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 = 0.0
# 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

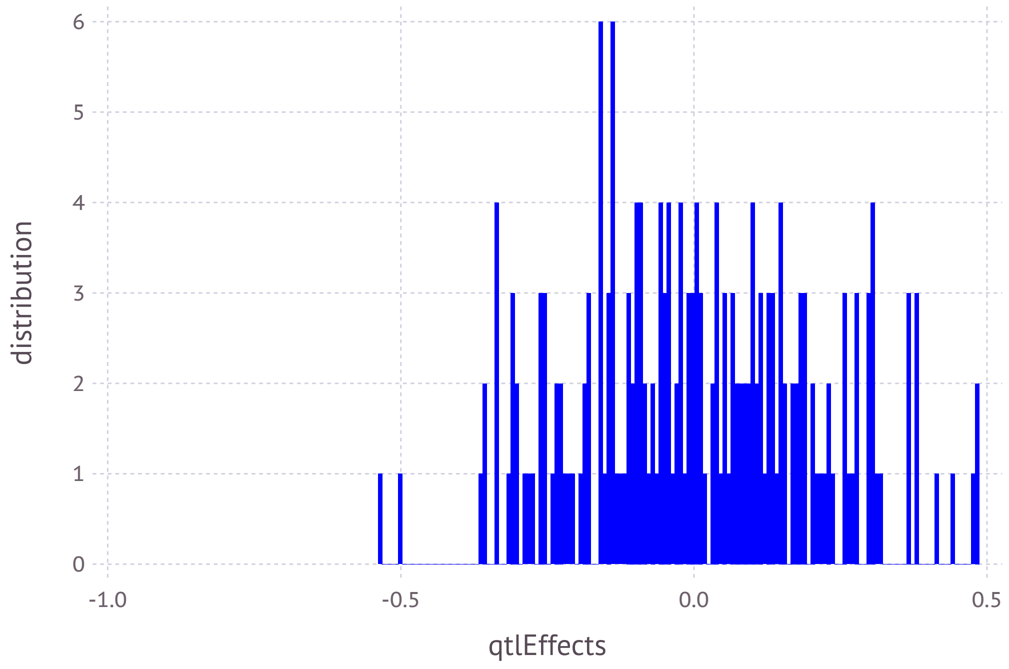
In [6]:
```
numChr    = 10
chrLength = 0.01
numLoci   = 20
nQTL      = 5
mutRate   = 0.0
locusInt  = chrLength/numLoci
mapPos    = collect(locusInt/2:locusInt:chrLength)
geneFreq  = fill(0.5,numLoci)
QTL = sample(1:numLoci,nQTL,replace=false)
qtlMarker = fill(false,numLoci)
qtlMarker[QTL] = true
Va = nQTL*numChr*0.5              # Va= nQTL*2pq*mean(alpha)^2; mean(alpha) = 0
qtlEffects= rand(Normal(0, sqrt(1/Va)),numLoci*numChr)  # Let alpha mu = 0.0
XSim.init(numChr,numLoci,chrLength,geneFreq,mapPos,qtlMarker,qtlEffects,mutRate)
```

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

```
Out[7]: 200-element Array{Float64,1}:
         -0.278953
         -0.0952244
          0.0765928
          0.311483
         -0.276034
          0.0824478
         -0.300198
         -0.354575
         -0.102106
         -0.288012
          0.182434
         -0.309952
          0.44491
          ⋮
         -0.497707
          0.130605
         -0.037989
          0.0457109
          0.063841
          0.181908
          0.474199
         -0.00455869
         -0.538252
          0.133681
          0.084391
         -0.334398
```

```
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]: 0.00719001159225456
```

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

```
Out[11]: 0.04155395337535214
```

