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

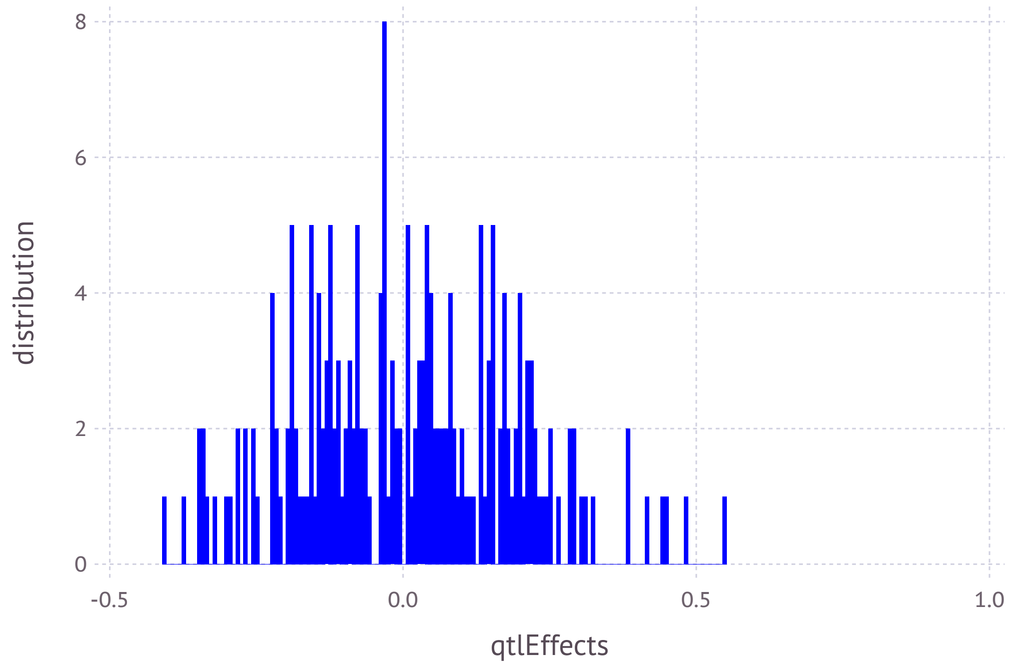
In [6]:
```
numChr    = 10
chrLength = 0.01
numLoci   = 20
nQTL      = 5
mutRate   = 0.0
locusInt  = chrLength/numLoci
mapPos    = collect(locusInt/2:locusInt:chrLength)
geneFreq  = fill(0.5,numLoci)
QTL = sample(1:numLoci,nQTL,replace=false)
qtlMarker = fill(false,numLoci)
qtlMarker[QTL] = true
Va = 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}:
         -0.0726127
          0.0333103
         -0.149569
          0.0804493
         -0.206486
          0.181822
         -0.170719
          0.200358
          0.0742214
         -0.155552
         -0.0296226
         -0.0766402
          0.027164
          ⋮
          0.167045
          0.164043
         -0.157787
         -0.0805932
         -0.345163
         -0.25454
         -0.219358
         -0.0593345
          0.308476
          0.0455779
         -0.0965569
          0.103331
```

```
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]: 0.006822013820960401
```

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

```
Out[11]: 0.034033850052836725
```

