In [1]:
```
# Founders: real haplotype data (ch1to10.200SNP)
# "Sample animals for sire and dam candidates" & "Dams and Sires": dam size = N of dams
# One Dam -> 1 male & 1 female (1 dam -> 2 offsprings)
# selection: increase
# 5 generation selection: increase
# heritability = 0.5
# Phenotypes : all animals in G0 to G4
# Genotypes  : all progeny in G5 and all sires in each generation
# Change muAlpha = 0.2
# 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)
```

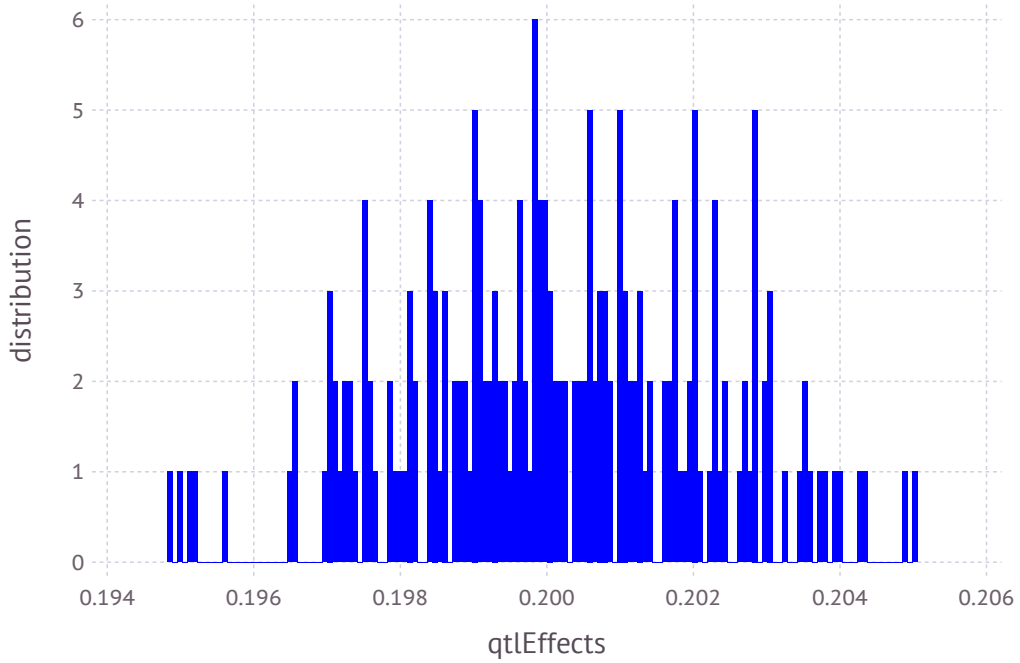
In [7]:
```
qtlEffects
```

Out[7]:
```
200-element Array{Float64,1}:
 0.200442
 0.199111
 0.197513
 0.202961
 0.200058
 0.200672
 0.199245
 0.19823
 0.202724
 0.199032
 0.202374
 0.20282
 0.200567
 ⋮
 0.199324
 0.199629
 0.20067
 0.202971
 0.203766
 0.199537
 0.198913
 0.201093
 0.200159
 0.198723
 0.199438
 0.201971
```