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
```

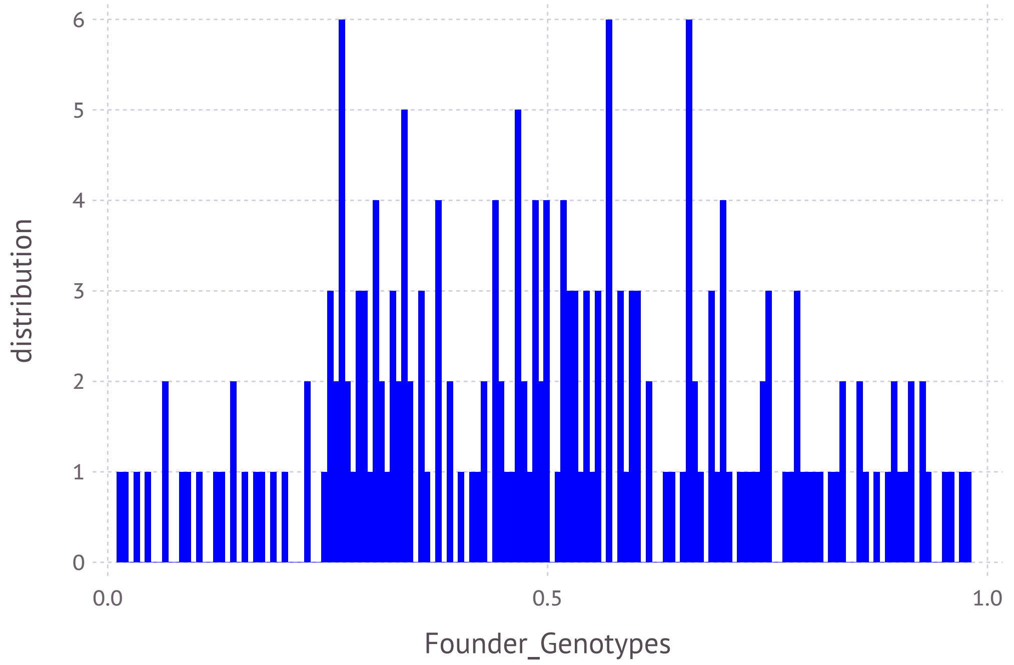
```
In [19]: gSPSire = XSim.getOurGenotypes(popSP[1])
         gSPDam = XSim.getOurGenotypes(popSP[2])
         gSP = [gSPSire;gSPSire];
```

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

```
Out[20]: 1x200 Array{Float64,2}:
          0.064125  0.8325  0.2855  0.950125  …  0.386125  0.895375  0.548375
```

```
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);
```

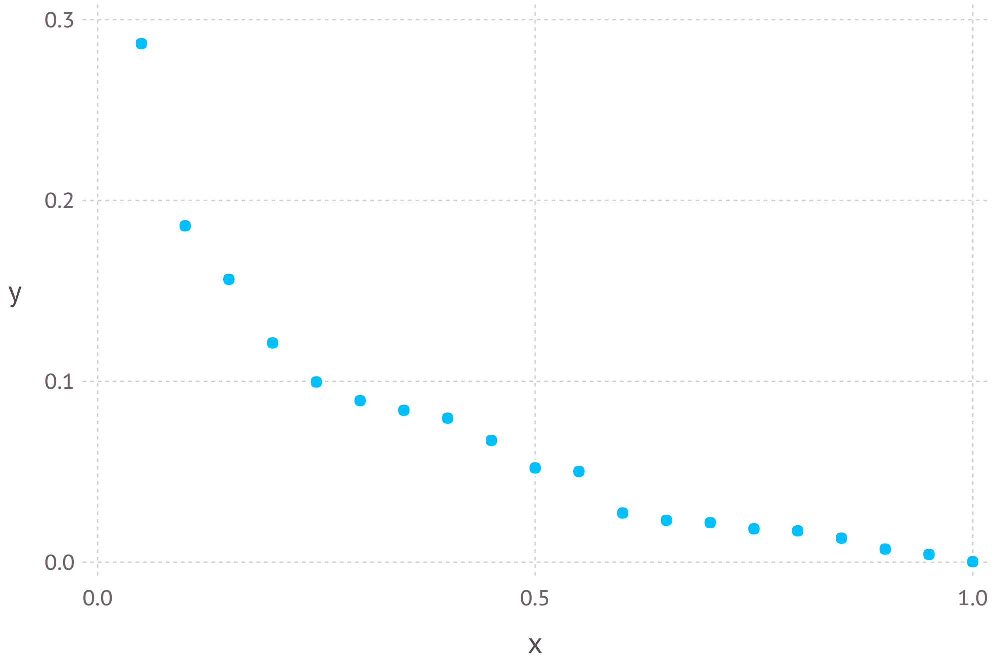
```
In [23]: for i=1:(nRows-20)
         LDMat[i,:] = corMat[i,(i+1):(i+20)].^2
         end
```

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

```
Out[24]: 1x20 Array{Float64,2}:
          0.000268249  0.0043338  0.00725586  …  0.121265  0.156414  0.186  0.286767
```

```
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]:
```
1.2427163323258397
```

