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

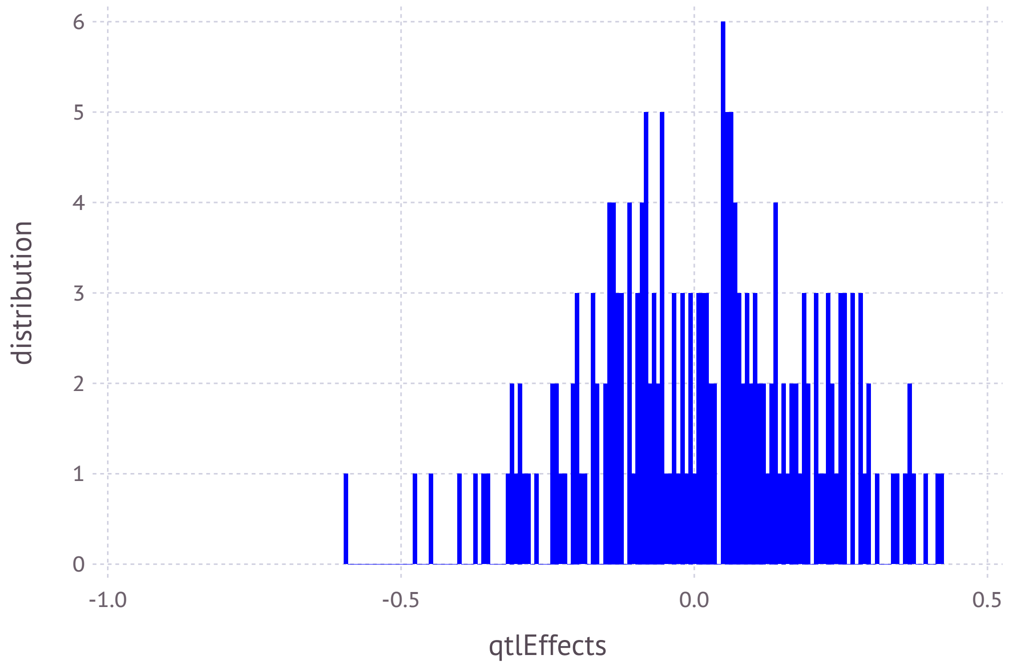
In [6]:
```
numChr    = 10
chrLength = 0.01
numLoci   = 20
nQTL      = 5
mutRate   = 0.0
locusInt  = chrLength/numLoci
mapPos    = collect(locusInt/2:locusInt:chrLength)
geneFreq  = fill(0.5,numLoci)
QTL = sample(1:numLoci,nQTL,replace=false)
qtlMarker = fill(false,numLoci)
qtlMarker[QTL] = true
Va = 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}:
         -0.0969943
          0.147682
          0.247411
         -0.0165462
         -0.374086
         -0.0354988
         -0.139635
          0.226643
         -0.135048
          0.0726994
          0.114301
         -0.202965
         -0.597707
          ⋮
         -0.0882726
          0.257159
          0.0932926
          0.048065
          0.296575
         -0.229716
          0.271303
         -0.10774
          0.00240593
         -0.0563711
          0.118733
          0.22583
```

```
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]: 0.012510633634305766
```

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

```
Out[11]: 0.036103307073694926
```

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
```

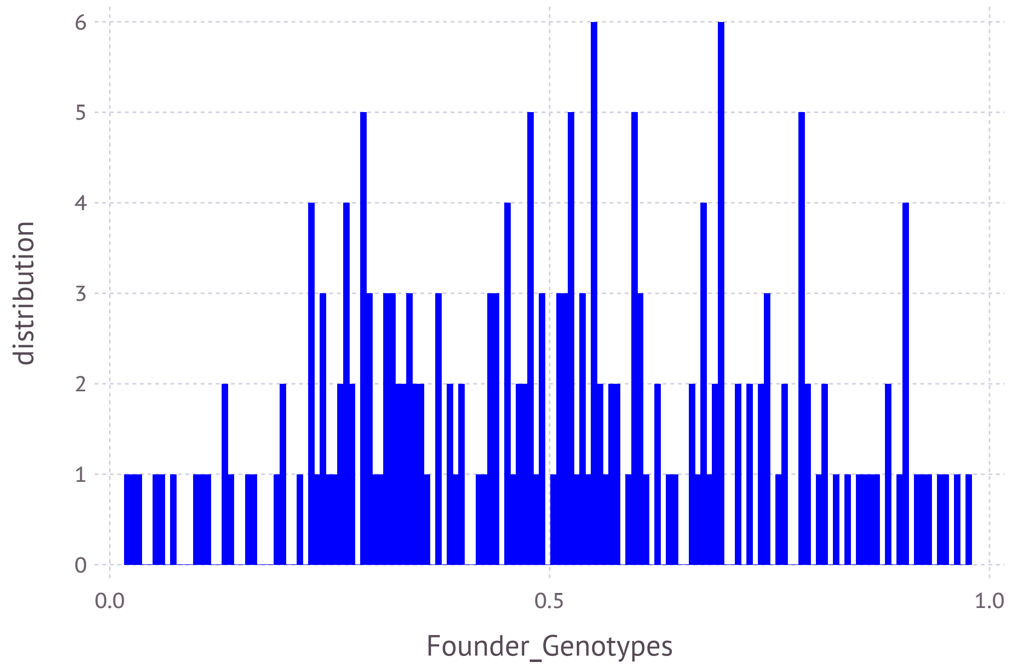
In [19]:
```
gSPSire = XSim.getOurGenotypes(popSP[1])
gSPDam = XSim.getOurGenotypes(popSP[2])
gSP = [gSPSire;gSPSire];
```

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

Out[20]: 1x200 Array{Float64,2}:
          0.059125  0.84025  0.290375  0.944875  …  0.371375  0.384  0.89625  0.534625
```

```
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);
```

