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)
```

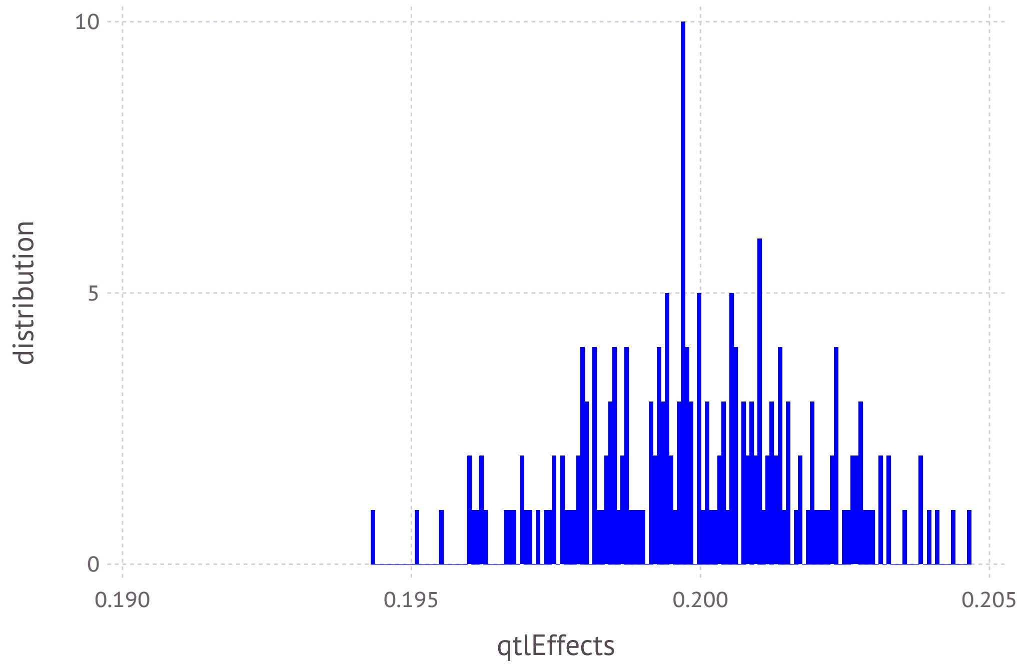
In [7]:
```
qtlEffects
```

Out[7]:
```
200-element Array{Float64,1}:
 0.202904
 0.200146
 0.198487
 0.201699
 0.199154
 0.201035
 0.197696
 0.199831
 0.197617
 0.198194
 0.19997
 0.203795
 0.199167
 ⋮
 0.1982
 0.197491
 0.200634
 0.198725
 0.199179
 0.200822
 0.198454
 0.197811
 0.200519
 0.199999
 0.20228
 0.199612
```

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]: 0.19994511441282548
```

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

```
Out[11]: 3.92219980652525e-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
```