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

Out[31]:
```
0.7387055814951132
```

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

Out[32]:
```
0.7387055814951132
```

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

Out[33]:
```
0.7387055814951132
```

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]: 3.9599136386841938

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

Out[36]: 3.952286325099084

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

Out[37]: 0.5362330591877899

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

Out[38]: 0.5438035626682531
```

Pedigree: All animals

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

```
Out[39]: 48000x3 Array{Int64,2}:
          40722  34932  38114
          40723  34746  37965
          40724  35635  40684
          40725  35443  37072
          40726  33151  39702
          40727  34103  37665
          40728  35362  39781
          40729  34747  38346
          40730  34244  37099
          40731  33634  38040
          40732  33725  40286
          40733  36260  37958
          40734  36274  38317
              ⋮
          88710  76521  80454
          88711  76470  78597
          88712  74301  79950
          88713  74686  79750
          88714  75780  80328
          88715  74904  80666
          88716  75544  79971
          88717  74210  77677
          88718  76061  80440
          88719  73843  78503
          88720  75133  79332
          88721  76514  79039
```

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

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

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

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

Create matrix of ``genotype" covariates for all animals

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

Out[43]: 48000

In [44]: nMarker = numChr*numLoci

Out[44]: 200

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

Out[45]: 48000x201 Array{Int64,2}:
          40722  0  2  0  2  2  0  0  0  2  0  …  1  1  2  1  1  1  1  1  1  1  2  1
          40723  0  1  1  2  2  1  1  1  1  1     2  2  2  0  2  2  0  2  2  0  2  0
          40724  0  2  1  2  1  0  0  0  2  0     2  2  2  0  2  2  0  2  2  0  2  0
          40725  0  1  1  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  2  1  1  1  1  1  1  1  2  1
          40727  0  2  0  2  2  0  0  0  2  0  …  2  1  1  1  2  2  0  1  1  1  2  0
          40728  0  2  0  2  2  0  0  0  2  0     0  0  2  2  0  0  2  0  0  2  2  2
          40729  0  2  0  2  2  0  0  0  2  0     1  1  2  1  1  1  0  1  1  1  1  1
          40730  0  2  0  2  2  1  1  1  1  1     1  1  2  1  0  1  1  0  0  1  2  2
          40731  0  1  1  2  2  1  1  1  1  1     2  0  2  2  1  0  0  0  0  0  2  2
          40732  0  2  1  2  2  1  1  1  2  0  …  2  0  2  1  1  1  0  1  1  0  2  1
          40733  0  1  2  2  1  1  1  1  1  1     1  1  2  1  1  1  1  1  1  1  2  1
          40734  0  1  1  2  2  1  1  1  1  1     1  1  2  1  1  1  1  1  1  1  2  1
              ⋮              ⋮              ⋮  ⋱     ⋮              ⋮              ⋮
          88710  0  2  0  2  2  0  0  0  2  0     2  0  1  2  0  1  0  0  0  0  2  2
          88711  0  1  1  2  2  1  1  1  1  1     2  0  0  2  0  2  0  0  0  0  2  2
          88712  0  2  0  2  2  0  0  0  2  0  …  1  0  1  2  1  1  2  1  1  1  2  2
          88713  1  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  0  0  0  2  0     0  1  2  2  0  1  1  1  1  1  2  2
          88715  1  2  0  2  2  0  0  0  2  0     1  1  2  2  1  1  1  0  0  2  2  2
          88716  0  2  0  2  2  0  0  0  2  0     1  0  1  2  0  1  1  0  0  1  2  2
          88717  1  2  0  2  2  0  0  0  2  0  …  2  1  1  1  2  2  0  1  1  1  2  0
          88718  0  2  0  2  1  0  0  0  2  0     0  0  2  2  0  0  2  0  0  2  2  2
          88719  0  2  0  2  2  0  0  0  2  0     1  0  1  2  0  1  1  0  0  2  0  2
          88720  0  2  0  2  2  0  0  0  2  0     1  0  2  2  0  0  1  0  0  1  2  2
          88721  0  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}:
          0  2  0  2  2  0  0  0  2  0  0  0  1  …  1  1  2  1  1  1  1  1  1  1  2  1
          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  2  1  2  1  0  0  0  2  0  0  1  0     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     2  1  1  1  1  2  0  1  1  0  2  1
          0  2  0  2  2  0  0  0  2  0  0  1  0     1  1  2  1  1  1  1  1  1  1  2  1
          0  2  0  2  2  0  0  0  2  0  0  1  1  …  2  1  1  1  2  2  0  1  1  1  2  0
          0  2  0  2  2  0  0  0  2  0  0  1  1     0  0  2  2  0  0  2  0  0  2  2  2
          0  2  0  2  2  0  0  0  2  0  0  1  1     1  1  2  1  1  1  0  1  1  1  1  1
          0  2  0  2  2  1  1  1  1  1  1  1  2     1  1  2  1  0  1  1  0  0  1  2  2
          0  1  1  2  2  1  1  1  1  1  1  1  2     2  0  2  2  1  0  0  0  0  0  2  2
          0  2  1  2  2  1  1  1  2  0  0  1  1  …  2  0  2  1  1  1  0  1  1  0  2  1
          0  1  2  2  1  1  1  1  1  1  1  1  1     1  1  2  1  1  1  1  1  1  1  2  1
          0  1  1  2  2  1  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  0  0  2  0  0  1  1     2  0  1  2  0  1  0  0  0  0  2  2
          0  1  1  2  2  1  1  1  1  1  1  0  1     2  0  0  2  0  2  0  0  0  0  2  2
          0  2  0  2  2  0  0  0  2  0  0  1  1  …  1  0  1  2  1  1  2  1  1  1  2  2
          1  2  0  2  2  0  1  1  1  1  1  1  1     2  2  2  0  2  2  0  2  2  0  2  0
          0  2  0  2  2  0  0  0  2  0  0  2  2     0  1  2  2  0  1  1  1  1  1  2  2
          1  2  0  2  2  0  0  0  2  0  0  2  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  2     1  0  1  2  0  1  1  0  0  1  2  2
          1  2  0  2  2  0  0  0  2  0  0  1  0  …  2  1  1  1  2  2  0  1  1  1  2  0
          0  2  0  2  1  0  0  0  2  0  0  2  1     0  0  2  2  0  0  2  0  0  2  2  2
          0  2  0  2  2  0  0  0  2  0  0  0  0     1  0  1  2  0  1  1  0  0  2  0  2
          0  2  0  2  2  0  0  0  2  0  0  2  2     1  0  2  2  0  0  1  0  0  1  2  2
          0  2  0  2  2  0  0  0  2  0  0  1  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}:
          40918
          44183
          43592
          41819
          40810
          43211
          43317
          43061
          41238
          44577
          43318
          41746
          41680
              ⋮
          76521
          76470
          74301
          74686
          75780
          74904
          75544
          74210
          76061
          73843
          75133
          76514
```

