  1  0  0  1  2  2
 0  2  1  2  1  0  0  0  2  0  0  1  0     2  1  0  2  1  1  1  0  0  1  2  2
 0  1  2  2  1  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  0  0  2  1  1  1  0  0  0  2  2
 0  2  0  2  2  0  0  0  2  0  0  1  1     2  0  1  1  1  2  0  1  1  0  2  1
 0  1  1  2  2  1  1  1  1  1  1  0  1  …  2  1  0  2  0  1  1  0  0  1  2  2
 0  0  2  2  2  0  0  0  2  0  0  2  2     2  2  1  2  2  2  0  0  0  1  2  2
 0  1  1  2  2  1  0  0  2  0  0  2  2     2  1  1  2  1  1  0  0  0  0  2  2
 0  1  2  2  1  0  0  0  2  0  0  2  1     2  1  1  1  1  2  0  1  1  0  2  1
 0  2  0  2  2  0  0  0  2  0  0  0  0     2  1  0  2  1  2  0  0  0  0  2  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}:
 41951
 42506
 43121
 41994
 42190
 43773
 41683
 44435
 44185
 41951
 41144
 43446
 40898
     ⋮
 76380
 75868
 74773
 74089
 72936
 76016
 74220
 74773
 75642
 75575
 74063
 75760
```

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

Out[53]:
```
1000-element Array{Int64,1}:
 41951
 42506
 43121
 41994
 42190
 43773
 41683
 44435
 44185
 41144
 43446
 40898
 44287
    ⋮
 74085
 75842
 75121
 76016
 74478
 76469
 73925
 76455
 75828
 75724
 75637
 76612
```

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}:
 41951
 42506
 43121
 41994
 42190
 43773
 41683
 44435
 44185
 41144
 43446
 40898
 44287
     ⋮
 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  -1.268  -0.396
 40723  -1.034   0.019
 40724  -0.966   1.022
 40725  -1.408  -0.649
 40726   0.249  -0.723
 40727   0.215   0.648
 40728   0.957   0.174
 40729  -0.363   0.437
 40730   0.831   0.522
 40731  -1.825  -0.167
 40732   0.959  -0.097
 40733  -1.123  -0.271
 40734  -0.104  -0.382
     ⋮
 88710   2.365   1.662
 88711   3.896   3.461
 88712   2.288   2.746
 88713   3.097   2.972
 88714   1.955   2.386
 88715   2.169   2.168
 88716   2.296   2.329
 88717   1.75    2.221
 88718   2.803   2.748
 88719   1.969   2.495
 88720   3.182   3.227
 88721   1.765   2.742
```

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
   8
  11
  18
  19
  23
  28
  31
  38
  39
  43
  48
  51
   ⋮
 158
 159
 163
 168
 171
 178
 179
 183
 188
 191
 198
 199
```

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

Out[83]:
```
150-element Array{Int64,1}:
   1
   2
   4
   5
   6
   7
   9
  10
  12
  13
  14
  15
  16
   ⋮
 185
 186
 187
 189
 190
 192
 193
 194
 195
 196
 197
 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	2	2	0	0	0	0	2	0	0	2	0	…	1	0	1	2	1	0	1	0	0	1	2	2
0	2	1	2	1	0	0	0	2	0	0	1	0		2	1	1	1	1	1	1	1	1	1	2	1
0	1	1	2	2	1	1	1	1	1	1	1	2		2	1	0	2	1	2	0	0	0	0	2	2
0	2	0	2	2	0	0	0	2	0	0	0	0		2	2	2	0	2	2	0	1	1	1	2	1
0	2	0	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	…	1	0	1	2	0	1	1	0	0	1	2	2
0	0	2	2	2	1	2	2	0	2	2	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	2	2		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	0	0	2	1	2	0	0	0	1	2	1
0	2	0	2	2	0	0	0	2	0	0	2	2		1	1	2	1	1	1	1	1	1	1	2	1
0	2	1	2	1	0	1	1	2	0	0	2	1	…	2	0	0	2	0	2	0	0	0	0	2	2
