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

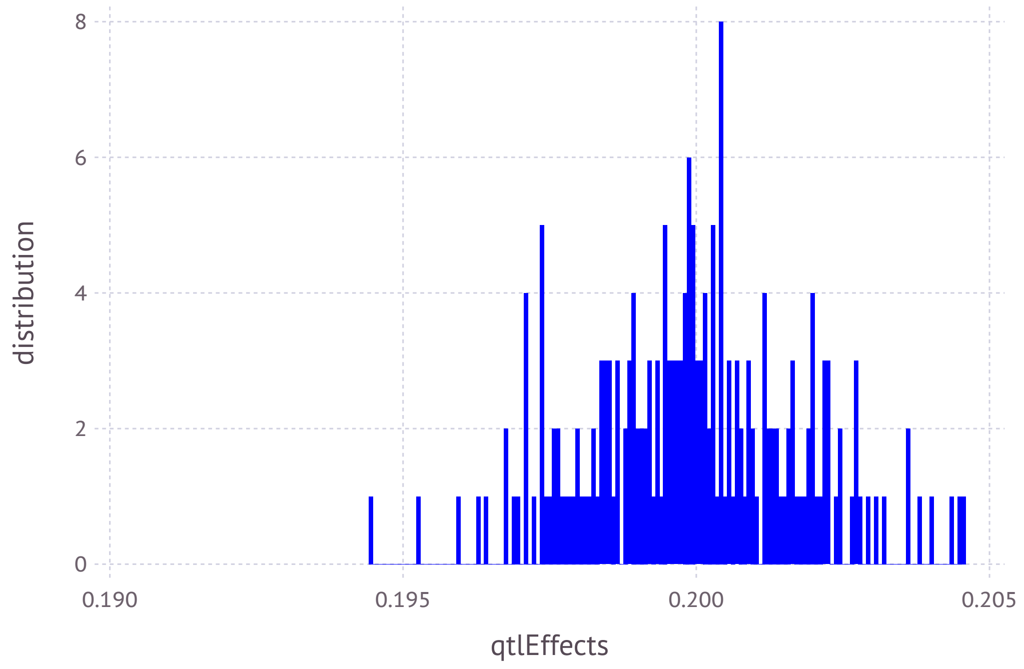
In [6]:
```
numChr    = 10
chrLength = 0.01
numLoci   = 20
nQTL      = 5
mutRate   = 0.0
locusInt  = chrLength/numLoci
mapPos    = collect(locusInt/2:locusInt:chrLength)
geneFreq  = fill(0.5,numLoci)
QTL = sample(1:numLoci,nQTL,replace=false)
qtlMarker = fill(false,numLoci)
qtlMarker[QTL] = true
mu = 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}:
          0.200927
          0.199604
          0.200393
          0.197668
          0.199966
          0.20201
          0.199461
          0.202706
          0.198393
          0.194417
          0.201982
          0.199463
          0.199839
          ⋮
          0.203227
          0.197966
          0.197394
          0.200176
          0.197362
          0.202178
          0.200134
          0.200363
          0.197349
          0.202077
          0.200426
          0.201487
```

```
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]: 0.1999327273209115
```

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

```
Out[11]: 3.3352232061580648e-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
```

