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)
```

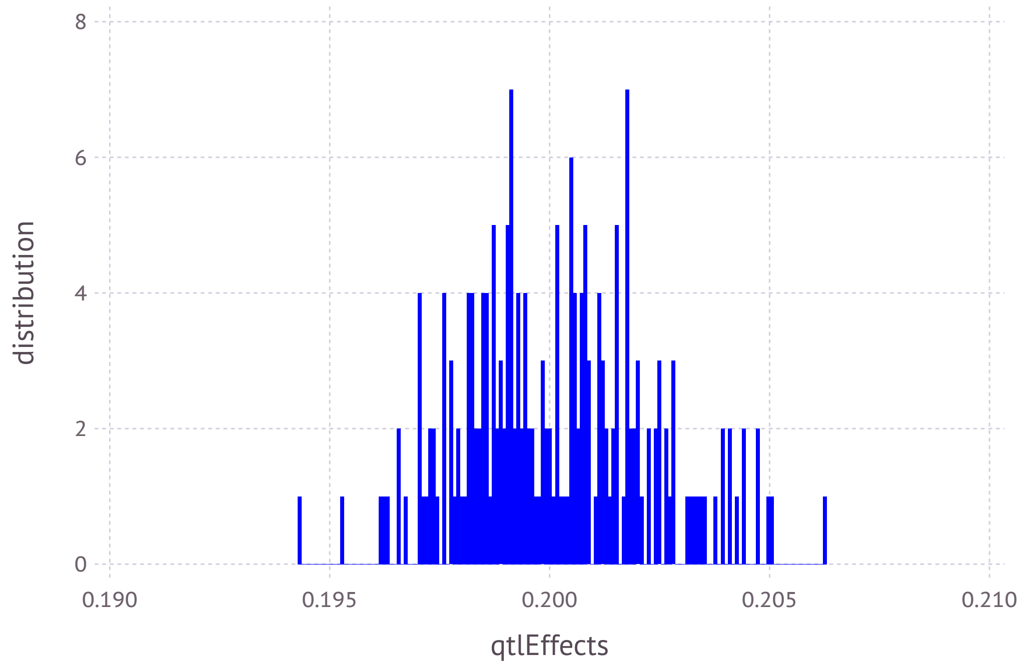
```
In [7]: qtlEffects
```

```
Out[7]: 200-element Array{Float64,1}:
         0.198558
         0.199151
         0.205085
         0.197745
         0.199973
         0.202516
         0.197115
         0.200831
         0.198221
         0.203483
         0.198039
         0.199497
         0.197081
         ⋮
         0.202808
         0.198719
         0.199645
         0.200094
         0.202092
         0.200182
         0.199038
         0.203781
         0.199859
         0.199396
         0.203291
         0.198235
```

```
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]:
```
0.20012199284492765
```

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

Out[11]:
```
4.449303776170777e-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
```

