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 = 0.3
# 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

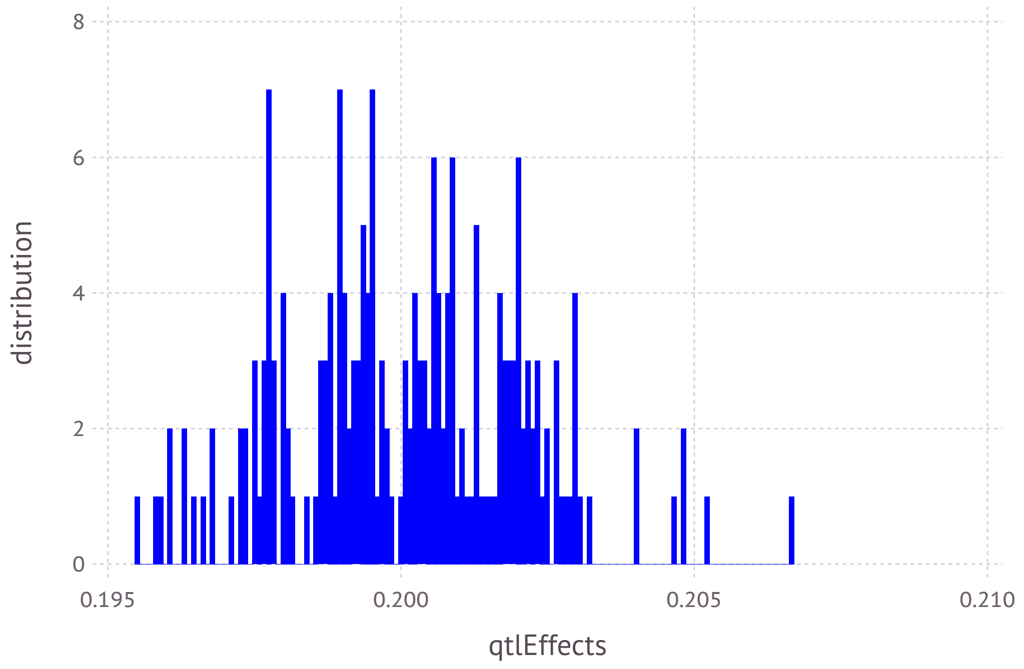
In [6]:
```
numChr    = 10
chrLength = 0.01
numLoci   = 20
nQTL      = 5
mutRate   = 0.0
locusInt  = chrLength/numLoci
mapPos    = collect(locusInt/2:locusInt:chrLength)
geneFreq  = fill(0.5,numLoci)
QTL = sample(1:numLoci,nQTL,replace=false)
qtlMarker = fill(false,numLoci)
qtlMarker[QTL] = true
mu = 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 = 0.2
XSim.init(numChr,numLoci,chrLength,geneFreq,mapPos,qtlMarker,qtlEffects,mutRate)
```

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

```
Out[7]: 200-element Array{Float64,1}:
         0.200838
         0.200426
         0.200536
         0.197814
         0.199337
         0.197259
         0.202763
         0.197982
         0.199015
         0.200604
         0.198611
         0.197361
         0.197553
         ⋮
         0.196026
         0.198824
         0.200201
         0.200591
         0.200417
         0.199355
         0.200864
         0.199645
         0.202104
         0.199852
         0.197651
         0.201702
```

```
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]: 0.20009081140132415
```