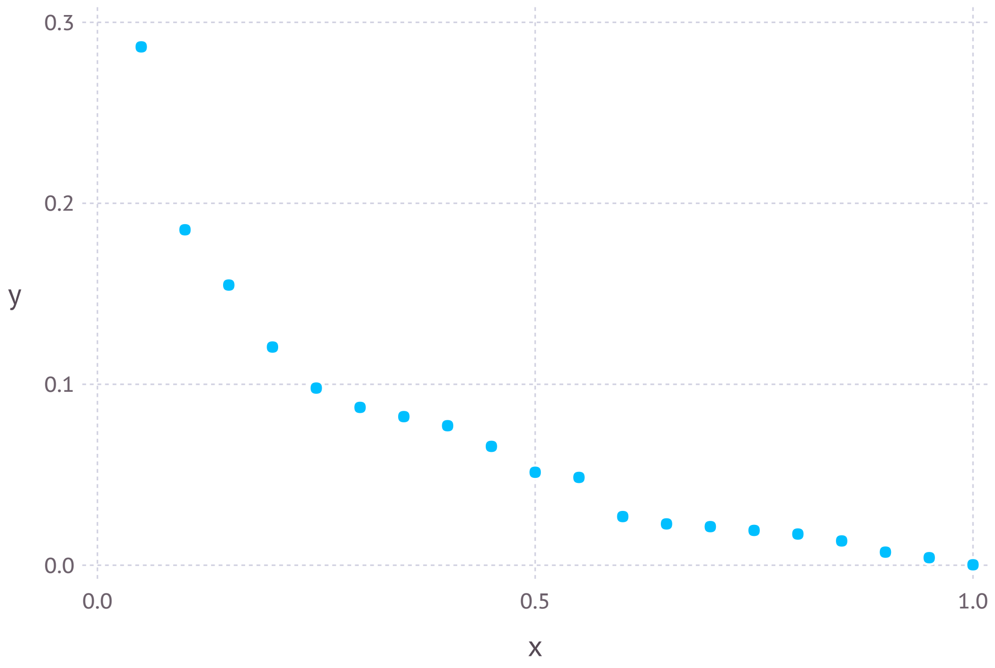
```
In [23]: for i=1:(nRows-20)
         LDMat[i,:] = corMat[i,(i+1):(i+20)].^2
         end
```

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

```
Out[24]: 1x20 Array{Float64,2}:
          0.00025703  0.00418538  0.00725611  …  0.154829  0.185392  0.286441
```

```
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]:
```
3.6482527319395466
```

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

Out[31]:
```
0.7672652869382454
```

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

Out[32]:
```
0.7672652869382454
```

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

Out[33]:
```
0.7672652869382454
```

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]:
```
6.2085754515401606
```

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

Out[36]:
```
6.2104773609429325
```

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

Out[37]: 0.4938710284245143
```

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

Out[38]: 0.49330489175798015
```

Pedigree: All animals

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