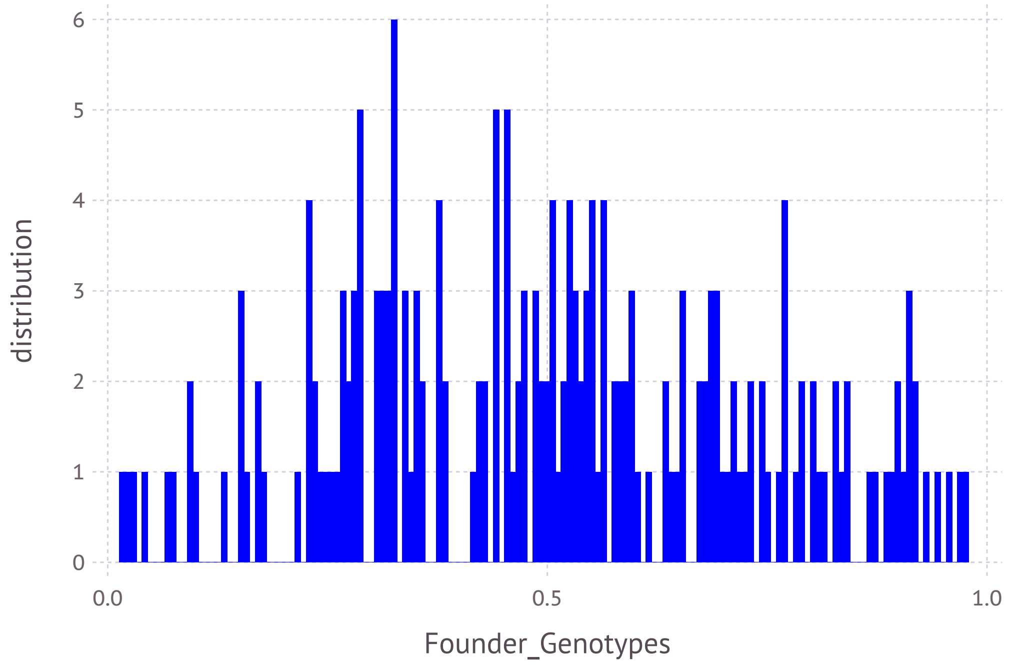
In [19]:
```
gSPSire = XSim.getOurGenotypes(popSP[1])
gSPDam = XSim.getOurGenotypes(popSP[2])
gSP = [gSPSire;gSPSire];
```

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

Out[20]:
```
1x200 Array{Float64,2}:
 0.066375  0.838625  0.282  0.947  0.82675  …  0.383375  0.89725  0.546125
```

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);
```

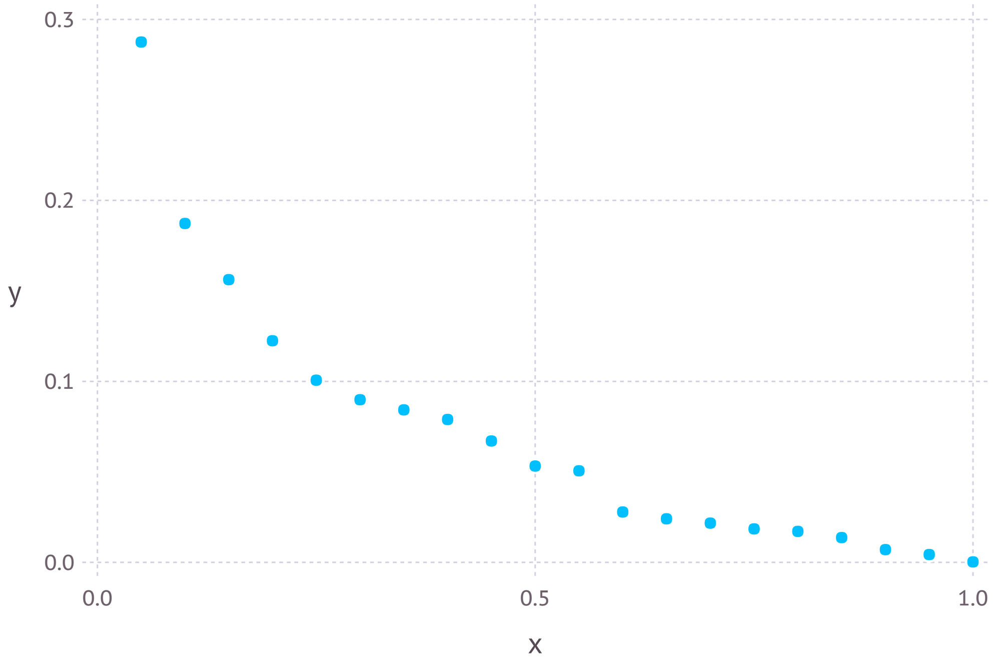
```
In [23]: for i=1:(nRows-20)
         LDMat[i,:] = corMat[i,(i+1):(i+20)].^2
         end
```

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

```
Out[24]: 1x20 Array{Float64,2}:
          0.0002876  0.00431906  0.00706058  …  0.156227  0.187263  0.287525
```

```
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]:
```
11.179763535417054
```

