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 = 0.1
# Phenotypes : all animals in G0 to G4
# Genotypes  : all progeny in G5 and all sires in each generation
# Change muAlpha = 0.2
# 10 chromosomes; 20 loci per chromosome = > 200 Loci (50 QTL & 150 Markers)
```

In [2]:
```
include("/home/nicole/Jupyter/XSimSel.jl")
```

Out[2]:
```
XSim
```

In [3]:
```
using DataFrames
```

In [4]:
```
using Distributions
```

In [5]:
```
using(Gadfly)
```

Initialize XSim

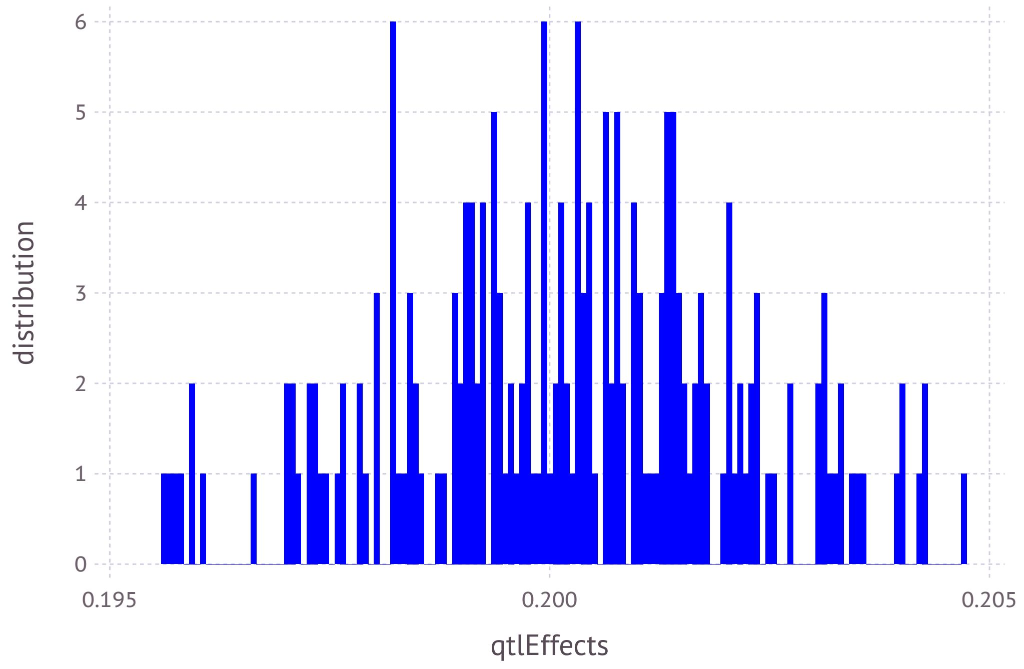
In [6]:
```
numChr    = 10
chrLength = 0.01
numLoci   = 20
nQTL      = 5
mutRate   = 0.0
locusInt  = chrLength/numLoci
mapPos    = collect(locusInt/2:locusInt:chrLength)
geneFreq  = fill(0.5,numLoci)
QTL = sample(1:numLoci,nQTL,replace=false)
qtlMarker = fill(false,numLoci)
qtlMarker[QTL] = true
mu = 100                                #  alpha ~ N(100,1)
Va = nQTL*numChr*0.5*mu*mu              # Va= nQTL*2pq*mean(alpha)^2
qtlEffects= rand(Normal(mu/sqrt(Va), sqrt(1/Va)),numLoci*numChr)  # Let alpha mu = 0.115
XSim.init(numChr,numLoci,chrLength,geneFreq,mapPos,qtlMarker,qtlEffects,mutRate)
```

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

```
Out[7]: 200-element Array{Float64,1}:
          0.201709
          0.201407
          0.200674
          0.201287
          0.199938
          0.201326
          0.202113
          0.198407
          0.198038
          0.203229
          0.197001
          0.199917
          0.203286
          ⋮
          0.196621
          0.202286
          0.204226
          0.198815
          0.197838
          0.201983
          0.203048
          0.197822
          0.202181
          0.198055
          0.200032
          0.203554
```