0	2	0	2	2	0	0	0	2	0	0	2	2		2	2	2	0	1	1	1	1	1	1	2	1
1	1	1	2	2	0	1	1	1	1	1	1	0		2	1	1	2	1	2	0	0	0	1	2	2
⋮				⋮					⋮				⋱			⋮				⋮				
0	2	0	2	2	0	0	0	2	0	0	1	1		1	0	1	2	0	1	0	0	0	1	1	2
0	2	2	2	1	0	0	0	2	0	0	1	0		2	1	1	2	1	2	0	0	0	1	2	2
0	1	1	2	2	1	1	1	1	1	1	1	2	…	2	1	0	2	0	1	1	0	0	1	2	2
0	2	1	2	1	0	0	0	2	0	0	1	0		2	1	0	2	1	1	1	0	0	1	2	2
0	1	2	2	1	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	0	0	2	1	1	1	0	0	0	2	2
0	2	0	2	2	0	0	0	2	0	0	1	1		2	0	1	1	1	2	0	1	1	0	2	1
0	1	1	2	2	1	1	1	1	1	1	0	1	…	2	1	0	2	0	1	1	0	0	1	2	2
0	0	2	2	2	0	0	0	2	0	0	2	2		2	2	1	2	2	2	0	0	0	1	2	2
0	1	1	2	2	1	0	0	2	0	0	2	2		2	1	1	2	1	1	0	0	0	0	2	2
0	1	2	2	1	0	0	0	2	0	0	2	1		2	1	1	1	1	2	0	1	1	0	2	1
0	2	0	2	2	0	0	0	2	0	0	0	0		2	1	0	2	1	2	0	0	0	0	2	2

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

Out[86]:
```
9000x50 Array{Int64,2}:
 2  0  0  0  0  1  1  0  1  1  0  2  0  …  1  2  0  2  0  0  1  0  0  1  1  2
 1  0  0  0  0  1  0  0  2  2  1  1  1     2  2  1  2  0  1  1  2  2  1  1  2
 1  1  1  1  0  0  0  0  2  2  0  2  0     1  2  0  2  0  2  0  2  0  0  0  2
 0  0  0  0  0  2  0  0  2  1  2  0  1     2  2  0  1  1  0  2  2  2  2  1  2
 0  1  1  1  0  1  1  0  1  0  0  0  2     1  1  0  2  0  2  0  2  1  1  0  2
 0  0  0  1  0  2  0  0  2  2  0  0  0  …  2  1  1  2  0  2  0  1  0  1  1  2
 2  2  2  0  0  1  0  0  2  2  1  1  0     0  2  1  2  0  1  0  1  1  2  1  2
 0  0  0  2  0  2  1  0  1  0  0  1  0     0  2  1  2  0  1  1  2  2  2  0  2
 2  1  1  0  1  2  2  0  0  1  1  0  1     2  1  0  2  0  0  2  2  0  0  1  2
 0  0  0  1  0  0  0  0  2  2  0  1  1     1  2  0  1  1  1  1  1  1  2  1  2
 1  1  0  2  0  1  1  0  1  1  0  0  2  …  0  1  1  2  1  2  0  1  0  0  0  2
 0  0  0  2  0  2  0  0  1  1  0  0  2     2  2  0  2  0  1  1  2  2  2  1  2
 1  1  1  1  1  1  0  0  1  1  1  2  0     0  1  0  2  1  1  0  2  0  1  1  2
 ⋮              ⋮              ⋮        ⋱     ⋮              ⋮
 0  0  0  1  0  1  1  0  1  1  0  1  0     1  1  1  2  0  1  0  0  0  1  1  1
 2  0  0  0  0  2  0  0  1  2  1  2  0     1  2  1  2  0  1  1  2  0  1  1  2
 1  1  1  1  0  1  0  0  2  1  1  2  0  …  2  2  1  1  0  0  2  2  0  0  1  2
 1  0  0  0  0  2  0  0  2  1  0  1  0     1  2  2  1  1  0  2  2  0  0  1  2
 2  0  0  1  0  2  0  0  2  2  1  2  1     0  1  0  1  0  0  2  1  0  1  1  2
 0  0  0  1  0  1  1  0  2  1  0  1  0     1  2  1  2  0  0  2  2  0  0  0  2
 0  0  0  1  0  0  0  0  2  2  0  0  0     1  0  1  1  0  0  2  1  0  1  0  2
 1  1  1  0  0  1  1  0  1  1  1  1  1  …  2  2  0  1  0  0  2  2  0  0  1  2
 2  0  0  0  0  1  0  0  2  1  2  1  1     1  1  1  2  0  0  2  2  0  1  1  2
 1  0  0  1  1  0  0  0  2  2  2  2  0     1  2  1  1  1  0  2  2  0  1  0  2
 2  0  0  1  0  1  0  1  2  2  1  0  1     2  2  1  2  0  0  2  1  1  1  0  2
 0  0  0  0  0  1  0  0  1  2  0  0  1     2  2  1  1  0  1  1  2  0  0  0  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]: 0.629149143670933

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

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