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

Out[31]:
```
0.8311891603900378
```

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

Out[32]:
```
1.9394413742434216
```

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

Out[33]:
```
1.9394413742434216
```

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]: 13.597574691564768

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

Out[36]: 13.583372633007723

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

Out[37]: 0.6014726342812449

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

Out[38]: 0.6179078413218262
```

Pedigree: All animals

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

Out[39]:
```
48000x3 Array{Int64,2}:
 40722  33530  36792
 40723  34514  39682
 40724  33511  38299
 40725  34384  39831
 40726  33459  39232
 40727  36529  37061
 40728  35875  38676
 40729  35569  36764
 40730  36224  39721
 40731  35858  38040
 40732  36214  37541
 40733  33263  37875
 40734  36357  38644
     ⋮
 88710  76718  77265
 88711  73432  77713
 88712  73521  76993
 88713  75932  78427
 88714  76175  77429
 88715  74989  80605
 88716  73512  79229
 88717  72762  79861
 88718  73002  79666
 88719  76281  79971
 88720  76398  78159
 88721  75932  79810
```

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  1  1  1  1  …  1  0  1  2  0  1  1  0  0  1  2  2
          40723  2  2  0  2  2  0  1  1  1  1     2  2  1  1  2  2  0  1  1  0  2  1
          40724  0  2  0  2  2  0  0  0  2  0     2  1  1  1  1  2  1  1  1  1  1  1
          40725  0  2  0  2  2  0  0  0  2  0     1  0  1  2  1  1  0  0  0  2  1  1
          40726  0  1  1  2  2  1  1  1  1  1     2  1  1  0  2  2  0  1  1  1  0  2
          40727  0  2  1  2  1  0  0  0  2  0  …  1  1  1  2  0  1  2  0  0  2  1  2
          40728  0  2  0  2  2  0  0  0  2  0     2  1  1  1  1  2  0  1  1  0  2  1
          40729  0  2  1  2  1  0  0  0  2  0     1  1  2  1  1  1  1  1  1  1  2  1
          40730  0  2  0  2  2  1  2  2  0  2     2  2  1  1  1  1  0  1  1  0  1  1
          40731  0  2  0  2  2  1  2  2  0  2     2  1  2  1  1  1  0  1  1  1  1  1
          40732  0  2  1  2  1  0  0  0  2  0  …  1  1  2  1  1  1  1  1  1  1  2  1
          40733  0  1  1  2  2  1  1  1  1  1     2  1  1  1  2  2  0  1  1  1  2  0
          40734  0  2  0  2  2  0  0  0  2  0     2  1  1  2  0  1  1  0  0  1  1  2
              ⋮              ⋮              ⋮  ⋱     ⋮              ⋮              ⋮
          88710  0  0  2  2  2  2  2  2  0  2     1  0  2  2  0  0  2  0  0  2  2  2
          88711  0  0  2  1  1  0  2  2  0  2     1  0  1  1  0  1  1  0  0  1  2  1
          88712  1  2  0  2  2  2  2  2  0  2  …  2  2  2  0  2  2  0  1  1  1  1  1
          88713  0  1  1  2  2  1  1  1  1  1     1  0  2  2  1  1  1  0  0  2  1  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  1  2  0  2  2  0  1  1  1  1     2  2  2  0  2  2  0  2  2  0  2  0
          88716  1  2  0  2  2  0  1  1  1  1     2  2  2  0  2  2  0  2  2  0  2  0
          88717  1  2  0  2  2  0  1  1  2  0  …  2  2  2  0  2  2  0  2  2  0  2  1
          88718  0  1  1  2  2  2  2  2  0  2     2  2  2  0  2  2  0  2  2  0  2  0
          88719  0  2  0  2  2  1  1  1  1  1     2  2  2  1  2  2  0  1  1  1  2  1
          88720  0  2  1  2  1  0  1  1  2  0     2  1  1  1  1  2  0  1  1  0  2  1
          88721  0  2  0  2  2  0  1  1  2  0     0  1  1  1  1  1  1  0  0  1  2  1
```