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
```

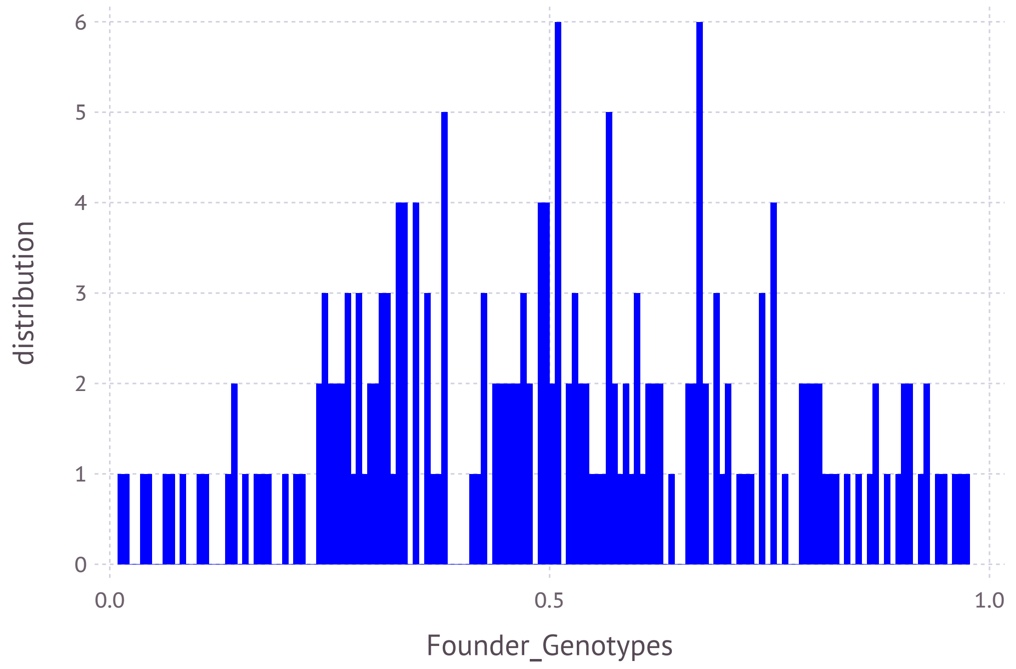
In [19]:
```
gSPSire = XSim.getOurGenotypes(popSP[1])
gSPDam = XSim.getOurGenotypes(popSP[2])
gSP = [gSPSire;gSPSire];
```

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

Out[20]:
```
1x200 Array{Float64,2}:
 0.071625  0.84125  0.282875  0.950625  …  0.379875  0.38275  0.90225  0.528
```

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);
```

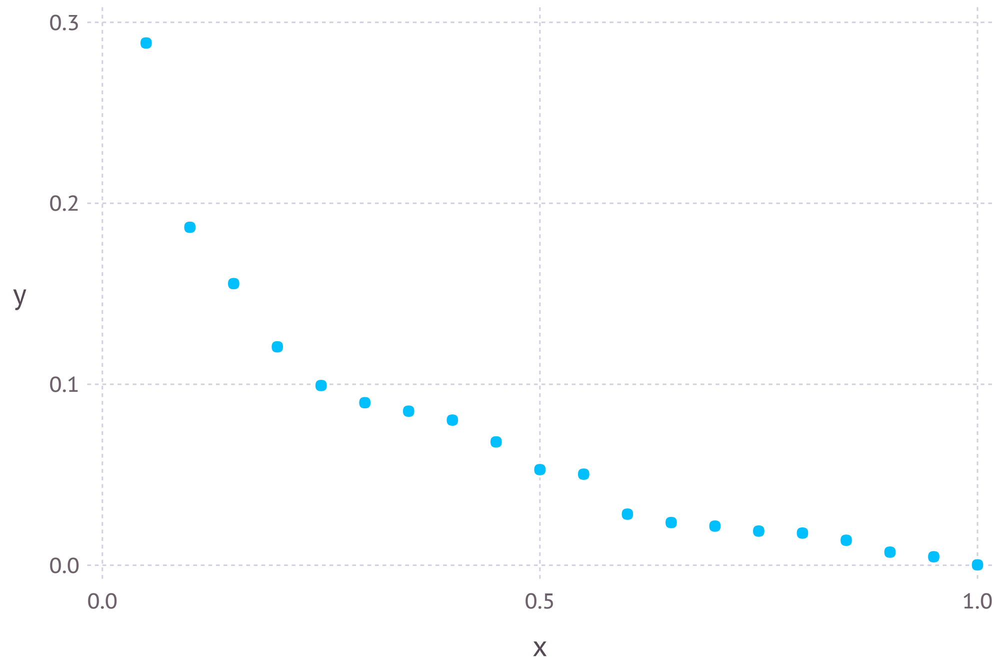
In [23]:
```
for i=1:(nRows-20)
LDMat[i,:] = corMat[i,(i+1):(i+20)].^2
end
```