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

Out[53]: 1000-element Array{Int64,1}:
           40918
           44183
           43592
           41819
           40810
           43211
           43317
           43061
           41238
           44577
           43318
           41746
           41680
               ⋮
           76712
           75668
           75904
           76692
           76067
           73810
           74906
           76223
           74239
           75202
           75423
           75774
```

```
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}:
          40918
          44183
          43592
          41819
          40810
          43211
          43317
          43061
          41238
          44577
          43318
          41746
          41680
              ⋮
          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   2.317   2.25
          40723   0.042  -0.504
          40724  -0.517  -0.646
          40725  -0.79    0.942
          40726   2.821   2.234
          40727   2.297   2.071
          40728   1.455   2.241
          40729   1.979   1.785
          40730   0.273   1.72
          40731   1.919   2.336
          40732   0.913   1.001
          40733   2.593   1.083
          40734   0.196   1.059
              ⋮
          88710   5.251   4.863
          88711   2.497   4.299
          88712   4.479   5.188
          88713   3.821   4.436
          88714   3.2     3.628
          88715   4.071   3.695
          88716   5.211   4.313
          88717   4.588   3.443
          88718   5.5     5.09
          88719   4.354   4.566
          88720   4.743   5.057
          88721   2.763   3.263
```

```
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}:
            4
            8
           15
           16
           17
           24
           28
           35
           36
           37
           44
           48
           55
            ⋮
          156
          157
          164
          168
          175
          176
          177
          184
          188
          195
          196
          197
```

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