```
Out[39]: 48000x3 Array{Int64,2}:
          40722  34293  40184
          40723  35547  36942
          40724  33462  36990
          40725  34443  38763
          40726  34244  39550
          40727  36592  38079
          40728  33142  38935
          40729  35155  39501
          40730  34205  38364
          40731  34948  39661
          40732  36167  39154
          40733  32820  39190
          40734  35463  37552
              ⋮
          88710  75613  80129
          88711  76100  79502
          88712  76580  78151
          88713  73249  77913
          88714  74597  80274
          88715  76071  78673
          88716  75719  79520
          88717  75933  78960
          88718  76218  79011
          88719  74542  80580
          88720  73593  79118
          88721  73593  79420
```

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

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

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

Create marker file for all animals

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

Out[48]: 48000x200 Array{Int64,2}:
 0  2  0  2  2  0  0  0  2  0  0  2  2  …  2  0  1  2  2  1  0  0  0  1  2  1
 0  1  1  2  2  1  0  0  2  0  0  2  2     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     0  0  2  2  1  1  1  1  1  1  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  1  2  2  0  1  1  0  2  0
 0  2  1  2  1  0  0  0  2  0  0  2  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  0  1  2  2  2  0  0  0  1  2  1
 0  2  0  2  2  0  1  1  2  0  0  2  2     2  1  1  1  1  2  0  1  1  0  2  1
 0  2  1  2  1  0  0  0  2  0  0  2  1     2  2  1  1  1  1  0  1  1  0  1  1
 0  0  2  1  1  0  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  2  2  …  2  0  1  2  2  2  0  0  0  2  1  1
 0  2  0  2  2  0  0  1  1  1  1  0  1     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     1  1  2  1  1  1  1  1  1  1  2  1
 ⋮              ⋮              ⋮        ⋱        ⋮              ⋮
 0  1  1  2  2  1  1  1  1  1  1  0  1     2  1  2  0  2  2  0  2  2  0  2  0
 0  0  2  1  1  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  1  2  2  0  2  2  0  1  …  2  2  2  1  2  2  0  1  1  1  2  1
 0  0  2  2  1  2  2  2  0  2  2  0  1     2  2  2  0  2  2  0  2  2  0  2  0
 1  1  1  2  2  1  2  2  0  2  2  0  1     2  1  1  1  1  2  0  1  1  0  2  1
 0  1  1  2  2  0  1  1  1  1  1  1  1     1  1  2  1  2  2  1  1  1  0  1  0
 1  1  1  2  1  2  2  2  0  2  2  0  1     2  1  2  0  2  2  0  2  2  0  2  1
 0  0  2  2  2  1  2  2  0  2  2  0  1  …  2  2  2  0  2  2  0  2  2  0  2  0
 0  0  2  2  2  1  2  2  0  2  2  0  1     2  1  1  1  1  1  0  1  1  0  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  0  0  2  0  0  2  1     2  1  2  0  2  2  0  2  2  0  2  0
 0  0  2  1  0  1  2  2  0  2  2  0  1     2  1  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}:
          41120
          43994
          43665
          41567
          44354
          43832
          41172
          43654
          41169
          43487
          44578
          43908
          42067
              ⋮
          75613
          76100
          76580
          73249
          74597
          76071
          75719
          75933
          76218
          74542
          73593
          73593
```

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

```
Out[53]: 1000-element Array{Int64,1}:
          41120
          43994
          43665
          41567
          44354
          43832
          41172
          43654
          41169
          43487
          44578
          43908
          42067
              ⋮
          76674
          73610
          76506
          76279
          76300
          75920
          74423
          75742
          74841
          75827
          75932
          75102
```

```
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}:
          41120
          43994
          43665
          41567
          44354
          43832
          41172
          43654
          41169
          43487
          44578
          43908
          42067
              ⋮
          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  3.3    3.023
          40723  4.055  3.856
          40724  4.354  4.355
          40725  3.313  3.55
          40726  3.769  2.617
          40727  0.82   2.632
          40728  0.942  2.883
          40729  2.769  4.354
          40730  2.25   2.343
          40731  5.375  4.886
          40732  4.86   4.401
          40733  2.886  3.951
          40734  2.514  2.694
              ⋮
          88710  5.377  6.086
          88711  7.606  8.132
          88712  7.672  6.865
          88713  5.129  7.175
          88714  6.509  6.865
          88715  7.551  7.985
          88716  7.767  7.696
          88717  7.436  6.583
          88718  6.541  6.418
          88719  7.493  7.271
          88720  8.397  6.19
          88721  6.439  6.545
```

```
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
            4
            8
            9
           16
           23
           24
           28
           29
           36
           43
           44
           48
           ⋮
          149
          156
          163
          164
          168
          169
          176
          183
          184
          188
          189
          196
```