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

Out[24]:
```
1x20 Array{Float64,2}:
 0.000246285  0.00469892  0.00725766  …  0.155643  0.186772  0.288589
```

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]: 2.760861967136385
```

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

```
Out[31]: 0.3230994792593835
```

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

```
Out[32]: 0.3230994792593835
```

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

```
Out[33]: 0.3230994792593835
```

```
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]: 4.56306034852921
```

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

```
Out[36]: 4.549616412549265
```

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

Out[37]: 0.26931104567749353
```

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

Out[38]: 0.26870775622557025
```

Pedigree: All animals

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

Out[39]:
```
48000x3 Array{Int64,2}:
 40722  32790  38855
 40723  35427  38734
 40724  35167  39565
 40725  33311  38510
 40726  35190  40361
 40727  36537  40213
 40728  32849  36820
 40729  33487  39323
 40730  35890  40169
 40731  33243  37701
 40732  36274  39538
 40733  35898  39745
 40734  33726  38970
     ⋮
 88710  75955  77056
 88711  75400  77762
 88712  73798  79440
 88713  73079  79126
 88714  75068  78575
 88715  74222  78339
 88716  75729  77657
 88717  74102  79173
 88718  73323  80458
 88719  73947  80151
 88720  75949  80705
 88721  76592  79411
```

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

```
In [46]: allID = GT[:,1];
In [47]: GTM = GT[:,2:end];
```

Create marker file for all animals

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

Out[48]: 48000x200 Array{Int64,2}:
          0  2  1  2  1  0  0  0  2  0  0  1  0  …  2  1  1  1  1  2  1  1  1  1  1  1
          0  2  0  2  2  0  0  0  2  0  0  1  1     0  0  2  2  0  0  2  0  0  2  2  2
          0  2  0  2  2  0  0  0  2  0  0  2  2     2  2  1  1  2  2  0  1  1  0  2  1
          0  2  0  2  2  0  0  0  2  0  0  1  1     1  1  1  2  1  1  1  0  0  1  2  2
          0  1  1  2  2  1  1  1  1  1  1  0  1     1  1  2  1  1  1  1  1  1  1  2  1
          0  2  1  2  1  0  0  0  2  0  0  2  1  …  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  2  1  0  0  0  0  2  2
          0  2  0  2  2  0  0  0  2  0  0  1  1     1  1  1  2  0  0  1  0  0  1  1  2
          0  2  0  2  2  1  1  1  1  1  1  1  2     2  2  2  1  2  2  0  1  1  1  2  1
          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
          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
          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  2  0  1  1  1  1  1  1  2     2  1  2  1  2  2  0  1  1  0  2  1
          ⋮              ⋮              ⋮        ⋱        ⋮              ⋮
          1  2  0  2  2  1  2  2  1  1  1  1  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  2  2  0  2  2  0  2  2  0  2  0
          1  2  0  2  2  2  1  2  1  1  1  0  1  …  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  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
          1  2  0  2  2  0  1  1  1  1  1  0  0     2  2  2  0  2  2  0  2  2  0  2  0
          0  2  0  2  2  0  2  2  1  1  1  1  2     2  2  2  0  2  2  0  2  2  0  2  0
          0  2  0  2  2  1  2  2  0  2  2  0  2  …  2  2  2  0  2  2  0  2  2  0  2  0
          1  2  0  2  2  2  2  2  0  2  2  0  2     1  1  2  1  1  1  1  1  1  1  2  1
          0  2  0  2  2  2  2  2  0  2  2  0  2     2  2  2  0  2  2  0  2  2  0  2  0
          0  1  1  2  1  2  2  2  0  2  2  0  2     2  2  2  0  2  2  0  2  2  0  2  0
          1  2  0  2  2  1  2  2  0  2  2  0  1     2  2  1  1  1  1  1  1  1  1  2  1
```

```
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}:
          41918
          43230
          41431
          42642
          42660
          41055
          40775
          43211
          41492
          43708
          42509
          44004
          41427
              ⋮
          75955
          75400
          73798
          73079
          75068
          74222
          75729
          74102
          73323
          73947
          75949
          76592
```