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]: 0.20013869455171027

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

Out[11]: 4.209941326326101e-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
```

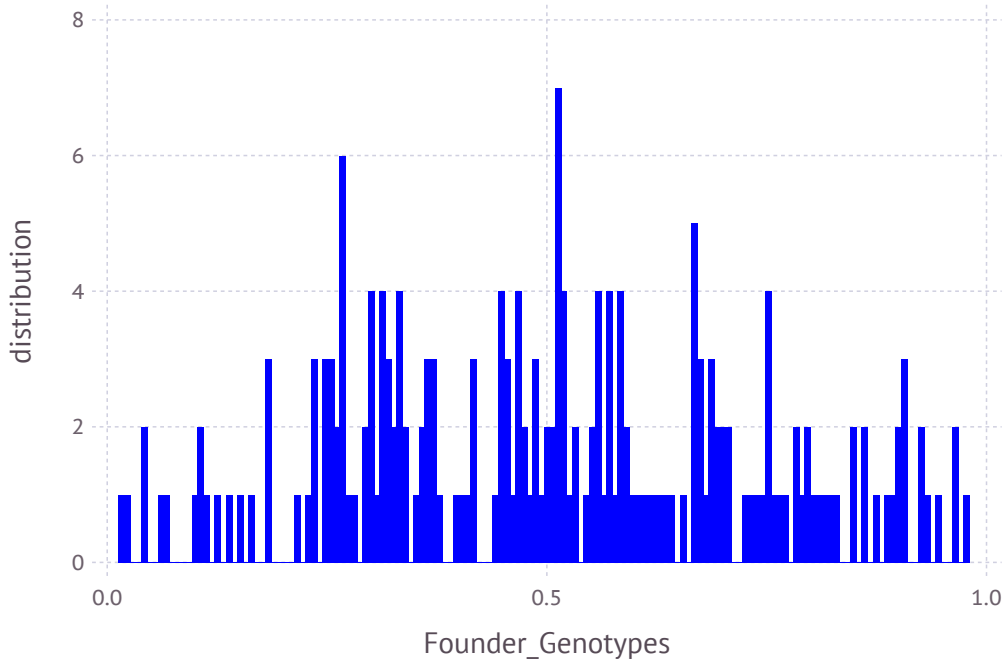
In [19]:
```
gSPSire = XSim.getOurGenotypes(popSP[1])
gSPDam = XSim.getOurGenotypes(popSP[2])
gSP = [gSPSire;gSPSire];
```

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

Out[20]:
```
1x200 Array{Float64,2}:
 0.066625  0.849375  0.27075  0.9465  …  0.367  0.379625  0.901  0.555125
```

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);
```

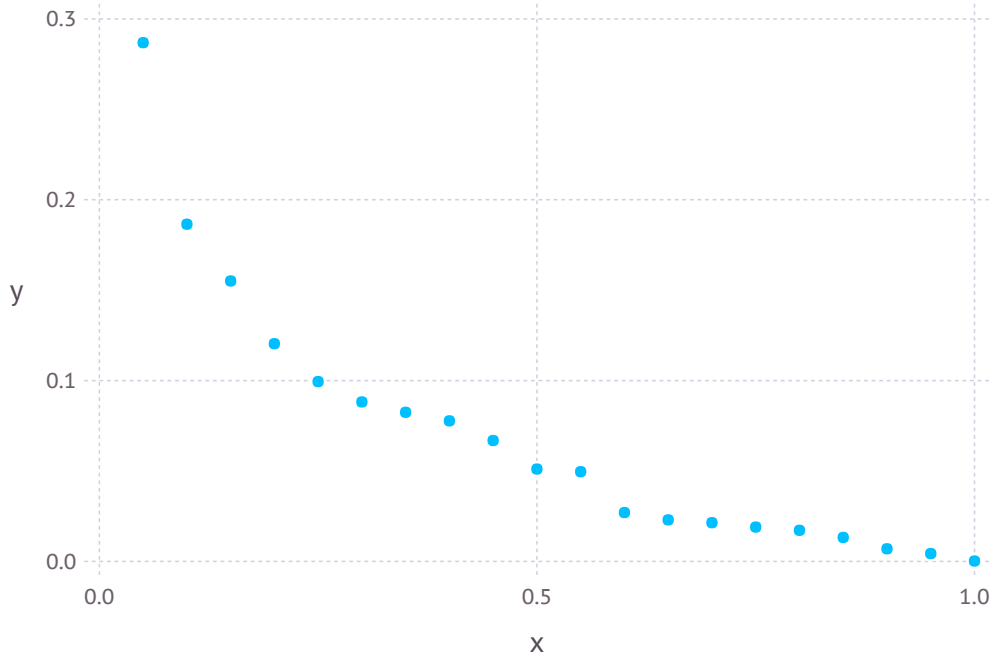
In [23]:
```
for i=1:(nRows-20)
LDMat[i,:] = corMat[i,(i+1):(i+20)].^2
end
```