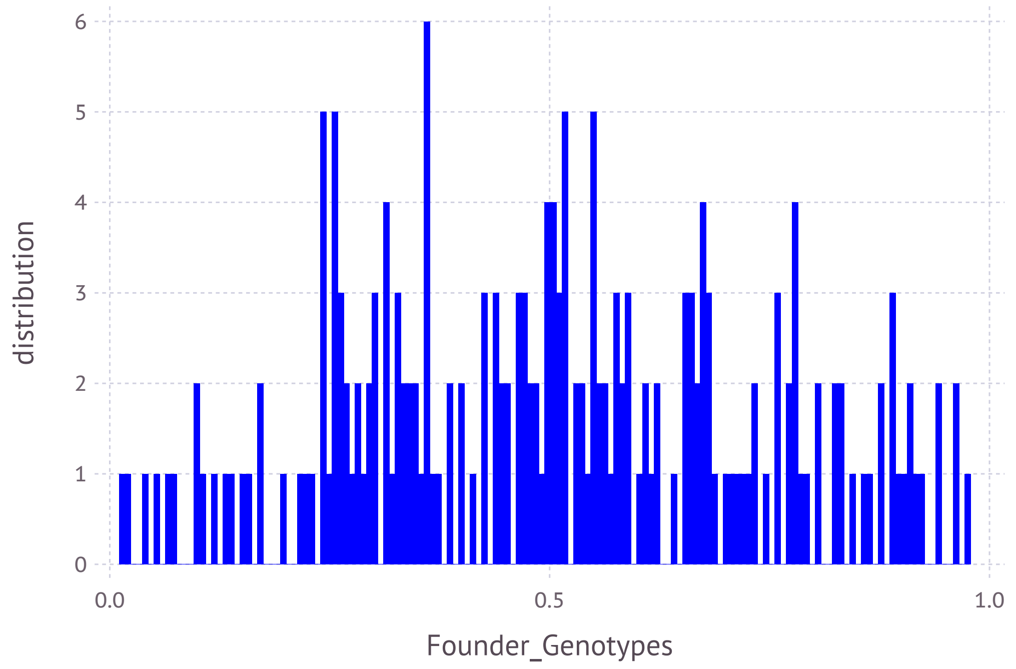
In [19]:
```
gSPSire = XSim.getOurGenotypes(popSP[1])
gSPDam = XSim.getOurGenotypes(popSP[2])
gSP = [gSPSire;gSPSire];
```

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

Out[20]:
```
1x200 Array{Float64,2}:
 0.066  0.8465  0.262  0.94125  0.82975  …  0.3845  0.891875  0.563625
```

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);
```

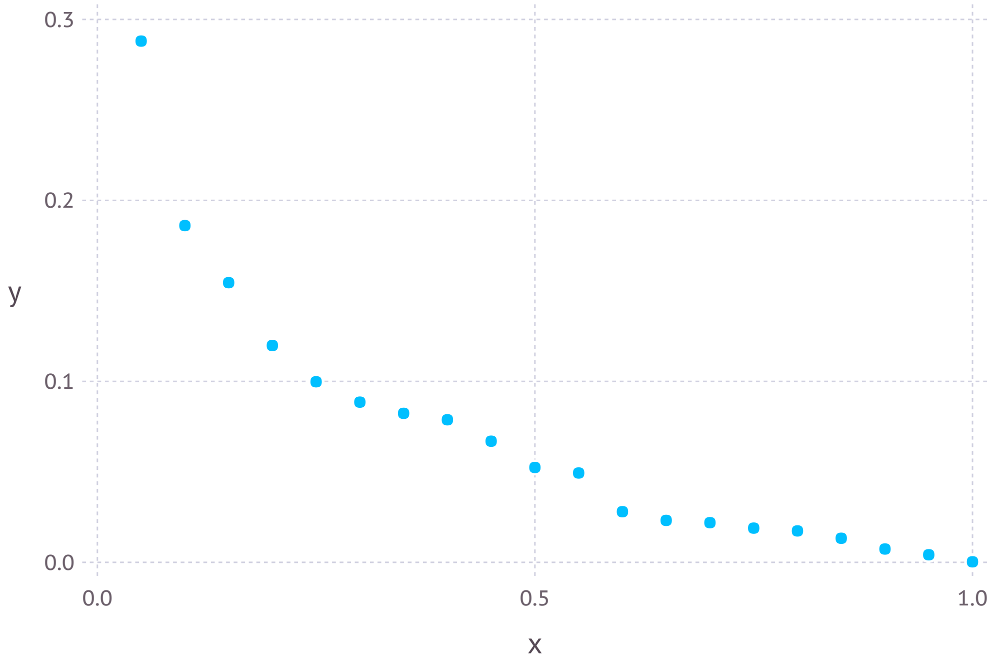
In [23]:
```
for i=1:(nRows-20)
LDMat[i,:] = corMat[i,(i+1):(i+20)].^2
end
```

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

Out[24]:
```
1x20 Array{Float64,2}:
 0.000300718  0.00419619  0.00738866  …  0.154544  0.186101  0.288048
```

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]:
```
8.683996773457539
```

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

Out[31]:
```
0.9121607757459183
```

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

Out[32]:
```
0.9121607757459183
```

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

Out[33]:
```
0.9121607757459183
```

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]: 11.556486447840987

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

Out[36]: 11.56425113738653

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

Out[37]: 0.5584818632984035

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

Out[38]: 0.5558957570875588
```