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

```
Out[53]: 1000-element Array{Int64,1}:
          41918
          43230
          41431
          42642
          42660
          41055
          40775
          43211
          41492
          43708
          42509
          44004
          41427
              ⋮
          75785
          76269
          73834
          76437
          75250
          73489
          75484
          73783
          74737
          75204
          73399
          73117
```

```
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}:
          41918
          43230
          41431
          42642
          42660
          41055
          40775
          43211
          41492
          43708
          42509
          44004
          41427
              ⋮
          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  2.843  2.152
          40723  2.919  2.833
          40724  2.953  2.912
          40725  2.92   2.26
          40726  2.715  2.458
          40727  4.55   3.223
          40728  3.172  2.377
          40729  4.297  2.584
          40730  3.746  3.014
          40731  1.538  1.953
          40732  3.963  3.844
          40733  4.167  3.471
          40734  2.175  2.784
              ⋮
          88710  4.181  4.644
          88711  5.527  5.858
          88712  5.358  5.749
          88713  4.204  4.432
          88714  6.894  5.637
          88715  4.738  5.224
          88716  6.49   5.697
          88717  5.829  5.431
          88718  4.311  4.321
          88719  5.078  5.454
          88720  4.992  5.528
          88721  4.621  4.723
```

```
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
            6
            8
           18
           19
           23
           26
           28
           38
           39
           43
           46
           48
            ⋮
          158
          159
          163
          166
          168
          178
          179
          183
          186
          188
          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
            7
            9
           10
           11
           12
           13
           14
           15
           16
            ⋮
          185
          187
          189
          190
          191
          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  0  2  2  1  2  2  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  1     1  0  1  2  1  1  1  0  0  2  2  1
          0  1  1  2  2  1  1  1  1  1  1  0  2     1  0  1  2  1  1  1  0  0  2  2  1
          0  1  1  1  1  0  1  1  2  0  0  0  0     1  1  1  2  1  1  1  1  1  1  2  1
          1  2  0  2  2  0  1  1  1  1  1  1  1     2  2  1  1  1  1  0  1  1  0  1  1
          0  1  1  2  2  1  2  2  1  1  1  1  2  …  1  1  1  2  0  2  0  0  0  1  1  2
          0  1  1  2  2  1  1  1  1  1  1  0  2     1  1  2  2  1  1  1  0  0  1  2  1
          0  2  0  2  2  0  1  1  1  1  1  1  1     2  2  1  1  1  2  1  1  1  1  1  1
          1  2  0  2  2  1  2  2  2  0  0  2  1     1  0  2  2  1  0  2  0  0  2  2  2
          0  1  1  2  2  1  1  1  1  1  1  0  1     2  0  1  2  0  1  0  0  0  0  2  2
          0  2  0  2  2  1  1  1  1  1  1  1  2  …  1  0  1  2  0  0  2  0  0  2  2  2
          0  2  0  2  2  1  1  1  1  1  1  0  1     2  2  2  0  2  2  0  2  2  0  2  0
          0  2  0  2  2  0  0  0  2  0  0  0  0     1  1  2  2  1  1  1  0  0  2  2  1
          ⋮              ⋮              ⋮        ⋱        ⋮              ⋮
          1  2  0  2  2  1  2  2  1  1  1  1  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  2  2  0  2  2  0  2  2  0  2  0
          1  2  0  2  2  2  1  2  1  1  1  0  1  …  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  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
          1  2  0  2  2  0  1  1  1  1  1  0  0     2  2  2  0  2  2  0  2  2  0  2  0
          0  2  0  2  2  0  2  2  1  1  1  1  2     2  2  2  0  2  2  0  2  2  0  2  0
          0  2  0  2  2  1  2  2  0  2  2  0  2  …  2  2  2  0  2  2  0  2  2  0  2  0
          1  2  0  2  2  2  2  2  0  2  2  0  2     1  1  2  1  1  1  1  1  1  1  2  1
          0  2  0  2  2  2  2  2  0  2  2  0  2     2  2  2  0  2  2  0  2  2  0  2  0
          0  1  1  2  1  2  2  2  0  2  2  0  2     2  2  2  0  2  2  0  2  2  0  2  0
          1  2  0  2  2  1  2  2  0  2  2  0  1     2  2  1  1  1  1  1  1  1  1  2  1
```

