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)
```

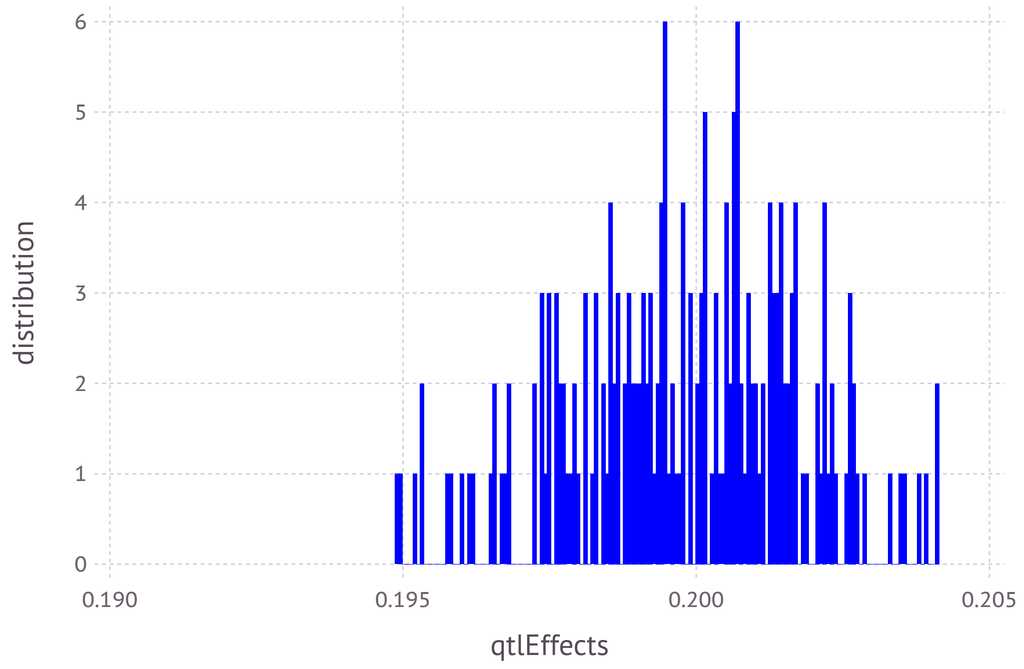
```
In [7]: qtlEffects
```

```
Out[7]: 200-element Array{Float64,1}:
         0.202076
         0.201334
         0.194863
         0.19877
         0.197508
         0.202082
         0.196794
         0.19914
         0.200507
         0.197758
         0.199447
         0.199195
         0.201827
         ⋮
         0.198571
         0.200627
         0.201279
         0.201697
         0.19912
         0.199618
         0.195316
         0.19652
         0.200644
         0.19883
         0.199482
         0.199374
```

```
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]:
```
0.1998092003576393
```

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

Out[11]:
```
3.787094340026868e-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
```