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

Out[24]:
```
1x20 Array{Float64,2}:
 0.00024352  0.00439141  0.00700461  …  0.155102  0.186456  0.286852
```

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]:
```
8.537599422029501
```

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

Out[31]:
```
0.5837096004565842
```

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

Out[32]:
```
0.5837096004565842
```

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

Out[33]:
```
0.5837096004565842
```

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

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

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

Out[35]: 11.050764198072008

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

Out[36]: 11.054165503688147

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

Out[37]: 0.45281780875715855

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

Out[38]: 0.4719556364805182

Pedigree: All animals

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

Out[39]:
```
48000x3 Array{Int64,2}:
 40722  32898  36728
 40723  34252  40496
 40724  35692  39422
 40725  34378  39471
 40726  34727  36915
 40727  35506  38516
 40728  35012  38686
 40729  33236  38406
 40730  35682  39510
 40731  35391  37422
 40732  32883  40616
 40733  33155  37675
 40734  32916  40236
     ⋮
 88710  75782  80568
 88711  75917  78870
 88712  76600  78226
 88713  76531  79064
 88714  75798  78859
 88715  76522  80369
 88716  76240  79502
 88717  74857  80052
 88718  76004  78432
 88719  76532  79511
 88720  75514  80231
 88721  73908  80563
```

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	1	1	1	1	…	2	0	1	2	1	2	0	0	0	1	1	2
40723	0	2	0	2	2	0	1	1	1	1		2	1	1	1	1	2	0	1	1	0	2	1
40724	0	2	0	2	2	1	1	1	1	1		1	0	1	2	1	1	1	0	0	2	2	1
40725	0	2	0	2	2	1	1	1	1	1		2	1	1	1	1	2	0	1	1	0	2	1
40726	0	2	0	2	2	0	0	0	2	0		1	1	1	2	0	0	2	0	0	2	2	2
40727	0	2	1	2	2	1	2	2	1	1	…	2	0	2	2	2	2	0	0	0	2	1	2
40728	0	2	0	2	2	0	0	0	2	0		1	1	2	1	1	1	1	1	1	1	2	1
40729	0	0	2	0	0	0	2	2	0	2		2	1	1	1	2	2	0	1	1	1	2	0
40730	0	2	0	2	2	0	0	0	2	0		2	2	2	1	2	2	0	1	1	1	2	1
40731	0	1	1	2	2	0	1	1	1	1		2	1	1	1	1	2	0	1	1	0	2	1
40732	0	1	1	1	1	0	1	1	1	1	…	1	0	1	2	1	2	0	1	1	0	2	1
40733	0	2	1	2	1	0	0	0	2	0		2	0	1	2	1	2	0	0	0	1	1	1
40734	0	1	1	2	2	1	2	2	0	2		1	2	1	2	1	2	0	1	1	0	2	2
⋮				⋮				⋮	⋱			⋮				⋮					⋮		
88710	0	1	1	2	2	1	1	1	1	1		2	2	1	1	1	1	1	1	1	1	2	1
88711	1	2	0	2	2	1	2	2	0	2		2	1	2	1	2	1	1	1	1	1	2	1
88712	1	2	0	2	2	2	2	2	0	2	…	2	2	1	1	1	1	0	1	1	0	1	1
88713	0	2	0	2	2	0	1	1	1	1		2	2	2	0	2	2	0	2	2	0	2	0
88714	0	2	0	2	2	2	2	2	0	2		2	2	2	0	2	2	0	2	2	0	2	0
88715	0	1	1	2	2	2	2	2	0	2		1	1	2	1	0	0	1	0	0	1	2	2
88716	0	2	0	2	2	1	1	1	1	1		2	2	2	1	2	2	0	1	1	0	2	0
88717	1	1	1	1	1	1	2	2	0	2	…	2	2	2	0	2	2	0	2	2	0	2	0
88718	0	1	1	2	2	1	2	2	0	2		1	1	2	1	1	1	1	1	1	1	2	1
88719	1	2	1	2	1	1	1	1	1	1		2	1	1	2	1	1	1	0	0	1	2	2
88720	1	2	1	2	1	1	1	1	1	1		2	2	1	1	2	2	0	1	1	0	2	1
88721	0	1	1	2	2	1	1	1	1	1		2	2	2	0	2	2	0	2	2	0	2	0