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

Out[86]:
```
9000x50 Array{Int64,2}:
 0  1  2  1  0  1  1  0  1  1  2  1  1  …  1  2  0  1  2  1  1  2  2  2  0  2
 0  0  1  1  0  2  0  0  2  0  0  1  0     1  2  1  1  2  0  1  1  1  1  2  2
 1  1  1  0  0  2  1  0  2  1  1  1  1     1  0  1  1  2  1  0  1  0  0  2  2
 1  0  1  0  0  1  0  1  1  1  1  0  1     1  2  0  1  2  1  0  1  1  0  1  2
 0  0  1  1  0  0  1  1  1  1  0  2  1     0  1  0  1  2  1  0  1  1  0  0  1
 1  1  2  1  0  1  1  1  1  1  2  0  2  …  1  2  0  0  2  2  0  2  1  0  1  1
 1  1  1  0  0  1  1  2  0  0  0  1  0     1  1  0  2  1  1  0  1  1  1  1  2
 0  0  1  1  0  1  0  1  1  1  1  1  1     1  2  0  1  2  1  1  2  2  1  1  1
 0  1  2  1  0  0  0  0  2  2  2  1  1     0  0  0  2  2  0  2  1  1  1  2  2
 1  1  1  0  0  1  1  2  1  1  0  0  0     2  1  0  2  1  1  1  2  0  0  0  2
 0  1  1  1  0  1  1  2  1  0  1  1  1  …  1  2  0  2  2  1  1  0  1  0  2  2
 0  1  1  1  0  0  0  0  2  2  1  0  2     2  2  0  1  1  1  1  2  2  2  0  2
 0  0  0  0  0  1  1  0  2  2  1  2  0     1  1  1  1  2  2  0  1  1  1  2  2
 ⋮              ⋮              ⋮        ⋱        ⋮              ⋮
 0  1  2  1  0  1  1  0  1  2  1  2  0     0  2  0  2  2  1  1  2  2  2  0  2
 0  1  1  0  0  1  2  1  2  2  1  1  1     1  2  0  2  2  1  1  2  2  2  0  2
 0  2  2  1  1  0  0  0  2  2  0  2  1  …  1  2  0  2  2  1  1  1  1  1  1  2
 1  1  1  1  0  1  1  0  2  2  0  2  0     2  2  0  1  2  1  0  1  1  1  1  2
 0  0  0  1  0  0  2  2  2  1  0  2  2     1  2  0  2  1  1  1  2  2  2  0  2
 0  0  1  0  0  0  1  2  1  1  0  1  1     1  2  0  2  2  2  0  2  2  2  0  2
 0  0  2  1  0  0  2  2  0  0  0  0  1     2  2  0  2  2  2  0  2  2  2  0  2
 0  1  2  0  0  0  1  1  1  0  0  2  1  …  0  2  0  2  2  0  0  2  2  2  0  2
 0  2  2  0  0  0  1  1  2  2  1  1  0     1  2  0  1  2  2  0  1  1  1  1  2
 0  2  2  0  0  1  1  2  0  0  0  1  1     1  2  1  2  2  2  0  2  2  2  0  2
 1  2  2  0  0  0  2  2  0  0  1  1  1     0  2  0  2  2  1  1  2  2  2  0  2
 0  1  2  0  1  1  1  1  1  0  1  1  1     2  2  0  1  2  1  1  2  2  1  1  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]:
```
0.6064871272370974
```

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

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