Pedigree: All animals

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

Out[39]: 48000x3 Array{Int64,2}:
          40722  36206  40095
          40723  33838  39667
          40724  33469  38263
          40725  33442  36795
          40726  35605  39103
          40727  35520  37061
          40728  34597  37364
          40729  36237  37016
          40730  34840  39869
          40731  35149  40360
          40732  33880  37542
          40733  36471  38813
          40734  32990  39891
              ⋮
          88710  74162  77298
          88711  75704  80449
          88712  75552  79054
          88713  76253  77258
          88714  73085  79624
          88715  75628  80311
          88716  75359  77456
          88717  75333  80608
          88718  76570  76925
          88719  74530  80250
          88720  76426  80699
          88721  75333  79652
```

```
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  …  2  1  2  1  2  2  0  1  1  1  1  1
  40723  0  2  0  2  2  0  0  0  2  0     1  0  1  2  0  1  1  0  0  1  2  2
  40724  0  2  0  2  2  0  0  0  2  0     1  1  2  1  1  1  1  1  1  1  2  1
  40725  0  1  1  1  1  0  1  1  1  1     2  2  2  0  2  2  0  2  2  0  2  0
  40726  0  2  0  2  2  1  1  1  1  1     1  1  2  1  1  1  1  1  1  2  2  2
  40727  0  1  1  1  1  0  1  1  1  1  …  2  2  1  1  2  2  0  1  1  0  1  1
  40728  0  2  0  2  2  1  1  1  1  1     1  1  2  1  1  1  1  1  1  1  2  1
  40729  0  1  1  1  1  0  1  1  1  1     1  1  2  1  1  1  1  1  1  1  2  1
  40730  0  2  0  2  2  0  1  1  2  0     1  1  1  2  1  1  1  0  0  1  2  2
  40731  0  2  0  2  2  0  0  0  2  0     2  2  1  2  2  2  0  0  0  0  2  1
  40732  0  2  1  2  1  0  0  0  2  0  …  2  2  2  1  2  2  0  1  1  1  2  0
  40733  0  2  0  2  2  0  0  0  2  0     2  1  2  1  2  1  1  1  1  1  2  1
  40734  0  2  0  2  2  0  0  0  2  0     1  1  1  2  2  2  0  1  1  0  2  1
      ⋮              ⋮              ⋮  ⋱        ⋮              ⋮              ⋮
  88710  0  2  0  2  2  0  1  1  1  1     2  2  2  0  2  2  0  2  2  0  2  0
  88711  0  1  1  2  1  1  1  1  1  1     2  2  2  1  2  2  0  1  1  0  2  1
  88712  0  0  2  2  2  0  2  2  0  2  …  2  2  2  0  2  2  0  2  2  0  2  0
  88713  0  1  1  2  2  1  2  2  0  2     2  2  2  0  2  2  0  2  2  0  2  0
  88714  0  0  2  1  1  1  2  2  0  2     2  2  2  1  2  2  0  1  1  0  2  1
  88715  0  0  2  2  2  2  2  2  0  2     2  2  2  1  2  2  0  1  1  1  2  0
  88716  0  1  1  1  1  1  2  2  0  2     2  2  2  0  2  2  0  2  2  0  2  0
  88717  1  1  1  1  1  0  2  2  0  2  …  1  1  2  1  2  2  0  2  2  0  2  0
  88718  0  2  0  2  2  1  2  2  0  2     2  2  2  0  2  2  0  2  2  0  2  0
  88719  0  1  1  2  2  2  2  2  0  2     2  2  2  0  2  2  0  2  2  0  2  0
  88720  0  1  1  2  2  1  2  2  0  2     2  2  2  1  2  2  0  1  1  1  2  0
  88721  0  0  2  1  1  0  2  2  0  2     2  2  2  1  2  2  0  1  1  1  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  0  2  2  0  0  0  2  0  0  2  2  …  2  1  2  1  2  2  0  1  1  1  1  1
 0  2  0  2  2  0  0  0  2  0  0  2  2     1  0  1  2  0  1  1  0  0  1  2  2
 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  1  1  1  1  0  1  1  1  1  1  0  1     2  2  2  0  2  2  0  2  2  0  2  0
 0  2  0  2  2  1  1  1  1  1  1  1  2     1  1  2  1  1  1  1  1  1  2  2  2
 0  1  1  1  1  0  1  1  1  1  1  1  2  …  2  2  1  1  2  2  0  1  1  0  1  1
 0  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  1  1  1  1  0  1  1  1  1  1  1  2     1  1  2  1  1  1  1  1  1  1  2  1
 0  2  0  2  2  0  1  1  2  0  0  1  1     1  1  1  2  1  1  1  0  0  1  2  2
 0  2  0  2  2  0  0  0  2  0  0  1  1     2  2  1  2  2  2  0  0  0  0  2  1
 0  2  1  2  1  0  0  0  2  0  0  2  1  …  2  2  2  1  2  2  0  1  1  1  2  0
 0  2  0  2  2  0  0  0  2  0  0  1  1     2  1  2  1  2  1  1  1  1  1  2  1
 0  2  0  2  2  0  0  0  2  0  0  0  0     1  1  1  2  2  2  0  1  1  0  2  1
 ⋮              ⋮              ⋮        ⋱        ⋮              ⋮
 0  2  0  2  2  0  1  1  1  1  1  1  1     2  2  2  0  2  2  0  2  2  0  2  0
 0  1  1  2  1  1  1  1  1  1  1  1  1     2  2  2  1  2  2  0  1  1  0  2  1
 0  0  2  2  2  0  2  2  0  2  2  0  0  …  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     2  2  2  0  2  2  0  2  2  0  2  0
 0  0  2  1  1  1  2  2  0  2  2  0  2     2  2  2  1  2  2  0  1  1  0  2  1
 0  0  2  2  2  2  2  2  0  2  2  0  2     2  2  2  1  2  2  0  1  1  1  2  0
 0  1  1  1  1  1  2  2  0  2  2  0  2     2  2  2  0  2  2  0  2  2  0  2  0
 1  1  1  1  1  0  2  2  0  2  2  0  2  …  1  1  2  1  2  2  0  2  2  0  2  0
 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  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  1  1  2  2  1  2  2  0  2  2  0  1     2  2  2  1  2  2  0  1  1  1  2  0
 0  0  2  1  1  0  2  2  0  2  2  0  2     2  2  2  1  2  2  0  1  1  1  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}:
          41418
          42716
          41986
          41954
          43403
          41478
          43709
          43399
          41168
          41045
          43619
          44244
          44484
              ⋮
          74162
          75704
          75552
          76253
          73085
          75628
          75359
          75333
          76570
          74530
          76426
          75333
```

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