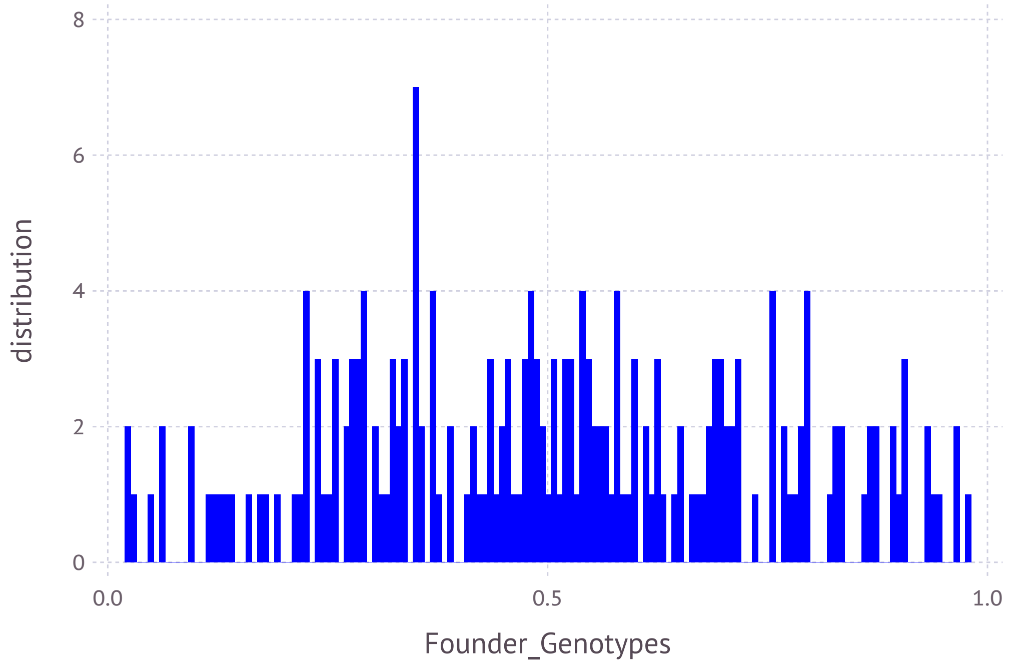
```
In [19]: gSPSire = XSim.getOurGenotypes(popSP[1])
         gSPDam = XSim.getOurGenotypes(popSP[2])
         gSP = [gSPSire;gSPSire];
```

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

Out[20]: 1x200 Array{Float64,2}:
          0.062  0.836375  0.275375  0.947875  …  0.37325  0.372125  0.9015  0.5465
```

```
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);
```

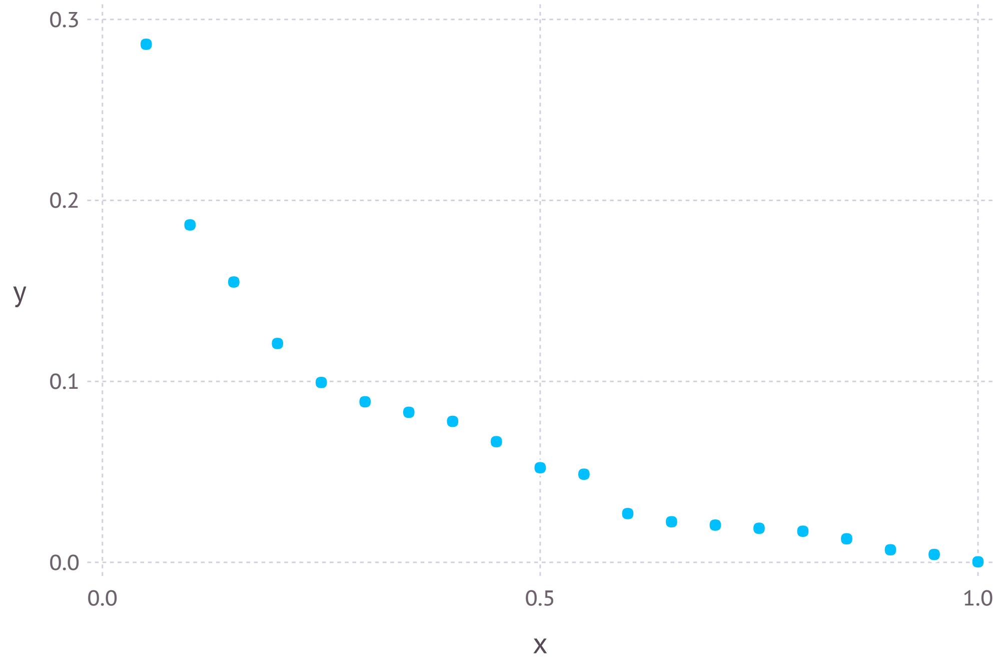
```
In [23]: for i=1:(nRows-20)
         LDMat[i,:] = corMat[i,(i+1):(i+20)].^2
         end
```

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

```
Out[24]: 1x20 Array{Float64,2}:
          0.000308811  0.00436592  0.00695608  …  0.154905  0.186478  0.286268
```

```
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]:
```
10.65073436277774
```