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

Out[11]: 4.010304539952925e-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
```

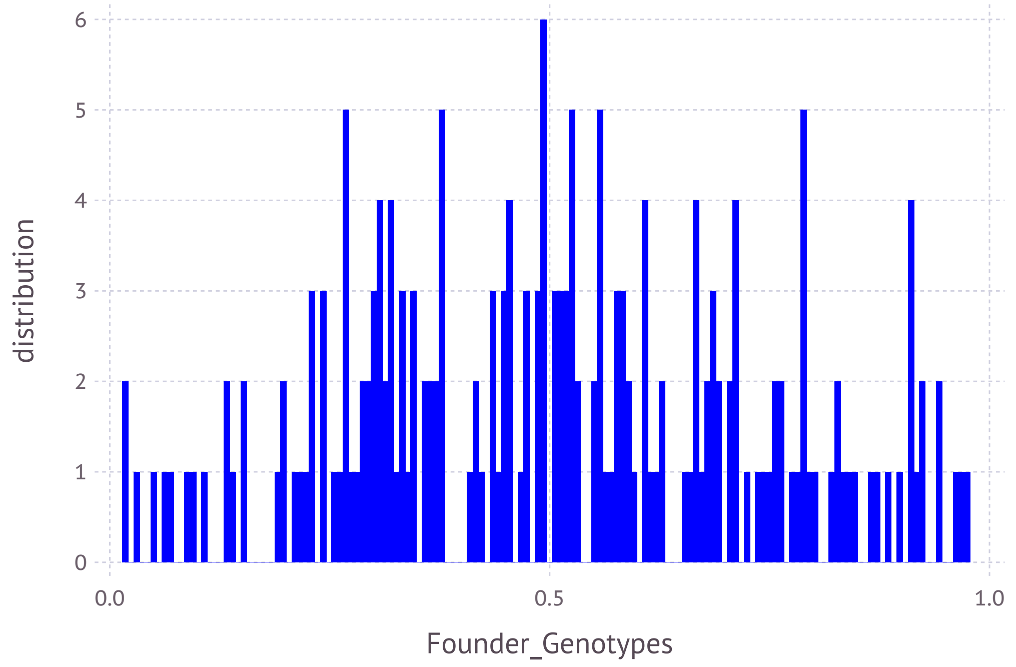
```
In [19]: gSPSire = XSim.getOurGenotypes(popSP[1])
         gSPDam = XSim.getOurGenotypes(popSP[2])
         gSP = [gSPSire;gSPSire];
```

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

```
Out[20]: 1x200 Array{Float64,2}:
          0.071375  0.827375  0.292375  …  0.3755  0.370125  0.91425  0.548375
```

```
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);
```

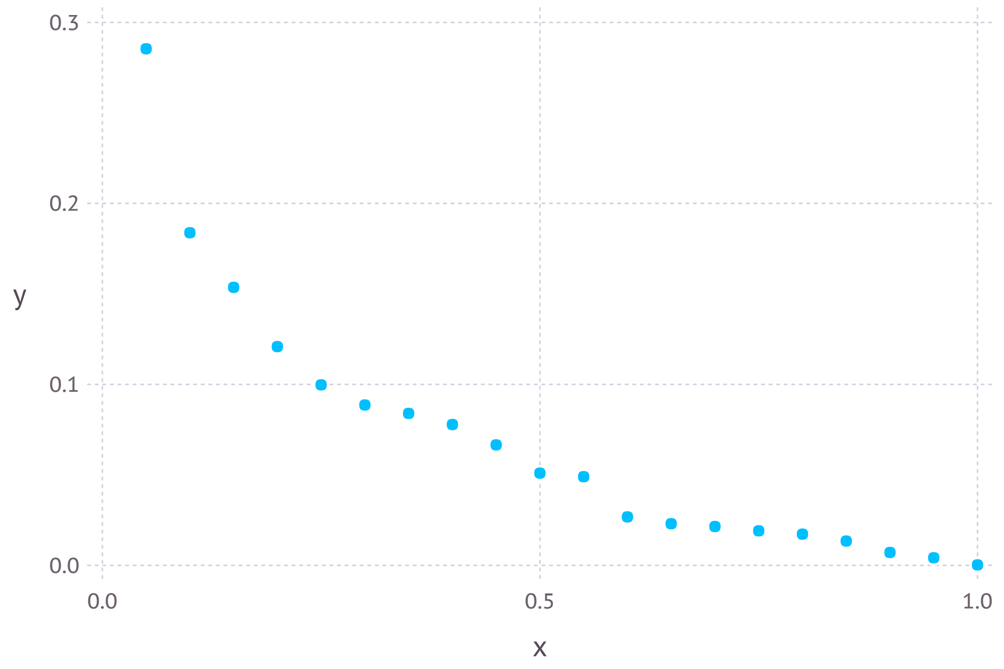
In [23]:
```
for i=1:(nRows-20)
LDMat[i,:] = corMat[i,(i+1):(i+20)].^2
end
```

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

Out[24]:
```
1x20 Array{Float64,2}:
 0.000289942  0.00421348  0.00713602  …  0.153635  0.18378  0.285434
```

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]:
```
11.133235142993678
```

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

Out[31]:
```
0.6474642543081246
```

In [32]:
```
XSim.common.varRes = (7*varGen)/3    #heritability = 0.3
```

Out[32]:
```
1.5107499267189575
```

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

Out[33]:
```
1.5107499267189575
```

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]: 13.22121030603552

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

Out[36]: 13.21232516523916

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

Out[37]: 0.5237363525074606

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

Out[38]: 0.518534392300554
```