```
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]: 0.20020411300544777
```

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

```
Out[11]: 3.722169769539395 3e-6
```

In [12]:
```
# Base Population
gen=0
nGenBase    = 5
popSizeBase = 8000

# Sample 20 sire and 400 dams
popSizeSP = 8000

# Purbred Populations - mating
popSize = 8000
nGener  = 5
nSires  = 200
nDams   = 4000
npop = 1
;
```

In [13]:

```
# Animals with genotypes
posOFF0S = 1
posOFF0E = popSizeSP
posOFF1S = posOFF0E + 1
posOFF1E = posOFF0E + popSize
posOFF2S = posOFF1E + 1
posOFF2E = posOFF1E + popSize
posOFF3S = posOFF2E + 1
posOFF3E = posOFF2E + popSize
posOFF4S = posOFF3E + 1
posOFF4E = posOFF3E + popSize
posOFF5S = posOFF4E + 1
posOFF5E = posOFF4E + popSize
println("posOFF1S: ",posOFF1S,"; posOFF1E: ",posOFF1E)
println("posOFF2S: ",posOFF2S,"; posOFF2E: ",posOFF2E)
println("posOFF3S: ",posOFF3S,"; posOFF3E: ",posOFF3E)
println("posOFF4S: ",posOFF4S,"; posOFF4E: ",posOFF4E)
println("posOFF5S: ",posOFF5S,"; posOFF5E: ",posOFF5E)

# FileName :
PedAll = "PedAll.txt"           # pedigree  file with all animals
PheAll = "PheAll.txt"           # phenotype file with all animsla
GenAll = "GenAll.txt"           # genotype  file with all animals

Gen = "Gen.txt"                 # genotype file with all progeny in G5 and all sires in each generation
Phe = "Phe.txt"                 # phenotype file with all animals in G1 to G4
QTL = "QTL.txt"                 # QTL with with all progeny in G5 and all sires in each generation
Mar = "Mar.txt"                 # Marker with with all progeny in G5 and all sires in each generation
GenNF = "GenNF.txt"             # remove fixed genes from genotype file
QTLNF = "QTLNF.txt"             # remove fixed genes from QTL file
MarNF = "MarNF.txt"             # remove fixed genes from Marker file
;
```

```
posOFF1S: 8001; posOFF1E: 16000
posOFF2S: 16001; posOFF2E: 24000
posOFF3S: 24001; posOFF3E: 32000
posOFF4S: 32001; posOFF4E: 40000
posOFF5S: 40001; posOFF5E: 48000
```

In [14]:
```
fileName = "SS"
pedText = fileName * ".ped"
genText = fileName * ".gen"
pheText = fileName * ".phe"
;
```

In [15]:
```
;rm $pedText $genText $pheText
```

```
rm: cannot remove 'SS.ped': No such file or directory
rm: cannot remove 'SS.gen': No such file or directory
rm: cannot remove 'SS.phe': No such file or directory
```

Sampling Founders

In [16]:
```
sires = XSim.sampleFounders("/home/nicole/Jupyter/HERdata/data/Clean/newdata/ch1to10.200SNP.group1")
dams  = XSim.sampleFounders("/home/nicole/Jupyter/HERdata/data/Clean/newdata/ch1to10.200SNP.group2");
```

```
Sampling 360 animals into base population.
Sampling 361 animals into base population.
```

Base population

In [17]:
```
baseSires, baseDams, baseGen = XSim.sampleRan(popSizeBase, nGenBase, sires, dams, gen=gen);
```

```
Generation     1: sampling  4000 males and  4000 females
Generation     2: sampling  4000 males and  4000 females
Generation     3: sampling  4000 males and  4000 females
Generation     4: sampling  4000 males and  4000 females
Generation     5: sampling  4000 males and  4000 females
```

Sample animals for sire and dam candidates

In [18]:
```
popSP = XSim.sampleRan(popSizeSP, 1, baseSires, baseDams, gen=baseGen,fileName=fileName);
```