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

Out[31]:
```
0.7056466733176778
```

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

Out[32]:
```
6.3508200598591005
```

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

Out[33]:
```
6.3508200598591005
```

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]: 12.150636522531496

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

Out[36]: 12.119438716505787

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

Out[37]: 0.577337467453749

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

Out[38]: 0.5752298718064962
```

Pedigree: All animals

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

Out[39]: 48000x3 Array{Int64,2}:
          40722  34395  39303
          40723  36710  40191
          40724  32874  40489
          40725  33388  36755
          40726  35781  38161
          40727  34044  38805
          40728  34857  39342
          40729  34291  40589
          40730  34043  38843
          40731  35950  36729
          40732  33308  36886
          40733  35986  40627
          40734  36620  37111
              ⋮
          88710  73738  77394
          88711  75351  78876
          88712  76280  77090
          88713  73258  78522
          88714  74166  78168
          88715  76643  79137
          88716  74640  79208
          88717  75652  78351
          88718  76517  77860
          88719  75319  80640
          88720  76373  79559
          88721  73775  78150
```

```
In [40]: PEDstream = open(PedAll, "w")

Out[40]: IOStream(<file PedAll.txt>)
```

```
In [41]: for i in 1:size(PED,1)
             @printf(PEDstream, "%19d%19d%19d \n", PED[i,1], PED[i,2], PED[i,3])
         end
```

```
In [42]: close(PEDstream)
```