Pedigree: All animals

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

```
Out[39]: 48000x3 Array{Int64,2}:
          40722  35931  37866
          40723  36682  40373
          40724  36325  39782
          40725  35228  39771
          40726  34447  38859
          40727  34740  36817
          40728  36687  38593
          40729  36258  38595
          40730  35501  38678
          40731  34049  38910
          40732  32910  37375
          40733  36062  38796
          40734  36054  40637
              ⋮
          88710  76061  78422
          88711  76156  79029
          88712  73574  77626
          88713  72737  79782
          88714  75492  79063
          88715  74972  80104
          88716  76476  79100
          88717  75458  80709
          88718  74571  79982
          88719  74298  78800
          88720  73817  80569
          88721  74384  80031
```

```
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  1  1  1  2  0  1  1  0  2  0
          40723  0  2  0  2  2  0  0  0  2  0     1  0  1  2  1  1  1  0  0  1  2  2
          40724  0  2  1  2  1  0  0  0  2  0     2  1  2  1  2  1  0  1  1  0  2  1
          40725  0  2  0  2  2  0  1  1  2  0     0  0  2  1  1  1  1  1  1  1  2  1
          40726  0  2  1  2  1  1  1  1  1  1     1  0  1  2  0  1  1  0  0  1  2  2
          40727  0  1  2  2  1  0  1  1  1  1  …  1  2  2  1  1  2  0  2  2  0  2  1
          40728  0  2  0  2  2  0  0  0  2  0     1  0  1  2  0  1  1  0  0  1  2  2
          40729  1  2  1  2  1  1  1  1  1  1     2  1  2  1  2  2  0  1  1  0  2  1
          40730  0  2  0  2  2  0  0  0  2  0     2  1  2  1  1  1  0  1  1  0  2  1
          40731  1  2  0  2  2  0  1  1  1  1     2  1  1  1  1  2  0  1  1  0  2  1
          40732  0  2  0  2  2  1  1  1  1  1  …  2  2  1  1  2  2  0  1  1  0  2  1
          40733  1  1  1  2  2  0  2  2  0  2     2  1  1  1  2  2  0  1  1  1  2  0
          40734  0  2  0  2  2  0  1  1  1  1     2  2  1  1  2  2  0  1  1  0  2  1
              ⋮              ⋮              ⋮  ⋱     ⋮              ⋮              ⋮
          88710  0  2  1  2  1  0  0  0  2  0     2  1  1  1  2  2  0  1  1  1  2  1
          88711  0  2  1  2  1  0  0  0  2  0     1  0  1  2  0  0  2  0  0  2  2  2
          88712  0  2  0  2  2  0  0  0  2  0  …  1  0  2  2  1  1  1  0  0  2  1  2
          88713  0  2  1  2  1  1  1  1  1  1     2  1  1  1  1  2  0  1  1  0  2  1
          88714  0  2  0  2  2  0  1  1  2  0     2  0  0  2  2  1  0  0  0  1  1  2
          88715  0  2  1  2  1  0  0  0  2  0     2  0  0  2  1  2  0  0  0  1  2  1
          88716  0  2  1  2  1  0  0  0  2  0     1  1  2  2  0  0  2  0  0  2  2  2
          88717  0  2  0  2  2  0  0  0  2  0  …  2  0  0  2  1  2  0  0  0  1  2  1