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

Out[83]:
```
150-element Array{Int64,1}:
   1
   2
   5
   6
   7
  10
  11
  12
  13
  14
  15
  17
  18
   ⋮
 186
 187
 190
 191
 192
 193
 194
 195
 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  0  2  2  1  1  1  1  2  …  2  1  0  2  1  2  0  0  0  0  2  2
          0  2  0  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  1  2  2  0  2  2  0  2     2  1  2  1  1  1  0  1  1  1  1  1
          0  2  1  2  1  0  1  1  2  0  0  2  2     2  2  2  0  2  2  0  2  2  0  2  0
          0  2  0  2  2  0  1  1  1  1  1  0  1     1  1  2  1  1  1  1  1  1  1  2  1
          0  0  2  2  2  2  2  2  0  2  2  0  2  …  2  0  0  2  1  2  1  1  1  1  1  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  1  2  2  1  0  1  1  1  1  1  1  0     2  1  2  1  1  2  0  1  1  0  1  1
          0  1  1  2  2  1  2  2  0  2  2  0  1     1  0  2  2  0  1  1  0  0  1  1  2
          0  2  0  2  2  0  1  1  1  1  1  0  1     2  1  2  1  2  2  0  1  1  1  1  1
          0  1  1  1  1  1  2  2  0  2  2  0  2  …  2  1  0  2  1  2  0  0  0  0  2  2
          1  2  0  2  2  1  1  1  1  1  1  1  2     2  2  2  0  1  1  1  1  1  1  2  1
          1  2  0  2  2  0  1  1  1  1  1  0  0     2  1  1  1  2  2  0  1  1  1  2  0
          ⋮              ⋮              ⋮        ⋱           ⋮              ⋮
          0  1  1  2  2  1  1  1  1  1  1  0  1     2  1  2  0  2  2  0  2  2  0  2  0
          0  0  2  1  1  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  1  2  2  0  2  2  0  1  …  2  2  2  1  2  2  0  1  1  1  2  1
          0  0  2  2  1  2  2  2  0  2  2  0  1     2  2  2  0  2  2  0  2  2  0  2  0
          1  1  1  2  2  1  2  2  0  2  2  0  1     2  1  1  1  1  2  0  1  1  0  2  1
          0  1  1  2  2  0  1  1  1  1  1  1  1     1  1  2  1  2  2  1  1  1  0  1  0
          1  1  1  2  1  2  2  2  0  2  2  0  1     2  1  2  0  2  2  0  2  2  0  2  1
          0  0  2  2  2  1  2  2  0  2  2  0  1  …  2  2  2  0  2  2  0  2  2  0  2  0
          0  0  2  2  2  1  2  2  0  2  2  0  1     2  1  1  1  1  1  0  1  1  0  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  0  0  2  0  0  2  1     2  1  2  0  2  2  0  2  2  0  2  0
          0  0  2  1  0  1  2  2  0  2  2  0  1     2  1  2  0  2  2  0  2  2  0  2  0
```

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

Out[86]: 9000x50 Array{Int64,2}:
          0  2  2  1  1  1  1  0  1  1  2  1  1  …  0  0  1  2  2  0  1  2  2  0  2  0
          0  2  1  1  1  2  2  0  0  0  0  0  2     2  1  0  2  2  1  1  2  2  1  2  1
          1  1  2  0  1  1  1  1  2  2  1  1  2     2  2  1  2  2  1  1  2  2  1  2  1
          1  2  1  2  1  2  2  1  2  2  2  2  2     1  1  1  2  2  1  2  2  2  2  2  2
          0  2  1  1  1  1  1  0  2  2  2  1  1     2  1  1  2  2  0  1  1  2  1  1  1
          2  2  2  0  0  1  0  1  2  2  2  1  1  …  2  0  0  2  2  0  1  2  1  0  2  1
          0  2  0  2  2  1  1  0  1  1  1  1  1     2  1  0  2  2  1  1  2  2  2  2  2
          2  2  1  1  1  1  1  1  2  2  2  2  2     1  1  0  2  1  1  1  2  2  1  2  1
          1  2  2  0  0  0  0  0  2  2  1  1  1     1  0  0  2  1  0  0  1  1  0  1  0
          0  2  1  1  0  1  1  1  2  2  2  2  2     2  1  0  2  2  2  1  2  2  2  2  1
          1  1  2  0  0  2  1  1  2  2  0  0  1  …  1  1  0  2  1  1  0  2  2  0  2  0
          0  2  1  1  2  1  1  1  2  2  1  0  0     1  1  0  2  2  0  1  2  2  2  2  1
          0  2  1  1  1  1  1  0  1  1  2  2  2     2  0  0  2  2  2  0  2  1  1  2  1
          ⋮              ⋮              ⋮           ⋱        ⋮              ⋮
          1  2  1  1  0  1  1  0  2  2  2  1  1     2  0  2  1  2  0  1  2  2  2  2  2
          2  1  2  0  1  2  1  2  2  2  0  0  2     1  1  1  2  2  0  2  2  2  2  2  2
          1  2  2  0  1  2  1  2  2  2  2  2  2  …  0  0  1  2  2  0  1  2  2  1  2  1
          2  2  2  0  1  1  1  1  1  1  2  2  2     1  1  1  2  2  0  0  2  2  2  2  2
          1  2  2  0  1  2  2  1  1  1  2  1  2     2  1  0  2  1  1  1  2  2  1  2  1
          1  2  1  1  2  1  0  1  2  2  2  2  2     2  2  2  2  2  0  2  2  2  2  1  1
          1  2  2  0  2  2  1  2  2  2  2  2  2     2  2  0  2  2  0  2  2  2  1  2  2
          2  2  2  0  1  2  1  2  2  2  2  1  1  …  2  1  0  2  2  0  2  2  2  2  2  2
          2  2  2  0  1  2  2  1  1  1  2  2  1     0  0  0  2  2  1  0  2  2  1  2  1
          1  1  1  1  0  2  0  2  2  2  2  2  2     1  1  1  2  2  2  1  2  2  2  2  2
          1  1  0  2  1  2  2  2  2  2  1  1  2     1  0  0  2  1  0  1  2  2  2  2  2
          2  1  2  0  1  2  2  1  2  2  2  1  1     1  0  1  2  1  0  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]:
```
0.6299396062162367
```