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

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}:
 44458
 42022
 42828
 41239
 44521
 44120
 43027
 42530
 44284
 43168
 44328
 43554
 42290
    ⋮
 75782
 75917
 76600
 76531
 75798
 76522
 76240
 74857
 76004
 76532
 75514
 73908
```

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

Out[53]:
```
1000-element Array{Int64,1}:
 44458
 42022
 42828
 41239
 44521
 44120
 43027
 42530
 44284
 43168
 44328
 43554
 42290
     ⋮
 75082
 73538
 76579
 76706
 73723
 76277
 76376
 76367
 73307
 75105
 76022
 75278
```

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}:
 44458
 42022
 42828
 41239
 44521
 44120
 43027
 42530
 44284
 43168
 44328
 43554
 42290
     ⋮
 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   7.548   8.406
 40723   6.54    6.976
 40724   8.884   9.189
 40725   8.79    9.402
 40726   7.234   7.788
 40727   9.737  10.399
 40728  10.097   9.178
 40729   7.941   7.996
 40730   7.434   7.598
 40731   7.838   7.597
 40732   7.93    8.783
 40733  10.443  10.006
 40734   7.863   8.181
     ⋮
 88710  10.287  11.588
 88711  12.451  12.368
 88712  12.042  12.185
 88713  11.116  12.395
 88714  11.303  11.376
 88715  12.121  11.4
 88716  12.446  11.986
 88717  11.078  12.192
 88718  11.377  11.393
 88719  12.306  12.204
 88720  11.195  11.399
 88721  13.539  12.784
```

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}:
   2
   6
   8
  15
  16
  22
  26
  28
  35
  36
  42
  46
  48
   ⋮
 155
 156
 162
 166
 168
 175
 176
 182
 186
 188
 195
 196
```

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

Out[83]:
```
150-element Array{Int64,1}:
   1
   3
   4
   5
   7
   9
  10
  11
  12
  13
  14
  17
  18
   ⋮
 185
 187
 189
 190
 191
 192
 193
 194
 197
 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	2	0	2	2	0	2	2	1	1	1	1	2	…	2	1	1	1	1	2	0	1	1	0	2	1
0	1	1	1	1	1	2	2	0	2	2	0	2		1	0	1	2	0	1	1	0	0	1	2	2
0	1	1	2	2	1	1	1	1	1	1	0	1		2	2	2	0	2	2	0	2	2	0	2	0
0	1	2	2	1	1	1	1	1	1	1	1	2		2	2	2	1	2	2	0	1	1	1	2	1
0	2	0	2	2	1	1	1	1	1	1	0	1		2	1	0	2	0	1	0	0	0	0	1	2
0	1	1	1	1	0	1	2	0	2	2	0	2	…	2	2	2	1	2	2	0	1	1	1	2	1
0	1	1	2	2	0	2	2	1	1	1	1	1		2	1	1	2	2	2	0	0	0	2	2	0
0	1	1	1	1	0	1	1	1	1	1	0	1		1	1	2	2	1	1	1	0	0	2	2	2
0	2	0	2	2	0	0	0	2	0	0	2	1		1	0	1	2	1	0	1	0	0	1	1	2
0	2	0	2	2	1	2	2	0	2	2	0	2		2	2	2	0	2	2	0	2	2	0	2	0
0	2	0	2	2	0	0	0	2	0	0	0	0	…	1	2	2	1	1	2	0	1	1	1	1	1
1	1	1	2	2	1	1	1	1	1	1	1	1		1	1	1	2	0	1	1	0	0	1	2	2
0	1	1	2	2	1	1	1	1	1	1	1	2		2	2	2	0	2	2	0	2	2	0	2	0
⋮				⋮				⋮			⋱			⋮					⋮						
0	1	1	2	2	1	1	1	1	1	1	1	2		2	2	1	1	1	1	1	1	1	1	2	1
1	2	0	2	2	1	2	2	0	2	2	0	2		2	1	2	1	2	1	1	1	1	1	2	1
1	2	0	2	2	2	2	2	0	2	2	0	2	…	2	2	1	1	1	1	0	1	1	0	1	1
0	2	0	2	2	0	1	1	1	1	1	1	2		2	2	2	0	2	2	0	2	2	0	2	0
0	2	0	2	2	2	2	2	0	2	2	0	2		2	2	2	0	2	2	0	2	2	0	2	0
0	1	1	2	2	2	2	2	0	2	2	0	2		1	1	2	1	0	0	1	0	0	1	2	2
0	2	0	2	2	1	1	1	1	1	1	0	1		2	2	2	1	2	2	0	1	1	0	2	0
1	1	1	1	1	1	2	2	0	2	2	0	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	2		1	1	2	1	1	1	1	1	1	1	2	1
1	2	1	2	1	1	1	1	1	1	1	1	1		2	1	1	2	1	1	1	0	0	1	2	2
1	2	1	2	1	1	1	1	1	1	1	1	1		2	2	1	1	2	2	0	1	1	0	2	1
0	1	1	2	2	1	1	1	1	1	1	0	2		2	2	2	0	2	2	0	2	2	0	2	0