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

Create marker file for all animals

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

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

```
In [49]: Mstream = open(GenAll, "w")
```

```
Out[49]: IOStream(<file GenAll.txt>)
```

In [50]:
```
for i in 1:size(M,1)
    @printf(Mstream, "%19d", allID[i])
    for j in 1:size(M,2)
        @printf(Mstream, "%3d", M[i,j])
    end
    @printf(Mstream, "\n")
end
```

In [51]:
```
close(Mstream)
```

Genotypes - all sires and all offsprings in G5

```
In [52]: AllSire = PED[posOFF1S:posOFF5E,2]
```

```
Out[52]: 40000-element Array{Int64,1}:
          40923
          44403
          42912
          41948
          41113
          41104
          43233
          43130
          42192
          41090
          44152
          40838
          43717
              ⋮
          76718
          73432
          73521
          75932
          76175
          74989
          73512
          72762
          73002
          76281
          76398
          75932
```

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

```
Out[53]: 1000-element Array{Int64,1}:
          40923
          44403
          42912
          41948
          41113
          41104
          43233
          43130
          42192
          41090
          44152
          40838
          43717
              ⋮
          74505
          73521
          73057
          73551
          75521
          76301
          75357
          72762
          75769
          76597
          76455
          75385
```

```
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}:
          40923
          44403
          42912
          41948
          41113
          41104
          43233
          43130
          42192
          41090
          44152
          40838
          43717
              ⋮
          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  12.581  12.304
          40723   9.705  10.513
          40724  13.29   11.708
          40725   9.454   9.535
          40726  14.642  13.495
          40727  11.24   11.893
          40728  11.198  11.903
          40729   9.535  11.119
          40730  11.591  11.113
          40731  11.631  10.921
          40732   9.98    9.123
          40733   7.723   8.738
          40734   8.343  10.314
              ⋮
          88710  14.709  14.292
          88711  12.885  13.3
          88712  15.469  15.675
          88713  16.688  15.481
          88714  10.87   14.093
          88715  13.858  15.091
          88716  16.864  15.085
          88717  14.185  13.698
          88718  13.115  13.69
          88719  12.201  13.492
          88720  12.957  14.086
          88721  11.368  13.893
```

```
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
            4
            7
            8
           20
           21
           24
           27
           28
           40
           41
           44
           47
            ⋮
          148
          160
          161
          164
          167
          168
          180
          181
          184
          187
          188
          200
```