Create matrix of ``genotype" covariates for all animals

```
In [43]: nObs = countlines(pedText)

Out[43]: 48000
```

```
In [44]: nMarker = numChr*numLoci

Out[44]: 200
```

```
In [45]: GT = convert(Array,readtable(genText,separator=' ',header=false))

Out[45]: 48000x201 Array{Int64,2}:
          40722  0  2  0  2  2  0  0  0  2  0  …  1  1  2  1  1  1  1  1  1  1  2  1
          40723  0  2  0  2  2  1  1  1  1  1     1  2  2  1  2  2  1  2  2  0  2  1
          40724  0  1  2  2  1  0  0  0  2  0     2  1  2  1  1  2  0  1  1  0  1  1
          40725  0  2  0  2  2  2  2  2  0  2     1  0  1  2  1  1  1  0  0  2  2  1
          40726  0  2  0  2  2  0  0  0  2  0     2  1  1  1  1  2  0  0  0  1  2  2
          40727  0  1  1  2  2  0  1  1  1  1  …  2  1  2  1  1  1  0  1  1  0  2  1
          40728  1  2  0  2  2  1  1  1  1  1     2  1  1  1  2  2  0  1  1  1  2  0
          40729  0  2  0  2  2  1  1  1  1  1     2  1  2  1  2  2  0  1  1  0  2  1
          40730  0  1  1  1  1  0  1  1  1  1     2  1  1  1  2  2  0  1  1  1  2  0
          40731  0  2  0  2  2  0  0  0  2  0     2  1  1  2  0  2  0  0  0  0  1  2
          40732  0  1  1  1  1  0  1  1  1  1  …  1  1  2  2  1  1  1  0  0  2  2  1
          40733  0  1  1  2  1  2  2  2  0  2     1  0  1  2  1  1  1  0  0  2  2  1
          40734  0  2  1  2  1  0  0  0  2  0     1  1  2  1  1  1  1  1  1  1  2  1
              ⋮              ⋮              ⋮  ⋱     ⋮              ⋮              ⋮
          88710  0  2  0  2  2  0  1  1  1  1     2  1  2  2  1  2  0  0  0  1  2  1
          88711  0  1  1  1  1  0  1  1  1  1     1  0  2  2  1  1  1  0  0  1  2  2
          88712  1  1  1  2  2  1  1  1  1  1  …  2  2  2  1  2  2  0  1  1  1  2  1
          88713  1  2  0  2  2  1  1  1  2  0     2  1  1  1  2  2  0  1  1  1  2  0
          88714  1  2  1  2  1  0  0  0  2  0     2  1  1  1  1  2  0  1  1  0  2  1
          88715  1  2  0  2  2  1  1  1  1  1     1  1  2  1  1  1  1  1  1  1  2  1
          88716  0  0  2  1  1  0  2  2  0  2     2  1  2  1  2  2  0  1  1  0  2  1
          88717  0  0  2  2  2  0  1  1  1  1  …  2  2  2  0  2  2  0  2  2  0  2  0
          88718  1  2  1  2  1  2  2  2  1  1     1  0  2  1  0  2  1  1  1  1  1  1
          88719  1  2  0  2  2  0  1  1  1  1     2  1  2  1  2  1  1  1  1  0  1  1
          88720  0  2  0  2  2  0  0  0  2  0     1  0  2  2  1  1  1  0  0  2  1  2
          88721  0  2  0  2  2  1  1  1  1  1     2  1  1  1  1  2  0  1  1  0  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  0  2  2  0  0  0  2  0  0  2  2  …  1  1  2  1  1  1  1  1  1  1  2  1
          0  2  0  2  2  1  1  1  1  1  1  1  2     1  2  2  1  2  2  1  2  2  0  2  1
          0  1  2  2  1  0  0  0  2  0  0  2  1     2  1  2  1  1  2  0  1  1  0  1  1
          0  2  0  2  2  2  2  2  0  2  2  0  2     1  0  1  2  1  1  1  0  0  2  2  1
          0  2  0  2  2  0  0  0  2  0  0  2  2     2  1  1  1  1  2  0  0  0  1  2  2
          0  1  1  2  2  0  1  1  1  1  1  1  2  …  2  1  2  1  1  1  0  1  1  0  2  1
          1  2  0  2  2  1  1  1  1  1  1  1  2     2  1  1  1  2  2  0  1  1  1  2  0
          0  2  0  2  2  1  1  1  1  1  1  0  1     2  1  2  1  2  2  0  1  1  0  2  1
          0  1  1  1  1  0  1  1  1  1  1  0  1     2  1  1  1  2  2  0  1  1  1  2  0
          0  2  0  2  2  0  0  0  2  0  0  1  1     2  1  1  2  0  2  0  0  0  0  1  2
          0  1  1  1  1  0  1  1  1  1  1  1  2  …  1  1  2  2  1  1  1  0  0  2  2  1
          0  1  1  2  1  2  2  2  0  2  2  0  1     1  0  1  2  1  1  1  0  0  2  2  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  2  0  2  2  0  1  1  1  1  1  1  2     2  1  2  2  1  2  0  0  0  1  2  1
          0  1  1  1  1  0  1  1  1  1  1  1  2     1  0  2  2  1  1  1  0  0  1  2  2
          1  1  1  2  2  1  1  1  1  1  1  1  2  …  2  2  2  1  2  2  0  1  1  1  2  1
          1  2  0  2  2  1  1  1  2  0  0  1  0     2  1  1  1  2  2  0  1  1  1  2  0
          1  2  1  2  1  0  0  0  2  0  0  2  1     2  1  1  1  1  2  0  1  1  0  2  1
          1  2  0  2  2  1  1  1  1  1  1  0  1     1  1  2  1  1  1  1  1  1  1  2  1
          0  0  2  1  1  0  2  2  0  2  2  0  1     2  1  2  1  2  2  0  1  1  0  2  1
          0  0  2  2  2  0  1  1  1  1  1  1  2  …  2  2  2  0  2  2  0  2  2  0  2  0
          1  2  1  2  1  2  2  2  1  1  1  1  1     1  0  2  1  0  2  1  1  1  1  1  1
          1  2  0  2  2  0  1  1  1  1  1  0  0     2  1  2  1  2  1  1  1  1  0  1  1
          0  2  0  2  2  0  0  0  2  0  0  0  0     1  0  2  2  1  1  1  0  0  2  1  2
          0  2  0  2  2  1  1  1  1  1  1  1  2     2  1  1  1  1  2  0  1  1  0  2  1
```

```
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}:
          43825
          43098
          43952
          42123
          43306
          42461
          44102
          44531
          41179
          41892
          41474
          42571
          41851
              ⋮
          73738
          75351
          76280
          73258
          74166
          76643
          74640
          75652
          76517
          75319
          76373
          73775
```