```
Generation     6: sampling  4000 males and  4000 females
```

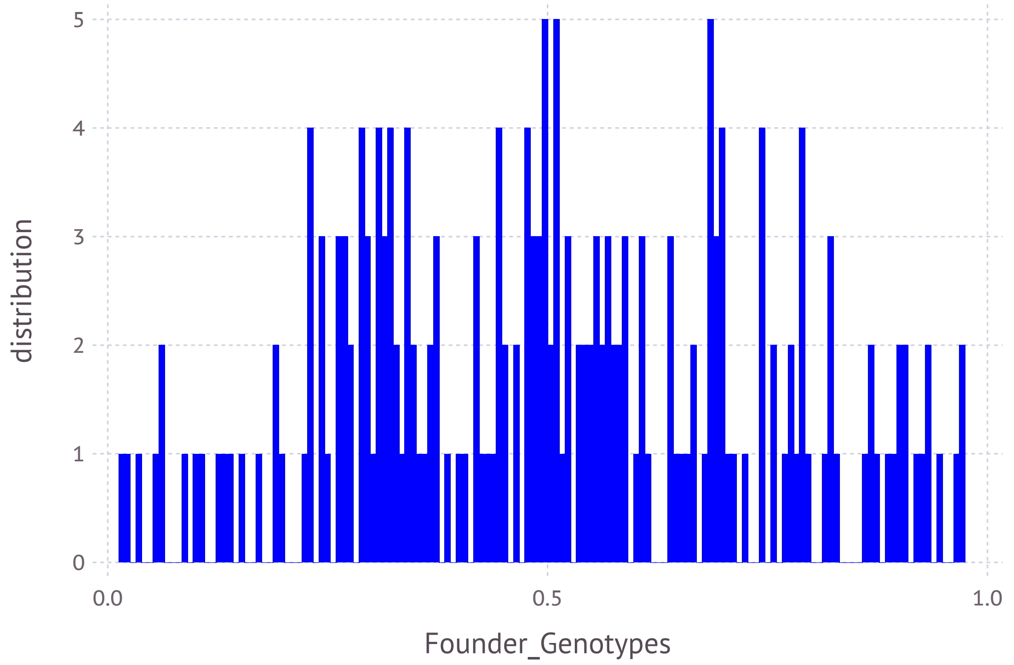
In [19]:
```
gSPSire = XSim.getOurGenotypes(popSP[1])
gSPDam = XSim.getOurGenotypes(popSP[2])
gSP = [gSPSire;gSPSire];
```

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

Out[20]:
```
1x200 Array{Float64,2}:
 0.060375  0.831375  0.287875  0.94625  …  0.375  0.3735  0.90175  0.545625
```

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);
```