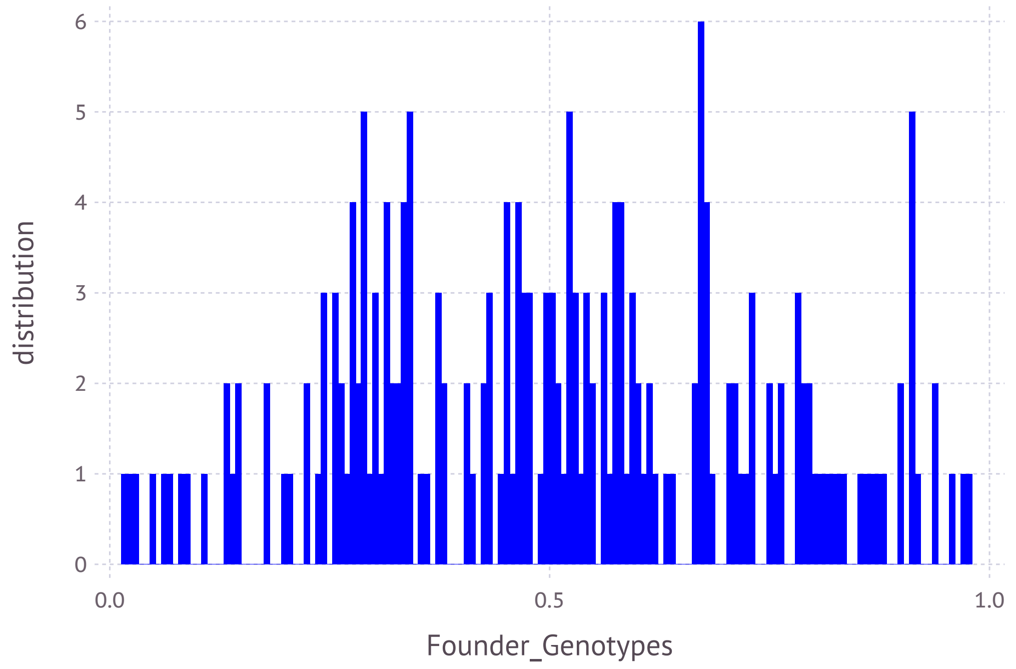
```
In [19]: gSPSire = XSim.getOurGenotypes(popSP[1])
         gSPDam = XSim.getOurGenotypes(popSP[2])
         gSP = [gSPSire;gSPSire];
```

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

```
Out[20]: 1x200 Array{Float64,2}:
          0.06325  0.82675  0.306  0.93975  0.8065  …  0.376375  0.897625  0.547875
```

```
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);
```

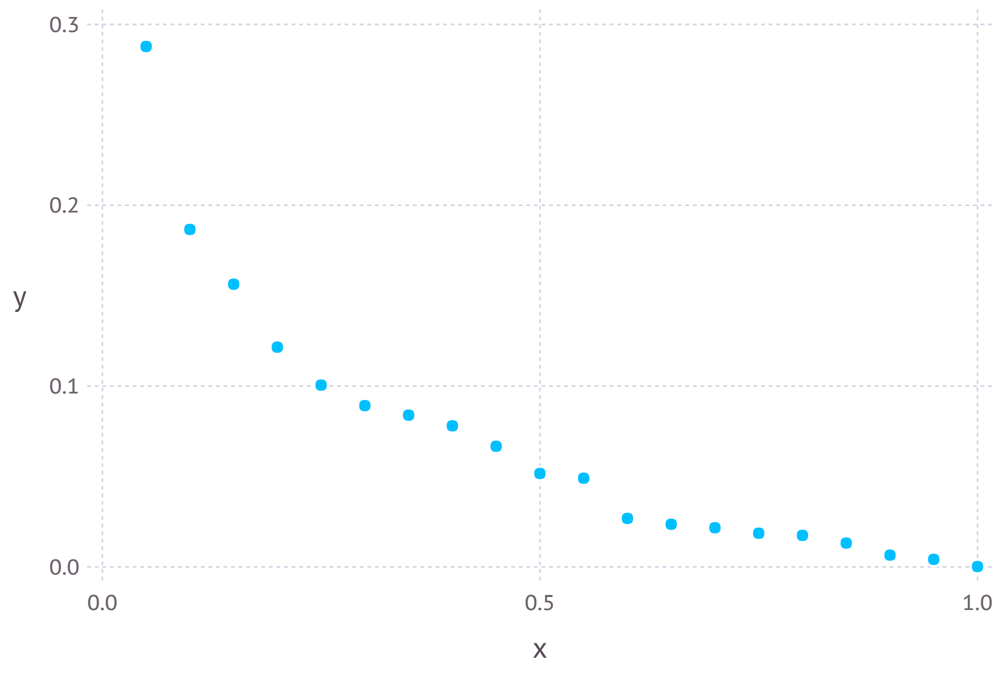
In [23]:
```
for i=1:(nRows-20)
LDMat[i,:] = corMat[i,(i+1):(i+20)].^2
end
```