```
Out[100]: 0.7799749874976987
```

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

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

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

```
Out[102]: 50-element Array{Float64,1}:
           -0.149569
            0.181822
            0.200358
            0.0364853
            0.0345378
           -0.0397492
            0.0530153
            0.0104987
           -0.251196
           -0.134451
           -0.0848827
           -0.102776
            0.0988489
            ⋮
           -0.223872
            0.174954
            0.170069
            0.0108448
           -0.342788
           -0.124895
            0.0706472
            0.0130358
           -0.409997
           -0.321945
            0.0455779
           -0.0965569
```

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

```
Out[103]: 48000-element Array{Float64,1}:
            1.36712
            0.935816
            0.0743273
            1.29309
            0.196495
           -0.192054
           -0.377776
            0.851681
           -0.530021
           -2.09583
           -0.899233
           -0.297922
            0.0333085
            ⋮
            1.24627
           -0.231791
            0.49228
            0.480204
           -0.602507
           -0.69156
           -0.807322
            0.386775
            0.29766
            0.0274147
            0.851479
            0.59262
```

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

```
Out[104]: 0.12080874740434926
```

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

```
Out[105]: 0.10081419428012037
```

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

```
Out[106]: 0.12864946354998236
```

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

Out[107]: 0.06356316767473408

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

Out[108]: 0.10403846293515909

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

Out[109]: 0.13053348889670896

In [110]: EAlphaG=onlyQTL*QTLo

Out[110]: 9000-element Array{Float64,1}:
           -0.222864
           -0.190687
            0.976479
            0.709272
            0.636861
            1.0767
            0.369808
            0.143809
            1.26542
            0.681257
            0.957017
           -0.138438
           -0.879752
            ⋮
            1.24627
           -0.231791
            0.49228
            0.480204
           -0.602507
           -0.69156
           -0.807322
            0.386775
            0.29766
            0.0274147
            0.851479
            0.59262
```

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

Out[111]: 0.12769880351606558

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

Out[112]: 0.006890056111716322

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

Out[113]: 0.09655708042504388

In [114]: meanBreedingValueSelectedG0=meanEAlphaS0-meanEAlphaG0

Out[114]: -0.024251666979305378

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

Out[115]: 0.13634569286912865

In [116]: meanBreedingValueSelectedG1=meanEAlphaS1-meanEAlphaG0

Out[116]: 0.015536945464779392

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

Out[117]: 0.003207624030735375

In [118]: meanBreedingValueSelectedG2=meanEAlphaS2-meanEAlphaG0

Out[118]: -0.11760112337361388

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

Out[119]: 0.14370827282122428

In [120]: meanBreedingValueSelectedG3=meanEAlphaS3-meanEAlphaG0

Out[120]: 0.022899525416875025
```

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

Out[121]: 0.14528793220846034

In [122]: meanBreedingValueSelectedG4=meanEAlphaS4-meanEAlphaG0

Out[122]: 0.024479184804111084

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

Out[123]: 0.13053348889670896

In [124]: meanBreedingValueSelectedG5=meanEAlphaS5-meanEAlphaG0

Out[124]: 0.009724741492359704
```