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

```
Out[53]: 1000-element Array{Int64,1}:
          43825
          43098
          43952
          42123
          43306
          42461
          44102
          44531
          41179
          41892
          41474
          42571
          41851
              ⋮
          75317
          75610
          74048
          75352
          74438
          75935
          73511
          74374
          76474
          73647
          75698
          74640
```

```
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}:
          43825
          43098
          43952
          42123
          43306
          42461
          44102
          44531
          41179
          41892
          41474
          42571
          41851
              ⋮
          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   9.504   9.948
          40723   9.208   9.361
          40724  11.288  11.131
          40725  11.272  10.947
          40726  10.162   9.952
          40727  11.692  12.336
          40728  10.93   10.937
          40729  11.5    11.749
          40730  12.987  11.329
          40731   9.272  10.339
          40732  10.416  11.147
          40733   9.982   9.734
          40734  10.884  10.957
               ⋮
          88710  11.81   11.933
          88711   9.332  12.134
          88712  14.797  11.526
          88713  10.692  11.724
          88714  15.141  12.525
          88715  13.932  12.548
          88716  11.244  12.344
          88717  14.003  14.104
          88718   8.789  10.94
          88719  16.541  13.927
          88720  14.414  12.933
          88721  15.606  12.513
```

```
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}:
            1
            3
           11
           12
           17
           21
           23
           31
           32
           37
           41
           43
           51
            ⋮
          152
          157
          161
          163
          171
          172
          177
          181
          183
          191
          192
          197
```

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

Out[83]:
```
150-element Array{Int64,1}:
   2
   4
   5
   6
   7
   8
   9
  10
  13
  14
  15
  16
  18
   ⋮
 186
 187
 188
 189
 190
 193
 194
 195
 196
 198
 199
 200
```

Genotype codes: 0, 1, 2

In [84]:
```
IDgen = convert(Array,readtable(Gen,separator=' ',header=false));
```

In [85]:
```
onlyID = IDgen[:,1]
QTLMarker = IDgen[:, 2:end]
```

Out[85]:
```
9000x200 Array{Int64,2}:
 0  1  1  1  1  0  1  1  1  1  1  0  1  …  2  1  1  1  2  2  0  1  1  1  2  0
 1  2  1  2  1  1  1  1  1  1  1  1  1     1  1  1  2  0  0  1  0  0  1  1  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
 0  2  0  2  2  0  0  0  2  0  0  1  1     1  0  1  2  1  1  1  0  0  2  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  0  0  2  0  0  1  1  …  2  2  2  1  2  2  0  1  1  1  2  0
 0  2  0  2  2  1  1  1  1  1  1  1  2     1  0  1  2  0  1  1  0  0  1  2  2
 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
 0  0  2  0  0  0  2  2  0  2  2  0  2     2  1  2  1  2  1  1  1  1  0  1  1
 0  2  1  2  1  0  0  0  2  0  0  2  1     0  0  2  2  1  1  1  1  1  1  2  1
 0  1  1  2  2  1  2  2  0  2  2  0  2  …  2  1  1  2  1  2  0  0  0  1  2  2
 0  2  0  2  2  1  1  1  1  1  1  0  1     2  2  1  1  1  2  0  1  1  0  2  1
 0  2  0  2  2  1  1  1  1  1  0  1  1     2  0  1  2  2  1  0  0  0  1  1  2
 ⋮              ⋮              ⋮        ⋱           ⋮              ⋮
 0  2  0  2  2  0  1  1  1  1  1  1  2     2  1  2  2  1  2  0  0  0  1  2  1
 0  1  1  1  1  0  1  1  1  1  1  1  2     1  0  2  2  1  1  1  0  0  1  2  2
 1  1  1  2  2  1  1  1  1  1  1  1  2  …  2  2  2  1  2  2  0  1  1  1  2  1
 1  2  0  2  2  1  1  1  2  0  0  1  0     2  1  1  1  2  2  0  1  1  1  2  0
 1  2  1  2  1  0  0  0  2  0  0  2  1     2  1  1  1  1  2  0  1  1  0  2  1
 1  2  0  2  2  1  1  1  1  1  1  0  1     1  1  2  1  1  1  1  1  1  1  2  1
 0  0  2  1  1  0  2  2  0  2  2  0  1     2  1  2  1  2  2  0  1  1  0  2  1
 0  0  2  2  2  0  1  1  1  1  1  1  2  …  2  2  2  0  2  2  0  2  2  0  2  0
 1  2  1  2  1  2  2  2  1  1  1  1  1     1  0  2  1  0  2  1  1  1  1  1  1
 1  2  0  2  2  0  1  1  1  1  1  0  0     2  1  2  1  2  1  1  1  1  0  1  1
 0  2  0  2  2  0  0  0  2  0  0  0  0     1  0  2  2  1  1  1  0  0  2  1  2
 0  2  0  2  2  1  1  1  1  1  1  1  2     2  1  1  1  1  2  0  1  1  0  2  1
```