Out[100]: 0.7955334624765782

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

Out[101]: 48000x50 Array{Int64,2}:

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

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

Out[102]:
```
50-element Array{Float64,1}:
  0.165508
 -0.114896
 -0.233182
 -0.00827532
  0.171354
 -0.181134
  0.301283
  0.286223
 -0.294021
  0.0866416
  0.0325196
 -0.0681697
  0.215509
  ⋮
  0.034502
  0.316403
 -0.114806
  0.00873458
  0.244641
  0.25004
  0.0236308
  0.100374
 -0.062465
  0.196034
 -0.0916927
 -0.26713
```

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

Out[103]:
```
48000-element Array{Float64,1}:
 -0.00629114
  2.71178
  1.0639
  0.924131
  1.56606
  2.05978
  1.48831
  0.971828
  2.11843
  0.832425
  2.24159
  3.22546
  2.77455
  ⋮
  1.78412
  2.14402
  0.97048
  1.40901
  1.46836
  1.86013
 -0.284883
  1.30007
  1.81471
  2.62134
  1.91248
  1.76012
```

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

Out[104]:
```
1.4697732128198182
```

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

Out[105]:
```
1.4681637177872777
```

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

Out[106]:
```
1.492712187977908
```

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

Out[107]: 1.5215953772131674

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

Out[108]: 1.5374080261040248

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

Out[109]: 1.5821084256385263

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

Out[110]:
```
9000-element Array{Float64,1}:
  2.34358
 -0.017859
  1.3571
  1.19817
  1.59905
  0.307048
  0.693762
  1.58403
  1.59336
  1.81246
  2.01044
  0.500125
  1.64955
  ⋮
  1.78412
  2.14402
  0.97048
  1.40901
  1.46836
  1.86013
 -0.284883
  1.30007
  1.81471
  2.62134
  1.91248
  1.76012
```

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

Out[111]: 1.5762935755860106

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

Out[112]: 0.10652036276619237

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

Out[113]: 1.4739118504555577

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

Out[114]: 0.004138637635739517

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

Out[115]: 1.5128079910692802

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

Out[116]: 0.043034778249462

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

Out[117]: 1.5365593125992398

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

Out[118]: 0.06678609977942163

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

Out[119]: 1.5328960901543127

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

Out[120]: 0.06312287733449451

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

Out[121]: 1.5926986315510256

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

Out[122]: 0.12292541873120744

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

Out[123]: 1.5821084256385263

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

Out[124]: 0.11233521281870806