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

Out[24]:
```
1x20 Array{Float64,2}:
 0.000237789  0.00421084  0.00655045  …  0.156335  0.186622  0.287791
```

In [25]:
```
plot(x=(1:20)/20*1,y=y)
```

Out[25]:



In [26]:
```
FCMstream = open("SNPCMF.txt", "w")
```

Out[26]:
```
IOStream(<file SNPCMF.txt>)
```

In [27]:
```
for i in 1:size(FCM,2)
    @printf(FCMstream, "%6.4f ", FCM[1,i])
end
```

In [28]:
```
close(FCMstream)
```

Selection - increase

In [29]:
```
aSPSire = XSim.getOurGenVals(popSP[1])
aSPDam = XSim.getOurGenVals(popSP[2])
aSP = [aSPSire;aSPDam];
```

In [30]:
```
mean(aSP)
```

Out[30]:
```
10.3145370140333
```

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

Out[31]:
```
0.7469294079345367
```

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

Out[32]:
```
1.7428352851805855
```

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

Out[33]:
```
1.7428352851805855
```

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

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

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

Out[35]: 12.345432221539614

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

Out[36]: 12.34954886729238

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

Out[37]: 0.5544677183148566

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

Out[38]: 0.5301630644831103
```

Pedigree: All animals

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

Out[39]: 48000x3 Array{Int64,2}:
          40722  33792  39233
          40723  35731  38071
          40724  35083  38882
          40725  33235  38814
          40726  32735  37452
          40727  36599  38080
          40728  34315  40279
          40729  32869  40185
          40730  34003  37951
          40731  35299  39652
          40732  34350  40353
          40733  35337  40573
          40734  34703  38089
              ⋮
          88710  73139  79490
          88711  74237  78732
          88712  73609  79596
          88713  75642  78203
          88714  75498  79484
          88715  75854  79893
          88716  74910  77341
          88717  75484  77758
          88718  76074  80716
          88719  75346  76952
          88720  74812  79078
          88721  75203  78889
```

```
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  1  2  0  2  2  1  1  1  1  1  …  2  1  1  1  1  2  0  1  1  0  2  1
          40723  0  1  1  2  2  0  1  1  1  1     2  1  0  2  0  0  2  1  1  1  2  2
          40724  0  1  1  2  1  1  1  1  1  1     2  1  1  1  2  1  1  1  1  0  2  1
          40725  0  1  1  1  1  0  1  1  1  1     1  0  1  2  0  1  1  0  0  1  2  2
          40726  0  2  1  2  1  0  0  0  2  0     1  0  2  2  0  0  1  0  0  1  2  2
          40727  0  2  0  2  2  1  1  1  1  1  …  2  0  0  2  0  2  0  0  0  0  2  2
          40728  1  2  1  2  1  1  1  1  1  1     2  2  2  0  2  2  0  2  2  0  2  0
          40729  0  1  1  2  2  1  1  1  1  1     0  0  2  2  0  0  2  0  0  2  2  2
          40730  0  2  1  2  1  0  0  0  2  0     2  1  1  1  0  2  0  0  0  0  2  2
          40731  0  2  1  2  1  0  0  0  2  0     1  0  2  2  1  1  1  0  0  1  2  2
          40732  0  1  1  2  2  0  1  1  1  1  …  2  2  2  0  2  2  0  2  2  0  2  0
          40733  0  1  1  1  2  0  1  1  1  1     2  1  1  1  1  2  0  1  1  0  2  1
          40734  0  2  0  2  2  0  1  1  2  0     2  2  2  1  2  2  0  1  1  0  2  0