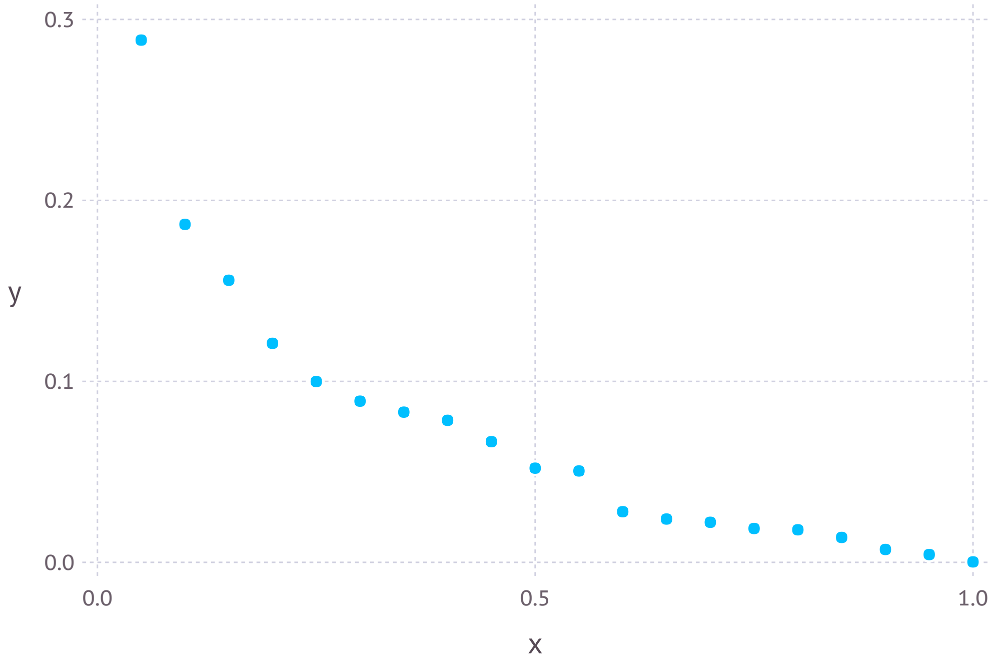
```
In [23]: for i=1:(nRows-20)
         LDMat[i,:] = corMat[i,(i+1):(i+20)].^2
         end
```

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

```
Out[24]: 1x20 Array{Float64,2}:
          0.000257163  0.00433186  0.00712074  …  0.155905  0.18676  0.288599
```

```
In [25]: plot(x=(1:20)/20*1,y=y)
```

Out[25]:



```
In [26]: FCMstream = open("SNPCMF.txt", "w")
```

```
Out[26]: IOStream(<file SNPCMF.txt>)
```

In [27]:
```
for i in 1:size(FCM,2)
    @printf(FCMstream, "%6.4f ", FCM[1,i])
end
```

In [28]:
```
close(FCMstream)
```

Selection - increase

In [29]:
```
aSPSire = XSim.getOurGenVals(popSP[1])
aSPDam = XSim.getOurGenVals(popSP[2])
aSP = [aSPSire;aSPDam];
```

In [30]:
```
mean(aSP)
```

Out[30]:
```
9.867723954940688
```

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

Out[31]:
```
0.5768046390677162
```

In [32]:
```
XSim.common.varRes = 9*varGen    #heritability = 0.1
```

Out[32]:
```
5.191241751609446
```

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

Out[33]:
```
5.191241751609446
```

In [34]:
```
popRMP = XSim.sampleSelNoLap(nSires, nDams, nGener,popSP[1],popSP[2],gen=popSP[3],fileName=fileName,direc
```

```
Generation     7: sampling  4000 males and  4000 females
Generation     8: sampling  4000 males and  4000 females
Generation     9: sampling  4000 males and  4000 females
Generation    10: sampling  4000 males and  4000 females
Generation    11: sampling  4000 males and  4000 females
```

```
In [35]: ymRMP = XSim.getOurGenVals(popRMP[1])    # for males: pop[1]
         mean(ymRMP)

Out[35]: 11.306631227839782

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

Out[36]: 11.31268718934223

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

Out[37]: 0.6025673906070986

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

Out[38]: 0.5944003098097455
```

Pedigree: All animals

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

Out[39]:
```
48000x3 Array{Int64,2}:
 40722  35544  39497
 40723  36167  38532
 40724  33215  36863
 40725  35742  39834
 40726  33763  37584
 40727  34996  37353
 40728  35382  40488
 40729  34159  36771
 40730  33268  39729
 40731  35171  40677
 40732  32752  39836
 40733  35843  38012
 40734  33935  40647
     ⋮
 88710  73619  80540
 88711  75481  78174
 88712  73498  78194
 88713  73485  80526
 88714  73924  79612
 88715  76339  78573
 88716  75071  80479
 88717  73485  80091
 88718  74891  79706
 88719  73399  79892
 88720  73171  79198
 88721  75122  77375
```