          88718  0  2  2  2  0  0  0  0  2  0     2  0  1  2  2  2  0  0  0  2  0  1
          88719  1  2  0  2  2  0  1  1  1  1     2  1  1  1  2  2  0  1  1  1  1  1
          88720  0  2  2  2  0  0  0  0  2  0     2  1  1  1  1  2  1  1  1  1  1  1
          88721  0  1  1  2  2  0  1  1  1  1     1  1  1  2  0  0  2  0  0  2  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  1  …  2  2  1  1  1  2  0  1  1  0  2  0
 0  2  0  2  2  0  0  0  2  0  0  2  2     1  0  1  2  1  1  1  0  0  1  2  2
 0  2  1  2  1  0  0  0  2  0  0  1  0     2  1  2  1  2  1  0  1  1  0  2  1
 0  2  0  2  2  0  1  1  2  0  0  1  1     0  0  2  1  1  1  1  1  1  1  2  1
 0  2  1  2  1  1  1  1  1  1  1  1  1     1  0  1  2  0  1  1  0  0  1  2  2
 0  1  2  2  1  0  1  1  1  1  1  1  0  …  1  2  2  1  1  2  0  2  2  0  2  1
 0  2  0  2  2  0  0  0  2  0  0  2  2     1  0  1  2  0  1  1  0  0  1  2  2
 1  2  1  2  1  1  1  1  1  1  1  1  1     2  1  2  1  2  2  0  1  1  0  2  1
 0  2  0  2  2  0  0  0  2  0  0  1  1     2  1  2  1  1  1  0  1  1  0  2  1
 1  2  0  2  2  0  1  1  1  1  1  0  0     2  1  1  1  1  2  0  1  1  0  2  1
 0  2  0  2  2  1  1  1  1  1  1  1  2  …  2  2  1  1  2  2  0  1  1  0  2  1
 1  1  1  2  2  0  2  2  0  2  2  0  0     2  1  1  1  2  2  0  1  1  1  2  0
 0  2  0  2  2  0  1  1  1  1  1  0  1     2  2  1  1  2  2  0  1  1  0  2  1
 ⋮              ⋮              ⋮        ⋱        ⋮              ⋮
 0  2  1  2  1  0  0  0  2  0  0  2  1     2  1  1  1  2  2  0  1  1  1  2  1
 0  2  1  2  1  0  0  0  2  0  0  2  1     1  0  1  2  0  0  2  0  0  2  2  2
 0  2  0  2  2  0  0  0  2  0  0  2  2  …  1  0  2  2  1  1  1  0  0  2  1  2
 0  2  1  2  1  1  1  1  1  1  1  1  1     2  1  1  1  1  2  0  1  1  0  2  1
 0  2  0  2  2  0  1  1  2  0  0  2  2     2  0  0  2  2  1  0  0  0  1  1  2
 0  2  1  2  1  0  0  0  2  0  0  1  0     2  0  0  2  1  2  0  0  0  1  2  1
 0  2  1  2  1  0  0  0  2  0  0  2  1     1  1  2  2  0  0  2  0  0  2  2  2
 0  2  0  2  2  0  0  0  2  0  0  1  1  …  2  0  0  2  1  2  0  0  0  1  2  1
 0  2  2  2  0  0  0  0  2  0  0  2  0     2  0  1  2  2  2  0  0  0  2  0  1
 1  2  0  2  2  0  1  1  1  1  1  0  0     2  1  1  1  2  2  0  1  1  1  1  1
 0  2  2  2  0  0  0  0  2  0  0  2  0     2  1  1  1  1  2  1  1  1  1  1  1
 0  1  1  2  2  0  1  1  1  1  1  1  1     1  1  1  2  0  0  2  0  0  2  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}:
          44282
          41305
          42739
          41658
          42902
          42155
          41285
          42071
          43308
          44096
          43738
          41361
          42126
              ⋮
          76061
          76156
          73574
          72737
          75492
          74972
          76476
          75458
          74571
          74298
          73817
          74384
```