```
Out[83]: 150-element Array{Int64,1}:
            1
            2
            3
            5
            6
            7
            9
           10
           11
           12
           13
           14
           18
            ⋮
          185
          186
          187
          189
          190
          191
          192
          193
          194
          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  1  2  1  0  0  0  1  1  1  1  1  …  1  0  1  2  1  1  1  0  0  2  1  2
 0  2  0  2  2  0  0  0  2  0  0  2  2     2  1  1  1  1  2  0  1  1  0  2  1
 0  2  0  2  2  0  0  0  2  0  0  1  1     2  1  0  2  0  2  1  0  0  1  1  2
 0  2  0  2  2  0  0  0  2  0  0  0  0     0  0  2  2  0  0  2  0  0  2  2  2
 0  2  0  2  2  0  0  0  2  0  0  1  1     2  1  1  1  1  2  0  1  1  0  2  1
 0  2  2  2  0  0  0  0  2  0  0  2  0  …  0  0  2  2  0  0  2  0  0  2  2  2
 0  2  0  2  2  0  0  0  2  0  0  1  1     1  1  1  2  1  1  1  0  0  1  2  2
 0  2  0  2  2  1  1  1  1  1  1  1  2     1  1  1  2  1  1  1  1  1  1  2  1
 0  0  2  2  2  2  2  2  0  2  2  0  2     2  0  2  1  1  1  1  1  1  1  2  1
 1  2  0  2  2  2  2  2  0  2  2  0  2     2  0  1  2  1  2  0  0  0  0  2  2
 0  2  1  2  1  0  0  0  2  0  0  1  0  …  1  0  1  2  1  1  1  0  0  2  2  1
 0  2  0  2  2  0  0  0  2  0  0  1  1     2  2  2  0  2  2  0  2  2  0  2  0
 0  1  1  2  2  0  1  1  1  1  1  1  1     2  1  1  1  2  2  0  1  1  1  1  1
 ⋮              ⋮              ⋮           ⋱        ⋮              ⋮
 0  2  0  2  2  0  0  0  2  0  0  1  1     2  0  1  2  0  1  0  0  0  0  2  2
 0  1  1  2  2  1  1  1  1  1  1  0  1     2  0  0  2  0  2  0  0  0  0  2  2
 0  2  0  2  2  0  0  0  2  0  0  1  1  …  1  0  1  2  1  1  2  1  1  1  2  2
 1  2  0  2  2  0  1  1  1  1  1  1  1     2  2  2  0  2  2  0  2  2  0  2  0
 0  2  0  2  2  0  0  0  2  0  0  2  2     0  1  2  2  0  1  1  1  1  1  2  2
 1  2  0  2  2  0  0  0  2  0  0  2  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  2     1  0  1  2  0  1  1  0  0  1  2  2
 1  2  0  2  2  0  0  0  2  0  0  1  0  …  2  1  1  1  2  2  0  1  1  1  2  0
 0  2  0  2  1  0  0  0  2  0  0  2  1     0  0  2  2  0  0  2  0  0  2  2  2
 0  2  0  2  2  0  0  0  2  0  0  0  0     1  0  1  2  0  1  1  0  0  2  0  2
 0  2  0  2  2  0  0  0  2  0  0  2  2     1  0  2  2  0  0  1  0  0  1  2  2
 0  2  0  2  2  0  0  0  2  0  0  1  1     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  1  1  0  2  1  2  1  1  0  0  0  …  0  2  2  1  1  2  2  1  0  1  0  0
 2  0  0  2  1  1  0  1  1  0  0  0  2     2  2  2  1  0  0  0  2  1  0  1  1
 2  0  1  1  1  1  1  0  2  1  1  0  1     0  2  2  1  0  0  1  2  1  1  0  0
 2  0  2  0  0  0  1  1  2  0  0  0  2     1  2  2  0  2  1  2  2  0  2  0  0
 2  0  1  1  1  1  1  2  1  0  2  2  0     0  2  2  2  1  0  1  2  1  0  1  1
 2  0  2  0  0  0  0  1  2  0  0  0  0  …  0  2  2  0  2  1  2  2  0  2  0  0
 2  0  1  1  2  1  1  0  2  1  0  1  1     1  2  2  1  1  2  2  2  0  1  0  0
 2  1  0  2  0  0  0  2  1  0  1  2  0     0  2  2  2  1  1  1  2  0  1  1  1
 2  2  2  0  0  1  0  0  2  1  1  1  1     1  2  2  0  1  0  1  2  2  1  1  1
 2  2  0  2  0  1  0  1  1  0  0  0  1     0  2  2  1  2  1  2  2  0  0  0  0
 2  0  2  0  1  1  0  0  2  0  0  0  1  …  0  2  2  2  1  1  1  1  0  1  0  0
 2  0  1  1  1  1  0  1  1  0  0  0  2     0  1  1  1  2  1  2  2  2  0  2  2
 2  1  0  2  1  0  0  1  2  0  0  2  1     0  2  2  1  2  1  2  1  1  0  1  1
 ⋮              ⋮              ⋮        ⋱        ⋮              ⋮
 2  0  1  1  1  2  0  0  2  0  1  0  1     1  2  2  1  2  2  2  2  0  0  0  0
 2  1  2  0  0  2  0  1  1  1  1  0  1     0  2  2  0  1  2  2  2  0  0  0  0
 2  0  0  2  0  0  0  1  2  0  0  0  2  …  2  2  2  2  0  1  1  2  0  2  1  1
 2  1  1  2  1  1  0  1  1  1  0  0  2     0  2  2  1  0  0  0  2  2  0  2  2
 2  0  0  2  0  1  0  1  1  0  0  1  1     0  2  2  2  0  2  2  2  0  1  1  1
 2  0  0  2  0  1  0  1  1  0  0  0  1     0  2  2  1  2  1  2  2  0  1  0  0
 2  0  0  2  0  0  0  0  2  0  0  0  2     1  2  2  1  0  1  1  2  0  1  0  0
 2  0  1  1  0  1  0  0  2  1  1  1  1  …  1  2  2  1  0  0  0  1  1  0  1  1
 2  0  1  1  0  2  0  0  2  0  0  0  1     0  2  2  0  2  1  2  2  0  2  0  0
 2  0  2  0  2  1  0  2  2  0  0  0  2     0  2  2  0  2  2  2  2  0  1  0  0
 2  0  0  2  0  2  0  2  0  0  0  0  2     0  2  2  1  1  1  1  2  0  1  0  0
 2  0  1  1  1  1  0  0  2  0  0  0  1     2  2  1  0  1  1  1  2  2  0  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]:
```
0.6535270066329267
```