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  2  2  0  2  2  0  2  2  0  2  0
 40723  1  1  1  2  2  2  2  2  0  2     2  1  1  1  2  2  0  1  1  1  2  0
 40724  0  2  0  2  2  0  0  0  2  0     1  0  2  1  2  1  1  1  1  0  1  1
 40725  0  2  1  2  1  0  0  0  2  0     1  1  2  1  1  1  1  1  1  1  2  1
 40726  0  1  1  1  1  0  1  1  1  1     2  1  1  1  1  2  1  1  1  1  1  1
 40727  1  2  0  2  2  0  1  1  1  1  …  1  0  1  2  0  1  1  0  0  1  2  2
 40728  1  2  0  2  2  0  1  1  1  1     1  1  2  1  1  1  1  1  1  1  2  1
 40729  1  2  0  2  2  0  1  1  1  1     1  1  2  1  1  1  1  1  1  1  2  1
 40730  0  1  1  1  1  0  1  1  1  1     2  0  1  2  2  1  0  0  0  1  2  1
 40731  0  1  1  1  1  0  1  1  1  1     0  0  2  2  0  0  2  0  0  2  2  2
 40732  0  1  1  2  2  0  0  0  2  0  …  1  0  1  2  0  1  1  0  0  1  2  2
 40733  0  2  0  2  2  0  0  0  2  0     2  1  1  1  1  2  0  1  1  0  2  1
 40734  0  2  1  2  1  0  0  0  2  0     2  1  2  1  1  2  0  1  1  1  1  1
     ⋮                 ⋮                 ⋮  ⋱     ⋮              ⋮              ⋮
 88710  0  2  1  2  1  0  0  0  2  0     2  1  1  1  1  2  0  1  1  0  2  1
 88711  0  2  0  2  2  0  1  1  1  1     2  1  2  1  1  2  0  0  0  1  1  2
 88712  0  2  0  2  2  1  1  1  1  1  …  2  2  2  1  2  2  0  1  1  1  2  1
 88713  0  2  0  2  2  0  0  0  2  0     2  0  0  2  0  2  0  0  0  0  2  2
 88714  0  1  1  2  2  1  1  1  1  1     2  2  2  0  2  2  0  2  2  0  2  0
 88715  0  2  0  2  2  0  1  1  1  1     2  1  1  1  1  2  0  1  1  0  2  1
 88716  0  2  0  2  2  0  1  1  2  0     2  2  2  1  2  2  0  1  1  1  2  1
 88717  0  2  0  2  2  0  0  0  2  0  …  2  0  0  2  0  2  0  0  0  0  2  2
 88718  0  1  1  2  2  0  1  1  1  1     1  2  2  2  1  2  0  0  0  2  1  2
 88719  0  2  1  2  1  0  0  0  2  0     2  0  0  2  0  2  0  0  0  0  2  2
 88720  0  1  1  2  2  0  1  1  1  1     2  0  1  2  1  1  1  0  0  1  2  2
 88721  0  2  0  2  2  0  0  0  2  0     2  1  2  2  2  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  1  2  1  0  0  0  2  0  0  2  2  …  2  2  2  0  2  2  0  2  2  0  2  0
 1  1  1  2  2  2  2  2  0  2  2  0  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     1  0  2  1  2  1  1  1  1  0  1  1
 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  1  2     2  1  1  1  1  2  1  1  1  1  1  1
 1  2  0  2  2  0  1  1  1  1  1  1  1  …  1  0  1  2  0  1  1  0  0  1  2  2
 1  2  0  2  2  0  1  1  1  1  1  1  1     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  1  1  1  1  0  1  1  1  1  1  0  1     2  0  1  2  2  1  0  0  0  1  2  1
 0  1  1  1  1  0  1  1  1  1  1  1  2     0  0  2  2  0  0  2  0  0  2  2  2
 0  1  1  2  2  0  0  0  2  0  0  2  2  …  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  1  1  1  1  2  0  1  1  0  2  1
 0  2  1  2  1  0  0  0  2  0  0  1  0     2  1  2  1  1  2  0  1  1  1  1  1
 ⋮              ⋮              ⋮        ⋱        ⋮              ⋮
 0  2  1  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  1  1  1  1  1  0  1     2  1  2  1  1  2  0  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  0  2  2  0  0  0  2  0  0  2  2     2  0  0  2  0  2  0  0  0  0  2  2