              ⋮              ⋮              ⋮  ⋱     ⋮              ⋮              ⋮
          88710  0  2  1  2  1  0  0  0  2  0     2  1  2  1  2  2  0  1  1  1  1  1
          88711  0  2  0  2  2  0  0  0  2  0     2  1  2  1  1  2  1  1  1  1  1  2
          88712  0  2  0  2  2  1  2  2  1  1  …  1  1  2  1  2  2  0  2  2  0  2  0
          88713  0  2  0  2  2  0  1  1  2  0     2  1  1  1  1  1  1  1  1  1  2  1
          88714  1  2  0  2  2  1  1  1  1  1     2  2  2  0  2  2  0  2  2  0  2  0
          88715  0  2  0  2  2  0  0  0  2  0     2  2  1  1  2  2  0  1  1  1  1  1
          88716  0  2  0  2  2  0  1  1  1  1     2  2  2  0  2  2  0  2  2  0  2  0
          88717  0  2  0  2  2  0  0  0  2  0  …  2  0  1  2  1  1  0  0  0  1  2  1
          88718  1  1  1  2  2  2  2  2  0  2     2  1  2  1  2  2  0  0  0  1  0  2
          88719  1  1  1  2  2  2  2  2  0  2     2  2  1  1  1  1  1  1  1  1  2  1
          88720  1  2  0  2  2  1  0  1  2  0     2  2  2  0  2  2  0  2  2  0  2  1
          88721  1  2  0  2  2  0  0  0  2  0     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}:
 1  2  0  2  2  1  1  1  1  1  1  1  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  0     2  1  0  2  0  0  2  1  1  1  2  2
 0  1  1  2  1  1  1  1  1  1  1  1  1     2  1  1  1  2  1  1  1  1  0  2  1
 0  1  1  1  1  0  1  1  1  1  1  0  1     1  0  1  2  0  1  1  0  0  1  2  2
 0  2  1  2  1  0  0  0  2  0  0  1  0     1  0  2  2  0  0  1  0  0  1  2  2
 0  2  0  2  2  1  1  1  1  1  0  1  2  …  2  0  0  2  0  2  0  0  0  0  2  2
 1  2  1  2  1  1  1  1  1  1  1  1  1     2  2  2  0  2  2  0  2  2  0  2  0
 0  1  1  2  2  1  1  1  1  1  1  0  1     0  0  2  2  0  0  2  0  0  2  2  2
 0  2  1  2  1  0  0  0  2  0  0  1  0     2  1  1  1  0  2  0  0  0  0  2  2
 0  2  1  2  1  0  0  0  2  0  0  2  1     1  0  2  2  1  1  1  0  0  1  2  2
 0  1  1  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  1  2  0  1  1  1  1  1  0  0     2  1  1  1  1  2  0  1  1  0  2  1
 0  2  0  2  2  0  1  1  2  0  0  2  2     2  2  2  1  2  2  0  1  1  0  2  0
 ⋮              ⋮              ⋮        ⋱        ⋮              ⋮
 0  2  1  2  1  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     2  1  2  1  1  2  1  1  1  1  1  2
 0  2  0  2  2  1  2  2  1  1  1  1  2  …  1  1  2  1  2  2  0  2  2  0  2  0
 0  2  0  2  2  0  1  1  2  0  0  2  2     2  1  1  1  1  1  1  1  1  1  2  1
 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
 0  2  0  2  2  0  0  0  2  0  0  2  2     2  2  1  1  2  2  0  1  1  1  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  0  0  0  2  0  0  1  1  …  2  0  1  2  1  1  0  0  0  1  2  1
 1  1  1  2  2  2  2  2  0  2  2  0  2     2  1  2  1  2  2  0  0  0  1  0  2
 1  1  1  2  2  2  2  2  0  2  2  0  2     2  2  1  1  1  1  1  1  1  1  2  1
 1  2  0  2  2  1  0  1  2  0  0  1  1     2  2  2  0  2  2  0  2  2  0  2  1
 1  2  0  2  2  0  0  0  2  0  0  2  1     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}:
          41182
          41827
          40786
          43901
          43285
          43614
          42727
          43130
          41673
          43285
          43359
          43298
          42278
              ⋮
          73139
          74237
          73609
          75642
          75498
          75854
          74910
          75484
          76074
          75346
          74812
          75203
```