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

```
Out[83]: 150-element Array{Int64,1}:
            2
            3
            5
            6
            9
           10
           11
           12
           13
           14
           15
           16
           17
            ⋮
          186
          189
          190
          191
          192
          193
          194
          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  1  1  2  2  1  1  1  1  1  1  1  2  …  2  1  1  0  2  2  0  1  1  1  0  2
          0  2  1  2  1  0  0  0  2  0  0  1  0     2  2  2  0  2  2  0  2  2  0  2  0
          1  2  0  2  2  1  1  1  1  1  1  1  2     2  0  0  2  0  1  1  0  0  1  2  2
          0  2  0  2  2  0  0  0  2  0  0  2  2     2  2  2  0  2  2  0  2  2  0  2  0
          0  1  1  2  2  1  2  2  0  2  2  0  1     1  0  1  2  0  1  1  0  0  1  2  2
          1  1  1  1  1  1  2  2  1  1  1  1  1  …  1  0  2  2  0  0  1  0  0  1  2  2
          0  1  1  2  2  1  2  2  0  2  2  1  1     2  2  2  0  2  2  0  2  2  0  2  0
          1  2  1  2  1  1  1  1  1  1  1  1  1     2  0  2  2  0  0  0  0  0  2  0  2
          0  2  0  2  2  1  1  1  1  1  1  1  2     2  1  1  2  2  2  1  1  1  1  2  2
          0  2  0  2  2  1  1  1  1  1  1  0  1     1  1  2  1  2  2  0  2  2  0  2  0
          0  1  2  2  1  1  1  1  1  1  1  1  1  …  2  2  2  0  2  2  0  2  2  0  2  0
          0  2  0  2  2  0  1  1  2  0  0  2  2     2  1  2  1  1  2  0  1  1  0  1  1
          0  2  0  2  2  0  0  0  2  0  0  0  0     2  0  1  1  1  2  1  1  1  1  1  1
          ⋮              ⋮              ⋮        ⋱           ⋮              ⋮
          0  0  2  2  2  2  2  2  0  2  2  0  2     1  0  2  2  0  0  2  0  0  2  2  2
          0  0  2  1  1  0  2  2  0  2  2  0  1     1  0  1  1  0  1  1  0  0  1  2  1
          1  2  0  2  2  2  2  2  0  2  2  0  2  …  2  2  2  0  2  2  0  1  1  1  1  1
          0  1  1  2  2  1  1  1  1  1  1  1  2     1  0  2  2  1  1  1  0  0  2  1  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
          1  2  0  2  2  0  1  1  1  1  1  1  1     2  2  2  0  2  2  0  2  2  0  2  0
          1  2  0  2  2  0  1  1  1  1  1  1  1     2  2  2  0  2  2  0  2  2  0  2  0
          1  2  0  2  2  0  1  1  2  0  0  2  1  …  2  2  2  0  2  2  0  2  2  0  2  1
          0  1  1  2  2  2  2  2  0  2  2  0  2     2  2  2  0  2  2  0  2  2  0  2  0
          0  2  0  2  2  1  1  1  1  1  1  0  1     2  2  2  1  2  2  0  1  1  1  2  1
          0  2  1  2  1  0  1  1  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  2  0  0  2  2     0  1  1  1  1  1  1  0  0  1  2  1
```

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

Out[86]:
```
9000x50 Array{Int64,2}:
 0  2  1  1  2  1  1  1  1  1  2  1  2  …  1  0  1  2  1  2  0  2  2  1  1  2
 0  2  0  0  1  1  0  1  1  0  2  1  2     0  2  1  2  0  2  2  2  2  2  2  0
 1  2  1  1  2  2  1  0  0  1  2  2  2     1  0  2  2  1  2  1  2  2  0  0  2
 0  2  0  0  2  1  0  1  0  1  1  1  1     2  0  2  2  0  2  2  2  2  2  2  0
 0  2  2  2  1  2  1  0  0  0  2  1  1     2  1  1  2  1  2  2  2  2  0  0  2
 1  1  2  2  1  2  1  0  0  1  2  2  2  …  1  0  1  2  0  2  1  2  2  1  0  2
 0  2  2  2  1  1  2  1  1  2  2  1  1     1  1  1  2  1  2  1  2  2  2  2  0
 1  2  1  1  2  1  2  1  1  2  2  1  1     1  1  1  2  0  2  1  2  2  0  0  2
 0  2  1  1  2  0  0  2  1  2  1  1  1     1  1  2  2  0  1  1  2  2  0  0  2
 0  2  1  1  2  2  2  1  1  2  2  1  2     1  1  2  2  1  2  1  2  2  1  1  0
 0  2  1  1  2  1  1  2  1  1  2  1  2  …  2  0  0  2  1  1  2  2  2  2  2  0
 0  2  1  1  2  1  1  1  1  1  1  0  0     1  1  1  2  1  2  2  2  2  2  2  1
 0  2  0  0  0  1  2  1  1  2  0  0  0     1  1  2  2  0  1  2  1  2  2  1  1
 ⋮              ⋮              ⋮        ⋱           ⋮              ⋮
 0  2  2  2  2  0  2  2  2  2  2  0  1     0  1  2  2  2  2  1  2  2  1  1  2
 0  1  2  2  1  1  1  1  0  1  2  2  2     2  0  2  2  1  2  1  2  2  0  0  1
 1  2  2  2  2  1  2  1  1  2  2  2  2  …  0  2  1  2  2  2  0  1  2  2  2  1
 0  2  1  1  2  2  1  1  1  1  2  1  2     2  2  2  2  1  2  1  2  2  1  1  2
 0  2  1  1  2  2  0  0  0  1  2  2  2     2  0  2  2  1  2  2  2  2  2  2  0
 1  2  1  1  1  2  2  1  1  2  2  2  2     2  0  2  2  1  2  2  2  2  2  2  0
 1  2  1  1  1  0  2  2  2  2  2  1  1     1  1  2  2  2  2  2  2  2  2  2  0
 1  2  1  1  1  1  1  1  1  1  2  2  2  …  1  1  1  2  1  2  1  2  2  2  2  1
 0  2  2  2  2  1  2  1  1  2  2  2  2     0  1  2  2  1  2  1  2  2  2  2  0
 0  2  1  1  2  2  2  0  0  2  2  1  1     1  2  1  2  0  2  1  2  2  1  1  1
 0  2  1  1  2  0  2  2  2  2  1  1  1     0  1  2  2  1  2  1  2  2  1  1  1
 0  2  1  1  2  2  2  0  0  1  2  2  2     1  1  2  2  1  2  0  1  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]:
```
0.439511482384733
```

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

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

Out[100]:
```
0.6643706192639757
```

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

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

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