 0  1  1  2  2  1  1  1  1  1  1  1  2     2  2  2  0  2  2  0  2  2  0  2  0
 0  2  0  2  2  0  1  1  1  1  1  0  1     2  1  1  1  1  2  0  1  1  0  2  1
 0  2  0  2  2  0  1  1  2  0  0  1  1     2  2  2  1  2  2  0  1  1  1  2  1
 0  2  0  2  2  0  0  0  2  0  0  1  1  …  2  0  0  2  0  2  0  0  0  0  2  2
 0  1  1  2  2  0  1  1  1  1  1  1  1     1  2  2  2  1  2  0  0  0  2  1  2
 0  2  1  2  1  0  0  0  2  0  0  1  0     2  0  0  2  0  2  0  0  0  0  2  2
 0  1  1  2  2  0  1  1  1  1  1  0  0     2  0  1  2  1  1  1  0  0  1  2  2
 0  2  0  2  2  0  0  0  2  0  0  1  1     2  1  2  2  2  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}:
          43785
          42784
          44267
          40737
          41283
          42363
          42886
          44143
          41515
          43182
          43625
          44611
          43132
              ⋮
          73619
          75481
          73498
          73485
          73924
          76339
          75071
          73485
          74891
          73399
          73171
          75122
```

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

Out[53]: 1000-element Array{Int64,1}:
          43785
          42784
          44267
          40737
          41283
          42363
          42886
          44143
          41515
          43182
          43625
          44611
          43132
              ⋮
          72827
          76351
          75239
          76606
          74300
          73127
          74521
          75877
          75310
          75099
          72794
          75531
```

```
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}:
          43785
          42784
          44267
          40737
          41283
          42363
          42886
          44143
          41515
          43182
          43625
          44611
          43132
              ⋮
          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   8.995  10.215
          40723   8.74    9.016
          40724  10.302  10.207
          40725   7.355   9.011
          40726  10.441  10.213
          40727   9.874  10.009
          40728  10.985  10.615
          40729   8.835   9.408
          40730   9.575   9.811
          40731  10.022   8.008
          40732  11.816  10.21
          40733   8.534   9.007
          40734   8.66    9.209
              ⋮
          88710  14.294  11.611
          88711   8.439  10.406
          88712   8.826  10.611
          88713  11.876  11.214
          88714   9.703  11.613
          88715   9.037  11.619
          88716  10.212  12.412
          88717  11.658  12.418
          88718  10.81   12.815
          88719  17.563  12.013
          88720   8.228  11.812
          88721  13.231  10.809
```

```
In [68]: PBVstream = open(PheAll, "w")

Out[68]: IOStream(<file PheAll.txt>)
```

```
In [69]: for i in 1:size(PBV,1)
             @printf(PBVstream, "%19d %10.3f %10.3f \n", PBV[i,1], PBV[i,2], PBV[i,3])
         end
```

```
In [70]: close(PBVstream )
```