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

```
Out[53]: 1000-element Array{Int64,1}:
          44282
          41305
          42739
          41658
          42902
          42155
          41285
          42071
          43308
          44096
          43738
          41361
          42126
              ⋮
          74902
          74366
          76673
          75209
          74440
          76517
          74394
          75251
          73574
          74974
          75284
          75263
```

```
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}:
          44282
          41305
          42739
          41658
          42902
          42155
          41285
          42071
          43308
          44096
          43738
          41361
          42126
              ⋮
          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   9.881  10.765
 40723  13.735  12.35 
 40724  11.718  10.765
 40725  11.882  10.175
 40726  11.934  11.562
 40727   8.384  10.969
 40728  11.858  11.575
 40729  13.519  12.367
 40730   8.446   9.773
 40731  10.512  10.57 
 40732  10.708  10.763
 40733  10.382  10.373
 40734  10.142  11.358
     ⋮                
 88710  13.129  12.969
 88711  11.543  12.963
 88712  15.761  14.371
 88713  12.379  12.17 
 88714  15.33   14.17 
 88715  13.054  12.78 
 88716  14.517  13.955
 88717  11.528  14.165
 88718  15.602  14.364
 88719  13.199  12.979
 88720  15.48   13.975
 88721  12.718  13.76 
```

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}:
            12
            14
            18
            19
            20
            32
            34
            38
            39
            40
            52
            54
            58
             ⋮
           159
           160
           172
           174
           178
           179
           180
           192
           194
           198
           199
           200
```

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

```
Out[83]: 150-element Array{Int64,1}:
            1
            2
            3
            4
            5
            6
            7
            8
            9
           10
           11
           13
           15
            ⋮
          184
          185
          186
          187
          188
          189
          190
          191
          193
          195
          196
          197
```

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

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

```
Out[86]: 9000x50 Array{Int64,2}:
          2  0  2  0  2  1  1  1  0  2  2  0  1  …  1  1  2  2  1  1  2  2  1  1  2  2
          1  1  1  0  2  1  1  2  0  2  1  1  1     1  1  1  2  1  1  1  1  2  0  2  1
          2  1  1  0  2  1  1  1  1  1  1  1  1     2  0  2  2  0  2  2  1  2  1  2  2
          2  0  1  0  1  0  0  2  0  2  1  2  0     2  0  1  2  1  1  1  2  1  1  2  1
          2  1  1  0  2  2  2  2  0  2  0  0  1     2  0  2  2  1  1  2  2  0  2  2  2
          1  1  0  0  2  2  2  1  0  2  1  1  1  …  0  2  0  1  0  1  1  2  0  2  2  2
          2  0  2  0  2  2  2  2  0  2  2  1  1     2  0  2  2  0  2  2  1  2  0  2  1
          1  1  1  0  1  1  1  1  1  1  1  1  1     2  0  1  2  2  0  1  1  2  0  2  1
          2  0  2  0  2  2  2  1  0  2  1  1  1     1  1  2  2  1  1  1  0  2  0  2  0
          1  1  1  0  2  2  2  2  0  2  1  1  1     1  1  2  1  1  1  2  2  1  1  2  2
          2  1  1  0  2  1  2  2  1  1  1  1  0  …  1  1  2  2  2  0  2  2  2  1  2  0
          2  1  2  0  2  1  1  2  1  1  1  1  1     2  0  2  2  0  1  2  2  2  1  1  1
          1  1  1  0  0  2  2  1  1  1  2  0  0     2  0  2  2  1  1  2  0  2  0  2  1
          ⋮              ⋮              ⋮           ⋱        ⋮              ⋮
          2  1  1  0  2  1  1  2  1  1  2  0  2     1  1  2  2  1  1  1  1  2  1  2  1
          2  1  1  0  2  2  2  2  0  2  1  1  0     0  2  2  2  1  1  2  2  0  2  2  2
          2  0  2  0  1  2  2  1  0  2  2  2  0  …  0  2  2  2  2  0  2  2  1  2  1  2
          1  2  1  0  1  1  1  2  2  1  1  0  2     1  1  2  2  2  0  2  1  2  0  2  1
          2  0  2  0  2  2  2  2  0  2  0  2  0     1  1  2  2  2  0  2  2  1  1  1  2
          1  2  0  0  1  0  1  2  0  2  1  1  1     2  0  2  2  2  0  2  2  2  1  2  1
          2  1  2  0  2  2  2  2  0  2  2  0  2     2  0  2  2  1  1  1  2  0  2  2  2
          1  1  1  0  1  2  2  1  1  1  0  0  1  …  2  0  2  2  0  2  2  2  2  1  2  1
          2  2  1  0  2  2  2  2  0  2  2  2  0     1  1  2  2  2  0  2  2  2  2  0  1
          0  2  0  1  0  2  2  2  0  2  1  2  0     2  0  2  2  2  0  2  1  2  1  1  1
          2  2  2  0  2  1  1  1  1  2  2  0  2     1  1  2  2  2  0  2  1  2  1  1  1
          1  1  2  1  0  2  2  1  0  2  2  2  0     1  1  2  2  2  0  2  2  0  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]:
```
0.43025693745240406
```