```
Out[102]: 50-element Array{Float64,1}:
           0.198558
           0.197745
           0.197115
           0.200831
           0.197943
           0.198233
           0.203954
           0.201409
           0.198914
           0.204741
           0.201086
           0.199276
           0.202783
           ⋮
           0.202246
           0.198318
           0.201654
           0.198299
           0.199069
           0.199323
           0.203227
           0.198651
           0.197316
           0.200559
           0.200177
           0.198235
```

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

Out[103]:
```
48000-element Array{Float64,1}:
 12.4083
 10.5984
 11.827
  9.61278
 13.6004
 12.0129
 12.0159
 11.211
 11.1858
 10.9966
  9.19219
  8.80236
 10.4123
  ⋮
 14.4198
 13.4267
 15.8127
 15.6032
 14.2115
 15.2392
 15.2318
 13.8137
 13.8135
 13.6175
 14.2231
 14.0139
```

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

Out[104]:
```
11.274748624490647
```

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

Out[105]:
```
11.936790515990033
```

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

Out[106]:
```
12.464521745515347
```

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

Out[107]: 12.945351190218108

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

Out[108]: 13.351625323143649

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

Out[109]: 13.712351458718398

In [110]: EAlphaG=onlyQTL*QTLo

Out[110]: 9000-element Array{Float64,1}:
           13.6004
           12.413
           12.9988
           11.8099
           12.6017
           12.6058
           13.4164
           12.4041
           12.6091
           13.4203
           13.8188
           13.4127
           10.6023
            ⋮
           14.4198
           13.4267
           15.8127
           15.6032
           14.2115
           15.2392
           15.2318
           13.8137
           13.8135
           13.6175
           14.2231
           14.0139
```

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

Out[111]: 13.671652038391931

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

Out[112]: 2.396903413901285

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

Out[113]: 12.557749347588247

In [114]: meanBreedingValueSelectedG0=meanEAlphaS0-meanEAlphaG0

Out[114]: 1.2830007230976008

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

Out[115]: 12.959760675101192

In [116]: meanBreedingValueSelectedG1=meanEAlphaS1-meanEAlphaG0

Out[116]: 1.6850120506105455

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

Out[117]: 13.370386707319023

In [118]: meanBreedingValueSelectedG2=meanEAlphaS2-meanEAlphaG0

Out[118]: 2.0956380828283763

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

Out[119]: 13.747396898629063

In [120]: meanBreedingValueSelectedG3=meanEAlphaS3-meanEAlphaG0

Out[120]: 2.4726482741384164
```

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

Out[121]: 14.09498975026347

In [122]: meanBreedingValueSelectedG4=meanEAlphaS4-meanEAlphaG0

Out[122]: 2.8202411257728226

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

Out[123]: 13.712351458718398

In [124]: meanBreedingValueSelectedG5=meanEAlphaS5-meanEAlphaG0

Out[124]: 2.437602834227752
```