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

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

Out[100]:
```
0.791402587923926
```

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

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

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

```
Out[102]: 50-element Array{Float64,1}:
            0.247411
           -0.0165462
            0.226643
           -0.135048
            0.099979
           -0.142188
            0.0788742
            0.155559
            0.218642
           -0.293835
            0.231313
           -0.132508
           -0.198957
            ⋮
            0.246054
           -0.0611037
           -0.0802046
            0.272102
           -0.350734
           -0.473415
            0.020689
            0.077208
           -0.166715
            0.110848
           -0.0882726
           -0.10774
```

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

Out[103]:
```
48000-element Array{Float64,1}:
 -0.0794819
  0.0751279
  0.0658285
 -0.307206
 -0.211082
  1.39428
  0.769341
  0.0864494
  1.32647
  1.23909
  0.504147
  0.290332
  0.985412
  ⋮
 -0.208814
  1.00227
  0.590372
  1.13034
  1.8123
  0.940077
  1.76381
  1.5325
  1.59865
 -0.0142094
  0.652584
  1.81938
```

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

Out[104]:
```
0.5395591658177789
```

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

Out[105]:
```
0.5881696258920688
```

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

Out[106]:
```
0.6439802870456152
```

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

Out[107]: 0.7472600203742952

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

Out[108]: 0.8654401428806421

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

Out[109]: 0.9429002530780374

In [110]: EAlphaG=onlyQTL*QTLo

Out[110]: 9000-element Array{Float64,1}:
            0.57729
           -0.298402
            1.01864
            0.179759
            0.730358
            1.53488
           -0.270298
            1.37975
            0.831321
            1.20389
            0.754483
            0.9875
            0.0337935
            ⋮
           -0.208814
            1.00227
            0.590372
            1.13034
            1.8123
            0.940077
            1.76381
            1.5325
            1.59865
           -0.0142094
            0.652584
            1.81938
```

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

Out[111]: 0.9320341590416799

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

Out[112]: 0.392474993223901

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

Out[113]: 0.63219803697969

In [114]: meanBreedingValueSelectedG0=meanEAlphaS0-meanEAlphaG0

Out[114]: 0.09263887116191116

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

Out[115]: 0.7068172611723053

In [116]: meanBreedingValueSelectedG1=meanEAlphaS1-meanEAlphaG0

Out[116]: 0.1672580953545264

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

Out[117]: 0.8480009226631599

In [118]: meanBreedingValueSelectedG2=meanEAlphaS2-meanEAlphaG0

Out[118]: 0.308441756845381

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

Out[119]: 1.0097878138580942

In [120]: meanBreedingValueSelectedG3=meanEAlphaS3-meanEAlphaG0

Out[120]: 0.4702286480403154
```

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

Out[121]: 1.0287229990808482

In [122]: meanBreedingValueSelectedG4=meanEAlphaS4-meanEAlphaG0

Out[122]: 0.48916383326306934

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

Out[123]: 0.9429002530780374

In [124]: meanBreedingValueSelectedG5=meanEAlphaS5-meanEAlphaG0

Out[124]: 0.4033410872602585
```