```
Out[53]: 1000-element Array{Int64,1}:
          41418
          42716
          41986
          41954
          43403
          41478
          43709
          43399
          41168
          41045
          43619
          44244
          44484
              ⋮
          75333
          75666
          74772
          75911
          74882
          73937
          73407
          75783
          75148
          74211
          75739
          74092
```

```
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}:
          41418
          42716
          41986
          41954
          43403
          41478
          43709
          43399
          41168
          41045
          43619
          44244
          44484
              ⋮
          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  10.943   9.623
          40723   3.965   6.591
          40724   7.023   8.197
          40725   9.289  10.205
          40726   8.82    8.384
          40727   7.939   8.996
          40728   9.183   9.008
          40729   9.097   9.408
          40730   6.986   7.995
          40731   8.824   9.589
          40732   9.04    9.804
          40733   7.296   8.203
          40734   7.192   7.189
              ⋮
          88710   9.909  11.991
          88711  12.485  12.398
          88712  12.853  12.194
          88713  12.544  12.79
          88714  13.63   12.206
          88715  12.41   12.396
          88716  11.941  11.795
          88717  10.933  11.392
          88718  12.356  11.378
          88719  13.162  13.391
          88720  12.968  11.993
          88721  12.587  11.403
```

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

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

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

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

Phenotypes - all animnls from G0 to G4

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

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

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

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

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

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

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

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

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

Out[77]: 40000
```

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

Out[78]: 3
```

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

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

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

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

Get files with QTL only or Markers only

QTL file and Markers file

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

```
Out[82]: 50-element Array{Int64,1}:
            3
            6
            8
           10
           13
           23
           26
           28
           30
           33
           43
           46
           48
            ⋮
          150
          153
          163
          166
          168
          170
          173
          183
          186
          188
          190
          193
```

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