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

Out[86]:
```
9000x50 Array{Int64,2}:
 2  0  2  1  1  1  0  1  1  2  1  2  0  …  1  1  1  1  2  1  0  1  1  1  0  1
 1  1  2  1  1  0  0  2  1  2  2  2  1     2  0  0  0  2  0  0  0  0  0  1  0
 1  1  1  2  0  1  0  0  1  1  1  1  1     0  2  1  1  1  1  0  1  1  1  0  2
 1  1  1  2  0  1  1  2  2  1  2  1  0     0  1  1  1  2  1  0  0  2  1  0  1
 2  1  1  1  1  1  0  1  1  2  2  1  0     1  0  1  0  2  0  0  1  2  1  0  0
 1  0  2  2  0  0  0  1  2  1  2  0  0  …  1  1  0  2  2  2  2  0  2  1  0  1
 1  0  2  0  2  2  0  0  0  2  2  1  0     0  1  0  2  2  1  1  1  1  1  0  0
 1  0  1  2  0  0  0  1  2  1  2  0  0     0  2  1  1  2  1  1  0  1  0  1  0
 2  0  0  1  1  0  0  0  2  0  1  1  1     2  0  0  2  2  1  2  1  0  0  1  0
 2  1  2  1  1  0  0  1  1  2  2  1  0     1  1  1  1  2  1  0  1  1  1  0  2
 2  0  0  2  0  1  0  0  0  2  1  0  1  …  2  0  0  1  2  1  0  1  2  1  0  1
 1  1  1  1  1  1  0  1  2  1  2  2  0     0  0  1  1  2  1  1  0  1  1  1  0
 1  1  1  1  1  1  0  1  1  2  2  1  1     0  1  1  0  2  0  1  0  2  2  0  2
 ⋮                 ⋮                 ⋮        ⋱        ⋮                 ⋮
 1  1  1  1  1  2  1  1  1  2  1  0  1     1  1  0  2  1  2  1  0  2  1  1  1
 2  1  2  0  2  2  1  1  1  2  2  1  0     0  2  2  2  2  1  1  1  2  2  1  1
 2  2  2  0  2  2  1  1  1  2  2  2  2  …  1  1  0  2  2  2  2  1  2  1  0  1
 2  0  1  1  1  1  2  1  1  1  2  2  0     2  0  2  1  2  2  2  0  2  2  0  2
 2  2  2  0  2  2  1  1  1  2  1  2  1     0  1  1  1  1  2  2  0  2  2  0  2
 1  2  2  2  0  1  2  1  2  2  2  2  0     0  1  0  2  2  2  0  1  2  1  1  0
 2  1  1  2  0  1  2  1  1  2  1  1  1     0  2  2  0  2  2  2  1  2  2  0  1
 1  1  2  1  1  2  1  1  2  2  1  2  1  …  1  1  2  1  2  2  1  0  2  2  0  2
 1  1  2  2  0  0  1  1  2  1  1  1  2     0  2  2  1  2  1  2  0  1  1  1  1
 2  1  1  1  1  2  1  2  2  1  1  2  1     1  1  0  2  2  2  2  1  2  1  1  0
 2  1  1  1  1  0  0  2  2  2  2  2  1     1  0  1  2  2  1  1  1  2  1  0  1
 1  1  1  1  1  2  2  2  2  1  1  2  1     1  1  2  2  2  2  0  0  2  2  0  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]: 0.6590653238417754

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

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

Out[100]:
```
0.8105561578976931
```

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

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

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

Out[102]:
```
50-element Array{Float64,1}:
 0.199111
 0.200672
 0.19823
 0.203512
 0.197549
 0.19987
 0.200874
 0.201755
 0.200023
 0.200716
 0.199044