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

```
Out[53]: 1000-element Array{Int64,1}:
          41182
          41827
          40786
          43901
          43285
          43614
          42727
          43130
          41673
          43359
          43298
          42278
          44224
              ⋮
          73142
          75404
          73643
          75334
          76454
          75174
          73809
          75643
          74792
          75168
          73163
          73893
```

```
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}:
          41182
          41827
          40786
          43901
          43285
          43614
          42727
          43130
          41673
          43359
          43298
          42278
          44224
              ⋮
          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  11.55   11.417
          40723   9.034   8.99
          40724   8.97    9.988
          40725   8.825   9.0
          40726  11.794  10.203
          40727   9.705   9.782
          40728  11.611  10.618
          40729  10.248  10.003
          40730   9.965   9.588
          40731   8.014   9.792
          40732  11.177   9.003
          40733   9.334  10.597
          40734   9.279  10.997
              ⋮
          88710  10.451  10.999
          88711  11.936  12.803
          88712  11.318  12.402
          88713  13.675  13.392
          88714  13.322  14.004
          88715  12.182  13.599
          88716  12.801  13.192
          88717  13.049  12.995
          88718  12.744  13.002
          88719  11.044  12.405
          88720  12.792  12.606
          88721  12.532  11.987
```

```
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
            5
            6
            9
           13
           21
           25
           26
           29
           33
           41
           45
           46
            ⋮
          149
          153
          161
          165
          166
          169
          173
          181
          185
          186
          189
          193
```

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

```
Out[83]: 150-element Array{Int64,1}:
            2
            3
            4
            7
            8
           10
           11
           12
           14
           15
           16
           17
           18
            ⋮
          187
          188
          190
          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}:
 1  2  0  2  2  1  0  1  2  0  0  1  1  …  1  1  2  2  1  1  1  0  0  2  2  2
 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  1  2  1  0  0  0  2  0  0  2  0     2  1  1  1  2  2  0  1  1  0  2  1
 0  2  0  2  2  0  0  0  2  0  0  1  1     2  0  1  2  0  2  0  0  0  0  1  2
 0  2  1  2  1  0  0  0  2  0  0  2  1     2  1  1  1  2  2  0  1  1  1  2  0
 1  2  0  2  2  0  0  0  2  0  0  2  2  …  1  2  1  2  0  1  0  0  0  1  0  2
 0  2  1  2  2  1  1  1  2  0  0  0  0     2  0  0  2  0  2  0  0  0  0  2  2
 1  2  0  2  2  1  1  1  1  1  1  0  1     1  1  2  1  1  1  1  1  1  1  2  1
 0  2  0  2  2  0  0  0  2  0  0  0  0     2  2  1  1  2  2  0  1  1  0  2  1
 0  1  1  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  2  2  …  1  1  2  1  1  2  0  1  1  0  1  1
 0  2  0  2  2  1  1  1  1  1  1  1  2     2  2  2  0  2  2  0  2  2  0  2  0
 2  2  0  2  2  0  2  2  0  2  2  0  0     2  1  1  1  1  2  1  1  1  1  1  1
 ⋮              ⋮              ⋮        ⋱        ⋮              ⋮
 0  2  1  2  1  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     2  1  2  1  1  2  1  1  1  1  1  2
 0  2  0  2  2  1  2  2  1  1  1  1  2  …  1  1  2  1  2  2  0  2  2  0  2  0
 0  2  0  2  2  0  1  1  2  0  0  2  2     2  1  1  1  1  1  1  1  1  1  2  1
 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
 0  2  0  2  2  0  0  0  2  0  0  2  2     2  2  1  1  2  2  0  1  1  1  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  0  0  0  2  0  0  1  1  …  2  0  1  2  1  1  0  0  0  1  2  1
 1  1  1  2  2  2  2  2  0  2  2  0  2     2  1  2  1  2  2  0  0  0  1  0  2
 1  1  1  2  2  2  2  2  0  2  2  0  2     2  2  1  1  1  1  1  1  1  1  2  1
 1  2  0  2  2  1  0  1  2  0  0  1  1     2  2  2  0  2  2  0  2  2  0  2  1
 1  2  0  2  2  0  0  0  2  0  0  2  1     2  2  2  0  2  2  0  2  2  0  2  0
```