```
Out[83]: 150-element Array{Int64,1}:
            1
            2
            4
            5
            7
            9
           11
           12
           14
           15
           16
           17
           18
            ⋮
          185
          187
          189
          191
          192
          194
          195
          196
          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  1  1  1  1  0  1  1  1  1  1  0  1  …  2  2  2  0  2  2  0  2  2  0  2  0
          0  2  0  2  2  0  1  1  1  1  1  0  1     2  2  2  0  2  2  0  2  2  0  2  0
          1  1  1  2  2  1  2  2  0  2  2  0  1     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  0  2  2  0  2  2  0  2  0
          1  2  0  2  2  1  1  1  1  1  1  1  2     2  2  2  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  2  2  0  0  0  1  1  2
          1  2  0  2  2  1  1  1  1  1  1  1  2     2  2  2  0  2  2  0  2  2  0  2  0
          1  1  1  1  1  0  1  1  1  1  1  1  2     1  1  2  1  1  1  1  1  1  1  2  1
          0  1  1  1  1  0  1  1  1  1  1  1  2     2  1  1  1  1  2  0  1  1  0  2  1
          0  2  0  2  2  1  1  1  1  1  1  1  2     2  2  2  1  2  2  0  1  1  1  2  1
          0  1  1  2  2  1  2  2  0  2  2  0  2  …  2  1  1  1  1  2  0  1  1  0  2  1
          0  1  1  2  2  0  1  1  1  1  1  1  2     2  1  1  1  2  2  0  1  1  1  2  0
          1  2  0  2  2  0  1  1  1  1  1  0  0     2  2  2  0  2  2  0  2  2  0  2  0
          ⋮              ⋮              ⋮        ⋱        ⋮              ⋮
          0  2  0  2  2  0  1  1  1  1  1  1  1     2  2  2  0  2  2  0  2  2  0  2  0
          0  1  1  2  1  1  1  1  1  1  1  1  1     2  2  2  1  2  2  0  1  1  0  2  1
          0  0  2  2  2  0  2  2  0  2  2  0  0  …  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     2  2  2  0  2  2  0  2  2  0  2  0
          0  0  2  1  1  1  2  2  0  2  2  0  2     2  2  2  1  2  2  0  1  1  0  2  1
          0  0  2  2  2  2  2  2  0  2  2  0  2     2  2  2  1  2  2  0  1  1  1  2  0
          0  1  1  1  1  1  2  2  0  2  2  0  2     2  2  2  0  2  2  0  2  2  0  2  0
          1  1  1  1  1  0  2  2  0  2  2  0  2  …  1  1  2  1  2  2  0  2  2  0  2  0
          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  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  1  1  2  2  1  2  2  0  2  2  0  1     2  2  2  1  2  2  0  1  1  1  2  0
          0  0  2  1  1  0  2  2  0  2  2  0  2     2  2  2  1  2  2  0  1  1  1  2  0
```

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

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

In [87]:
```
onlyMar = QTLMarker[:,MarkerPos];
```

In [88]:
```
QTLstream = open(QTL, "w")
Marstream = open(Mar, "w");
```

In [89]:
```
for i in 1:size(onlyID,1)
    @printf(QTLstream, "%19d", onlyID[i])
    for j in 1:size(onlyQTL,2)
        @printf(QTLstream, "%3d", onlyQTL[i,j])
    end
    @printf(QTLstream, "\n")
end
```

In [90]:
```
for i in 1:size(onlyID,1)
    @printf(Marstream, "%19d", onlyID[i])
    for j in 1:size(onlyMar,2)
        @printf(Marstream, "%3d", onlyMar[i,j])
    end
    @printf(Marstream, "\n")
end
```

In [91]:
```
close(QTLstream)
close(Marstream)
```

Remove Fixed Gene from the panel

In [92]:
```
VQM = var(QTLMarker,1)
QMnoFixed = QTLMarker[:,VQM .> 0]
VQ = var(onlyQTL,1)
QnoFixed = onlyQTL[:,VQ .> 0]
VM = var(onlyMar,1)
MnoFixed = onlyMar[:,VM .> 0];
```

In [93]:
```
GenNFstream = open(GenNF, "w")
QTLNFstream = open(QTLNF, "w")
MarNFstream = open(MarNF, "w");
```

In [94]:
```
for i in 1:size(onlyID,1)
    @printf(GenNFstream, "%19d", onlyID[i])
    for j in 1:size(QMnoFixed,2)
        @printf(GenNFstream, "%3d", QMnoFixed[i,j])
    end
    @printf(GenNFstream, "\n")
end
```

In [95]:
```
for i in 1:size(onlyID,1)
    @printf(QTLNFstream, "%19d", onlyID[i])
    for j in 1:size(QnoFixed,2)
        @printf(QTLNFstream, "%3d", QnoFixed[i,j])
    end
    @printf(QTLNFstream, "\n")
end
```

In [96]:
```
for i in 1:size(onlyID,1)
    @printf(MarNFstream, "%19d", onlyID[i])
    for j in 1:size(MnoFixed,2)
        @printf(MarNFstream, "%3d", MnoFixed[i,j])
    end
    @printf(MarNFstream, "\n")
end
```

In [97]:
```
close(GenNFstream)
close(QTLNFstream)
close(MarNFstream)
```

Check heritability

In [98]:
```
P = AllPBV[:,2]
BV = AllPBV[:,3];
```

In [99]:
```
VP = var(P)
VBV = var(BV)
H = VBV/VP
```

Out[99]:
```
0.6450410406836294
```

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

          WARNING: imported binding for cor overwritten in module Main

Out[100]: 0.7991910626042393
```