Phenotypes - all animnls from G0 to G4

In [71]:
```
OFFG0toG4 = PED[posOFF0S:posOFF4E,1]
```

Out[71]:
```
40000-element Array{Int64,1}:
 40722
 40723
 40724
 40725
 40726
 40727
 40728
 40729
 40730
 40731
 40732
 40733
 40734
     ⋮
 80710
 80711
 80712
 80713
 80714
 80715
 80716
 80717
 80718
 80719
 80720
 80721
```

In [72]:
```
OFFG0toG4ID= DataFrame()
OFFG0toG4ID[:ID] = OFFG0toG4;
```

In [73]:
```
typeof(OFFG0toG4ID)
```

Out[73]:
```
DataFrames.DataFrame
```

In [74]:
```
AllPBV= readtable(pheText,separator=' ',header=false);
```

```
In [75]: rename!(AllPBV,:x1,:ID)
         head(AllPBV);
```

```
In [76]: OFFG0toG4PBV = join(OFFG0toG4ID, AllPBV, on = :ID, kind = :inner);
```

```
In [77]: Row = size(OFFG0toG4PBV,1)

Out[77]: 40000
```

```
In [78]: Col = size(OFFG0toG4PBV,2)

Out[78]: 3
```

```
In [79]: Phestream = open(Phe, "w")

Out[79]: IOStream(<file Phe.txt>)
```

```
In [80]: for i in 1:size(OFFG0toG4PBV,1)
             @printf(Phestream, "%19d %10.3f %10.3f \n", OFFG0toG4PBV[i,1], OFFG0toG4PBV[i,2], OFFG0toG4PBV[i,3])
         end
```

```
In [81]: close(Phestream)
```

Get files with QTL only or Markers only

QTL file and Markers file

```
In [82]: QTLPos = XSim.common.G.qtl_index
```

```
Out[82]: 50-element Array{Int64,1}:
            3
            5
            6
           14
           20
           23
           25
           26
           34
           40
           43
           45
           46
            ⋮
          154
          160
          163
          165
          166
          174
          180
          183
          185
          186
          194
          200
```

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

```
Out[83]: 150-element Array{Int64,1}:
            1
            2
            4
            7
            8
            9
           10
           11
           12
           13
           15
           16
           17
            ⋮
          187
          188
          189
          190
          191
          192
          193
          195
          196
          197
          198
          199
```

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

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

Out[86]:
```
9000x50 Array{Int64,2}:
 0  2  0  1  1  1  1  0  1  1  2  0  1  …  0  1  0  0  2  2  2  2  2  0  2  2
 0  2  0  2  0  0  2  0  0  1  1  1  1     0  1  0  0  2  1  2  2  0  2  2  1
 1  2  1  2  0  0  2  0  1  1  0  1  1     0  1  0  0  1  2  2  2  1  1  2  1
 0  2  0  1  2  1  2  0  1  1  1  0  0     2  0  0  0  2  2  1  2  0  2  2  0
 2  0  0  1  1  2  1  0  2  2  1  0  0     0  1  2  1  1  2  1  2  1  1  2  0
 0  2  0  1  2  2  1  0  2  2  0  0  0  …  0  1  0  0  1  2  2  2  1  1  2  1
 1  2  0  2  0  2  2  0  1  2  1  1  1     1  1  1  1  1  2  2  1  1  1  1  1
 1  1  0  2  2  2  1  0  1  1  1  0  0     2  0  1  0  1  2  2  2  1  1  2  0
 1  1  0  2  0  1  1  0  1  1  0  2  2     1  0  1  0  0  2  2  2  0  2  2  2
 1  2  0  2  0  1  1  0  1  2  1  1  1     0  0  0  0  1  2  1  2  2  0  2  2
 0  2  1  1  1  2  1  1  1  1  0  2  1  …  1  1  0  1  0  2  2  2  0  2  2  0
 1  1  0  1  1  1  2  0  1  2  1  2  1     1  0  1  0  1  2  1  2  1  1  2  2
 0  2  0  0  1  1  0  1  2  1  2  0  0     0  0  2  1  0  2  2  2  1  1  2  0
 ⋮              ⋮              ⋮        ⋱        ⋮              ⋮
 1  1  0  1  2  1  1  0  2  2  1  1  1     1  2  2  1  1  2  1  2  1  1  2  1
 0  2  0  2  2  2  0  0  1  2  1  1  1     0  0  0  1  0  2  2  2  1  1  2  2
 0  2  1  1  2  2  0  0  2  2  0  1  1  …  0  0  0  0  2  2  1  2  1  2  2  1
 0  2  0  0  2  2  1  1  2  1  1  1  2     0  0  1  0  2  2  1  2  2  0  2  2
 1  2  1  0  2  1  2  1  2  0  1  1  1     1  1  0  1  0  2  1  2  0  2  2  0
 0  2  0  2  1  1  2  0  0  0  0  1  2     0  2  1  1  1  2  1  2  0  2  2  1
 0  2  0  1  1  2  0  0  2  2  1  2  2     1  1  1  1  1  2  2  2  1  2  2  1
 0  2  0  1  1  2  2  1  2  1  1  1  1  …  0  0  0  0  2  1  2  2  2  0  2  2
 1  2  0  1  2  2  0  0  1  2  0  2  2     1  1  1  0  1  2  1  2  1  2  2  2
 1  1  0  2  2  2  1  0  2  2  2  0  1     2  1  0  0  2  2  0  2  2  0  2  2
 1  2  0  2  1  2  1  0  2  2  0  2  2     0  1  1  1  0  2  2  2  1  1  1  2
 0  2  0  1  1  2  0  0  1  2  1  2  1     0  1  0  1  1  2  1  2  1  1  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]:
```
0.14985558167971724
```

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

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