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

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

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

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

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

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

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

Remove Fixed Gene from the panel

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

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

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

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

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

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

Check heritability

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

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

Out[99]:
```
0.4018686462290748
```

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

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

Out[100]:
```
0.6308596167172458
```

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

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

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

```
Out[102]: 50-element Array{Float64,1}:
           0.202904
           0.199154
           0.201035
           0.197617
           0.199167
           0.201343
           0.197355
           0.201018
           0.196654
           0.198454
           0.201713
           0.203092
           0.201489
           ⋮
           0.201481
           0.199255
           0.199777
           0.198016
           0.20154
           0.198713
           0.203504
           0.199967
           0.201945
           0.199247
           0.1982
           0.199179
```

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

```
Out[103]: 48000-element Array{Float64,1}:
           11.3817
            9.0016
            9.98578
            9.00298
           10.1928
            9.82232
           10.5896
           10.0063
            9.57884
            9.78069
            8.98769
           10.5761
           10.9748
            ⋮
           10.9807
           12.7641
           12.3774
           13.3942
           13.9878
           13.578
           13.1751
           12.9833
           12.9922
           12.3806
           12.5845
           11.995
```

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

```
Out[104]: 10.305993311946171
```

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

```
Out[105]: 10.857006534408065
```

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

```
Out[106]: 11.303139837414601
```

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

Out[107]: 11.639192777911681

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

Out[108]: 12.005831794656686

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

Out[109]: 12.334898682874332

In [110]: EAlphaG=onlyQTL*QTLo

Out[110]: 9000-element Array{Float64,1}:
           11.1861
           10.7981
           11.5741
           10.3896
           11.566
           10.5918
           10.8036
           11.5932
           10.7829
           11.7834
           10.5806
           11.1817
           11.3832
             ⋮
           10.9807
           12.7641
           12.3774
           13.3942
           13.9878
           13.578
           13.1751
           12.9833
           12.9922
           12.3806
           12.5845
           11.995
```

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

Out[111]: 12.301470928526989

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

Out[112]: 1.9954776165808177

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

Out[113]: 11.383790624887213

In [114]: meanBreedingValueSelectedG0=meanEAlphaS0-meanEAlphaG0

Out[114]: 1.0777973129410423

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

Out[115]: 11.775415196844676

In [116]: meanBreedingValueSelectedG1=meanEAlphaS1-meanEAlphaG0

Out[116]: 1.4694218848985052

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

Out[117]: 11.978245258154468

In [118]: meanBreedingValueSelectedG2=meanEAlphaS2-meanEAlphaG0

Out[118]: 1.6722519462082968

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

Out[119]: 12.362939424658542

In [120]: meanBreedingValueSelectedG3=meanEAlphaS3-meanEAlphaG0

Out[120]: 2.056946112712371
```

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

Out[121]: 12.669853964196106

In [122]: meanBreedingValueSelectedG4=meanEAlphaS4-meanEAlphaG0

Out[122]: 2.363860652249935

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

Out[123]: 12.334898682874332

In [124]: meanBreedingValueSelectedG5=meanEAlphaS5-meanEAlphaG0

Out[124]: 2.028905370928161
```