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

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

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

Out[102]: 50-element Array{Float64,1}:
           0.200393
           0.20201
           0.202706
           0.194417
           0.199839
           0.199925
           0.198409
           0.204593
           0.199903
           0.200427
           0.201274
           0.197668
           0.198378
           ⋮
           0.198434
           0.199725
           0.202711
           0.201648
           0.200264
           0.19919
           0.19926
           0.201289
           0.19734
           0.200444
           0.197966
           0.197362
```

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

Out[103]: 48000-element Array{Float64,1}:
            9.58996
            6.58712
            8.1855
           10.1833
            8.37918
            8.97992
            8.98935
            9.39095
            7.99042
            9.56793
            9.77765
            8.18611
            7.18064
            ⋮
           11.9768
           12.3723
           12.1619
           12.7834
           12.1792
           12.3681
           11.7846
           11.3687
           11.3748
           13.3827
           11.9686
           11.3728

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

Out[104]: 8.668180163821793

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

Out[105]: 9.365385835186368

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

Out[106]: 9.963291791206261
```

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

Out[107]: 10.506639154784093

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

Out[108]: 11.04175254802033

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

Out[109]: 11.538781685507601

In [110]: EAlphaG=onlyQTL*QTLo

Out[110]: 9000-element Array{Float64,1}:
            9.58792
            9.38778
           11.1604
           11.3696
           10.1979
            8.98614
           10.7746
            9.79783
           10.9856
           10.1843
            9.38266
            9.77817
           10.5719
            ⋮
           11.9768
           12.3723
           12.1619
           12.7834
           12.1792
           12.3681
           11.7846
           11.3687
           11.3748
           13.3827
           11.9686
           11.3728
```

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

Out[111]: 11.48388884357604

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

Out[112]: 2.8157086797542465

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

Out[113]: 10.020647113966255

In [114]: meanBreedingValueSelectedG0=meanEAlphaS0-meanEAlphaG0

Out[114]: 1.3524669501444624

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

Out[115]: 10.552256161131329

In [116]: meanBreedingValueSelectedG1=meanEAlphaS1-meanEAlphaG0

Out[116]: 1.8840759973095356

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

Out[117]: 11.055418453929915

In [118]: meanBreedingValueSelectedG2=meanEAlphaS2-meanEAlphaG0

Out[118]: 2.3872382901081224

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

Out[119]: 11.575150983835087

In [120]: meanBreedingValueSelectedG3=meanEAlphaS3-meanEAlphaG0

Out[120]: 2.906970820013294
```

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

Out[121]: 12.020257827755096

In [122]: meanBreedingValueSelectedG4=meanEAlphaS4-meanEAlphaG0

Out[122]: 3.3520776639333025

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

Out[123]: 11.538781685507601

In [124]: meanBreedingValueSelectedG5=meanEAlphaS5-meanEAlphaG0

Out[124]: 2.870601521685808
```