```
Out[100]: 0.3917984852559423
```

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

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

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

```
Out[102]: 50-element Array{Float64,1}:
           0.200674
           0.199938
           0.201326
           0.199743
           0.199959
           0.203195
           0.200113
           0.200338
           0.199502
           0.203101
           0.195585
           0.201661
           0.1996
           ⋮
           0.199112
           0.200839
           0.201562
           0.200313
           0.203918
           0.199916
           0.199452
           0.202067
           0.201115
           0.199098
           0.201983
           0.203554
```

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

```
Out[103]: 48000-element Array{Float64,1}:
           10.2252
            9.0277
           10.2187
            9.0222
           10.2088
           10.0294
           10.6237
            9.4019
            9.82423
            8.02062
           10.2143
            9.02155
            9.23587
            ⋮
           11.6299
           10.4167
           10.6266
           11.2182
           11.6078
           11.6196
           12.4155
           12.4338
           12.8386
           12.0306
           11.8371
           10.8356
```

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

```
Out[104]: 9.87843932800409
```

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

```
Out[105]: 10.343476141121615
```

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

```
Out[106]: 10.596362788898881
```

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

Out[107]: 10.842002168681029

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

Out[108]: 11.073881230680657

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

Out[109]: 11.321978700148117

In [110]: EAlphaG=onlyQTL*QTLo

Out[110]: 9000-element Array{Float64,1}:
           10.2223
           10.2114
           10.2324
           10.4067
           11.0208
           11.227
           11.4297
           11.213
           11.0192
           10.8304
           10.6209
           11.2203
            9.4115
            ⋮
           11.6299
           10.4167
           10.6266
           11.2182
           11.6078
           11.6196
           12.4155
           12.4338
           12.8386
           12.0306
           11.8371
           10.8356
```

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

Out[111]: 11.300962557757117

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

Out[112]: 1.4225232297530273

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

Out[113]: 10.83891941637456

In [114]: meanBreedingValueSelectedG0=meanEAlphaS0-meanEAlphaG0

Out[114]: 0.9604800883704705

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

Out[115]: 10.861917043114675

In [116]: meanBreedingValueSelectedG1=meanEAlphaS1-meanEAlphaG0

Out[116]: 0.9834777151105847

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

Out[117]: 11.083686417203879

In [118]: meanBreedingValueSelectedG2=meanEAlphaS2-meanEAlphaG0

Out[118]: 1.2052470891997888

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

Out[119]: 11.323252225577065

In [120]: meanBreedingValueSelectedG3=meanEAlphaS3-meanEAlphaG0

Out[120]: 1.4448128975729748
```

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

Out[121]: 11.556391990875444

In [122]: meanBreedingValueSelectedG4=meanEAlphaS4-meanEAlphaG0

Out[122]: 1.6779526628713537

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

Out[123]: 11.321978700148117

In [124]: meanBreedingValueSelectedG5=meanEAlphaS5-meanEAlphaG0

Out[124]: 1.443539372144027
```