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

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

Out[100]:
```
0.809089070595065
```

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

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

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

```
Out[102]: 50-element Array{Float64,1}:
            0.311483
           -0.354575
            0.00793983
            0.271844
           -0.145448
            0.169581
           -0.215543
           -0.158637
           -0.180167
            0.0635007
            0.298307
            0.148968
            0.37014
            ⋮
           -0.151618
           -0.0308617
            0.146673
           -0.136285
            0.177039
           -0.0528884
           -0.251958
           -0.262687
            0.256572
            0.474199
           -0.00455869
           -0.538252
```

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

Out[103]: 48000-element Array{Float64,1}:
           2.28294 
           1.56428 
           1.01443 
           2.33966 
           1.96134 
           2.54011 
           3.0348  
           1.31721 
           2.57327 
           1.6535  
           1.65925 
           1.7691  
           0.741404
           ⋮       
           2.87665 
           0.692606
           2.2562  
           0.630318
           0.898278
           2.98366 
           3.0404  
           3.09579 
           2.58726 
           1.94531 
           2.45342 
           2.01786 

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

Out[104]: 1.3706723226645268

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

Out[105]: 1.4956977188052911

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

Out[106]: 1.6694782575516753
```

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

Out[107]: 1.831813708121996

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

Out[108]: 1.9166051854321942

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

Out[109]: 2.0079730151961575

In [110]: EAlphaG=onlyQTL*QTLo

Out[110]: 9000-element Array{Float64,1}:
           1.62084
           1.55041
           2.19367
           3.70218
           1.7372
           2.31991
           1.6369
           1.60152
           1.40875
           2.39581
           2.34007
           0.594775
           0.354449
           ⋮
           2.87665
           0.692606
           2.2562
           0.630318
           0.898278
           2.98366
           3.0404
           3.09579
           2.58726
           1.94531
           2.45342
           2.01786
```

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

Out[111]: 1.995600018340378

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

Out[112]: 0.6249276956758512

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

Out[113]: 1.6049949065763294

In [114]: meanBreedingValueSelectedG0=meanEAlphaS0-meanEAlphaG0

Out[114]: 0.2343225839118026

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

Out[115]: 1.8235170197292703

In [116]: meanBreedingValueSelectedG1=meanEAlphaS1-meanEAlphaG0

Out[116]: 0.4528446970647435

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

Out[117]: 1.9766145344210806

In [118]: meanBreedingValueSelectedG2=meanEAlphaS2-meanEAlphaG0

Out[118]: 0.6059422117565538

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

Out[119]: 1.9959686169094095

In [120]: meanBreedingValueSelectedG3=meanEAlphaS3-meanEAlphaG0

Out[120]: 0.6252962942448828
```

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

```
Out[121]: 2.081985139834612
```

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

```
Out[122]: 0.7113128171700851
```

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

```
Out[123]: 2.0079730151961575
```

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

```
Out[124]: 0.6373006925316307
```