 0.19808
 0.197825
 ⋮
 0.201995
 0.200983
 0.201196
 0.19658
 0.20101
 0.199899
 0.1986
 0.196961
 0.194821
 0.19952
 0.198913
 0.201093
```

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

Out[103]:
```
48000-element Array{Float64,1}:
  8.38196
  6.99434
  9.1977
  9.3962
  7.80477
 10.3812
  9.1876
  7.98948
  7.59329
  7.59265
  8.78962
  9.99118
  8.19555
  ⋮
 11.5778
 12.3881
 12.1578
 12.3782
 11.3956
 11.3992
 11.9985
 12.181
 11.3958
 12.1639
 11.365
 12.786
```

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

Out[104]:
```
8.535427731296094
```

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

Out[105]:
```
9.040402976667202
```

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

Out[106]:
```
9.605604806773757
```

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

Out[107]:
```
10.089540143122921
```

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

Out[108]:
```
10.558382238437558
```

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

Out[109]:
```
11.045706745224958
```

In [110]:
```
EAlphaG=onlyQTL*QTLo
```

Out[110]:
```
9000-element Array{Float64,1}:
 10.1867
  9.19763
  8.80572
  9.78716
  9.97264
 10.2067
  9.1746
  9.97866
  9.17215
  9.60203
  8.38592
  8.37232
  9.79762
  ⋮
 11.5778
 12.3881
 12.1578
 12.3782
 11.3956
 11.3992
 11.9985
 12.181
 11.3958
 12.1639
 11.365
 12.786
```

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

Out[111]: 10.992266398843896

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

Out[112]: 2.4568386675478013

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

Out[113]: 9.532705658690467

In [114]:
```
meanBreedingValueSelectedG0=meanEAlphaS0-meanEAlphaG0
```

Out[114]: 0.9972779273943733

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

Out[115]: 10.164079584744904

In [116]:
```
meanBreedingValueSelectedG1=meanEAlphaS1-meanEAlphaG0
```

Out[116]: 1.62865185344881

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

Out[117]: 10.581930090652442

In [118]:
```
meanBreedingValueSelectedG2=meanEAlphaS2-meanEAlphaG0
```

Out[118]: 2.0465023593563476

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

Out[119]: 11.029446372915697

In [120]:
```
meanBreedingValueSelectedG3=meanEAlphaS3-meanEAlphaG0
```

Out[120]: 2.4940186416196024

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

Out[121]: 11.51555643197346

In [122]:
```
meanBreedingValueSelectedG4=meanEAlphaS4-meanEAlphaG0
```

Out[122]: 2.9801287006773656

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

Out[123]: 11.045706745224958

In [124]:
```
meanBreedingValueSelectedG5=meanEAlphaS5-meanEAlphaG0
```

Out[124]: 2.510279013928864