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

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

```
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]:
```
0.13369552783537533
```

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

          WARNING: imported binding for cor overwritten in module Main

Out[100]: 0.36790566812123354
```

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

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

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

Out[102]: 50-element Array{Float64,1}:
           0.202076
           0.194863
           0.199447
           0.199195
           0.198119
           0.202193
           0.201398
           0.199462
           0.19831
           0.198579
           0.196568
           0.202214
           0.202701
           ⋮
           0.200924
           0.198447
           0.198919
           0.198246
           0.202284
           0.195199
           0.201144
           0.200895
           0.200895
           0.201279
           0.201697
           0.200644
```

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

Out[103]: 48000-element Array{Float64,1}:
            9.98625
            9.39206
           11.1835
           10.9972
            9.99081
           12.396
           10.9921
           11.8045
           11.3759
           10.3965
           11.201
            9.78874
           10.984
            ⋮
           11.9934
           12.1837
           11.599
           11.8047
           12.5956
           12.6022
           12.3885
           14.1797
           10.9858
           13.996
           12.9797
           12.5994
```

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

Out[104]: 10.699431430355798
```

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

Out[105]: 11.313523040061126
```

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

Out[106]: 11.54402059200891
```

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

Out[107]: 11.746503405438915

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

Out[108]: 12.018419512809087

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

Out[109]: 12.190287777475238

In [110]: EAlphaG=onlyQTL*QTLo

Out[110]: 9000-element Array{Float64,1}:
           11.3759
           11.5923
           10.7778
           11.5901
           12.2021
           11.9822
           12.7786
           12.7865
           13.1841
           11.3869
           11.785
           10.5914
           11.7924
            ⋮
           11.9934
           12.1837
           11.599
           11.8047
           12.5956
           12.6022
           12.3885
           14.1797
           10.9858
           13.996
           12.9797
           12.5994
```

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

Out[111]: 12.175657773084367

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

Out[112]: 1.4762263427285696

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

Out[113]: 11.911632365234935

In [114]: meanBreedingValueSelectedG0=meanEAlphaS0-meanEAlphaG0

Out[114]: 1.212200934879137

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

Out[115]: 11.798202250711615

In [116]: meanBreedingValueSelectedG1=meanEAlphaS1-meanEAlphaG0

Out[116]: 1.098770820355817

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

Out[117]: 11.943757392406432

In [118]: meanBreedingValueSelectedG2=meanEAlphaS2-meanEAlphaG0

Out[118]: 1.2443259620506346

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

Out[119]: 12.285358806645345

In [120]: meanBreedingValueSelectedG3=meanEAlphaS3-meanEAlphaG0

Out[120]: 1.585927376289547
```

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

Out[121]: 12.35413787478871

In [122]: meanBreedingValueSelectedG4=meanEAlphaS4-meanEAlphaG0

Out[122]: 1.6547064444329127

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

Out[123]: 12.190287777475238

In [124]: meanBreedingValueSelectedG5=meanEAlphaS5-meanEAlphaG0

Out[124]: 1.4908563471194398
```