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

          WARNING: imported binding for cor overwritten in module Main

Out[100]: 0.6506087893319158

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

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

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

```
Out[102]: 50-element Array{Float64,1}:
           0.197361
           0.201882
           0.201227
           0.197731
           0.199295
           0.204016
           0.202133
           0.199071
           0.200268
           0.199173
           0.195886
           0.20009
           0.200453
           ⋮
           0.200633
           0.200663
           0.19864
           0.197544
           0.200626
           0.20023
           0.200101
           0.200591
           0.199355
           0.199852
           0.197651
           0.201702
```

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

Out[103]:
```
48000-element Array{Float64,1}:
 10.7891
 12.3987
 10.7884
 10.2049
 11.6106
 11.0113
 11.6022
 12.402
  9.80758
 10.6022
 10.8157
 10.4126
 11.4028
  ⋮
 13.0049
 13.0035
 14.4096
 12.2073
 14.2098
 12.8005
 13.9911
 14.208
 14.3994
 13.0098
 13.9984
 13.8075
```

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

Out[104]:
```
11.16739700615123
```

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

Out[105]:
```
11.679826207236447
```

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

Out[106]:
```
12.112739495069688
```

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

Out[107]: 12.509030523253967

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

Out[108]: 12.90109802575528

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

Out[109]: 13.252522222262506

In [110]: EAlphaG=onlyQTL*QTLo

Out[110]: 9000-element Array{Float64,1}:
           12.3987
           12.402
           12.9976
           11.9881
           13.0011
           11.2135
           13.3983
           11.8096
           12.5999
           12.6039
           12.2013
           12.6036
           10.5975
            ⋮
           13.0049
           13.0035
           14.4096
           12.2073
           14.2098
           12.8005
           13.9911
           14.208
           14.3994
           13.0098
           13.9984
           13.8075
```

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

Out[111]: 13.214394572723121

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

Out[112]: 2.046997566571891

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

Out[113]: 12.187682967089714

In [114]: meanBreedingValueSelectedG0=meanEAlphaS0-meanEAlphaG0

Out[114]: 1.0202859609384838

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

Out[115]: 12.531431876401811

In [116]: meanBreedingValueSelectedG1=meanEAlphaS1-meanEAlphaG0

Out[116]: 1.3640348702505811

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

Out[117]: 12.925562405846865

In [118]: meanBreedingValueSelectedG2=meanEAlphaS2-meanEAlphaG0

Out[118]: 1.7581653996956348

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

Out[119]: 13.2842916559892

In [120]: meanBreedingValueSelectedG3=meanEAlphaS3-meanEAlphaG0

Out[120]: 2.1168946498379704
```

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

Out[121]: 13.61789797671257

In [122]: meanBreedingValueSelectedG4=meanEAlphaS4-meanEAlphaG0

Out[122]: 2.4505009705613396

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

Out[123]: 13.252522222262506

In [124]: meanBreedingValueSelectedG5=meanEAlphaS5-meanEAlphaG0

Out[124]: 2.085125216111276
```