Supporting Text

Selective Breeding Strategy

It was not immediately obvious what the best breeding strategy would be, particularly when we knew that multiple loci were contributing to the phenotype. Backcrossing iteratively against the low-responder parent would be the fastest way to "purge" the genome of unlinked elements of BALB/c origin but would risk the loss of susceptibility loci during the segregation, especially recessive loci or those with only minor phenotypic impact. Conversely, a succession of intercrosses would be safer but slower in enriching for susceptibility loci. To compare the various possibilities, we performed *in silico* simulations of such breedings. Computer code was written that simulated successive generations of mice, calculated virtual phenotypes based on different genetic models (modeling up to four loci with different relative weights), and selected the higher responder for breeding of the next generation. A number of breeding variables could be tested with this software (see below and Fig. 6). Considering the results from these simulations, we opted for a mixed backcross/intercross scheme, aiming for three productive breeding cages constituted from the strongest responders at each generation after the F_2, phenotyping by serum transfer five to seven offspring from each breeding pair and following this scheme for six generations.

The breedings were performed largely according to plan (Fig. 7*a*), and the evolution of the phenotypes at each generation matched expectations: high responses in the H_3 resulting from intercross of F_2 parents, with a drop in the H_4 due to the influx of SJL alleles. However, the reality schematized in Fig. 7*a* did not always live up to the idealized computer simulations (Fig. 6). Some breeders proved nonperformant, so that we had to rely more heavily than planned on a particular breeding cage at certain generations, significantly increasing the risk of allele loss. For the most part, animals in the pedigree did not display the fast onset of arthritis characteristic of the BALB/c parent, as shown on the biparametric plots of Fig. 2b. The scores of the association testing (see text) were not nearly as strong as could have been hoped for from the simulations. These observations suggested that the full phenotype requires the contribution of a larger number of genes than could be obtained in any one mouse, and that some loci contributing to high responsiveness were lost during the selection.

Statistical genetic analysis of the pedigree

The markers used were a combination of microsatellites and SNPs (105 markers altogether, with an average spacing of 18.3 Mb, listed in the Table 1). Because microsatellite length is dichotomous in any inbred pair, with negligible variation over the duration of the experiment (two instances of mutations resulting in a novel microsatellite length were observed in the pedigree), it was possible to combine the information from both types of markers. Two different analytical strategies were used:

The first test was designed to match the particular aspects of the pedigree, deriving combinatorial information from different generations in the cohort. Significance was estimated by comparison with a large number of randomly generated virtual pedigrees of identical structure, asking how frequent the phenotypic correlations and segregation patterns of markers in the true dataset would prove to be when a large number of neutral loci were simulated to segregate down an identical pedigree. This approach avoids any assumption on the distribution of the data or biases introduced by the particular pedigree structure. Several independent metrics of association were combined: (*i*) a measure of correlation between marker distribution and phenotype for the 32 mice of the founder F_2 cohort (the Pearson correlation coefficient between the Severity Index and the genotype at each marker; (*ii*) a measure of the persistence of BALB/c alleles at the H_6 generation, reflecting the persistence of high-responsiveness alleles that were positively selected during the crosses and the loss of BALB/c alleles at loci with no phenotypic impact; and (*iii*) a measure of phenotype/genotype correlation (Spearman) in the selected animals of the H_5 cohort. For the simulated pedigree, the exact family structure of the true pedigree was simulated (code in *Supporting Text*), with 10,000 unselected markers segregating neutrally with simple Mendelian transmission (all virtual markers were unlinked). These virtual animals were attributed the same arthritis phenotypes as the real counterparts, the same association measures were calculated, and the probability was estimated as a combination of the same three measures. The results are illustrated in Fig. 3*a*; full numeric data are presented in Table 2. Although no metric conferred significance by itself, significant association ($P < 0.001$) could be discerned from the combined information, on chromosomes 2 (50-160 Mb) and 6 (50-90 Mb), with lower signals on chromosomes 1, 7, 9, 10, 11, 12, 13, and 19. Interestingly, the analysis also highlighted loci where the BALB/c allele was negatively correlated with arthritis severity (not shown). This was the case on *chr5* and on *chr1*, quite close to a region that is positively associated with severity.

The second analysis used a more conventional family-based tests, treating the entire pedigree as a group of families, and searching for linkage using QTDT software {4846}, which allows association mapping for quantitative traits using nuclear families of any size with or without parental information, and is particularly robust when each sib set is large. Several authors have suggested that, for collagen-induced arthritis (CIA) and proteoglycan-induced arthritis (PGIA), different loci may affect the speed of onset and the maximum severity {4860, 4862}. Thus, onset and ankle thickening metrics were used in this analysis, as well as the combined Severity Index. The entire pedigree was divided into 16 nuclear families, and simple linear association was tested with Abecasis' orthogonal model {4846}. Fig. 3*b* presents the regions with logarithm of odds scores >2 found on *Chrs 1*, *-2*, *-6*, and *-7*, with a particularly robust association on *Chr2*. Association signals were observed for both time of onset and degree of inflammation, with no clear distinction between the two metrics.

Computer code for breeding simulations and pedigree analysis

In all instances, the S-Plus code runs on S-PLUS Ver. 2000 or 6.1 for Windows, following the embedded documentation ("commented" following # symbols).

Breeding simulations

These scripts simulate a selective breeding experiment from two inbred lines with high and low phenotypes, respectively. A variable number of breeding cages are set up at each generation, and a chosen number of offspring are obtained, their phenotypes computed on the basis of the genetic model and relative gene weights, and the next breeders chosen on the basis of the deduced phenotypes.

In all cases, the low-responder parents are given allele 0, the high-responder parent allele 1 (genotypes coded as 0,1,2).

A. Data matrices specifying the GeneModels and Breeding Strategies

The Breed.genmodel matrix specifies the number and relative weights of individual loci that determine the virtual phenotype. Each column represents a different genetic model (up to five causal genes); the relative phenotypic weight is given as triplets of values (rows 1 to 3, 4 to 6, etc.), where the first is the phenotypic value associated with the 0 genotype, the second the phenotypic value of the 1 genotype, and the third the phenotypic value of the 2 genotype). Low-responder genotypes should be given a token phenotypic weight >0, because zero values are used to indicate that the locus has no impact and will be discounted from the calculation of “Controlling Loci.”

Values can be edited in the matrix before running the script. In the example here, model #1 (column 1) is a two equal-gene model, both additive; model #5 has four additive gees,one of which is   , model #5 has four equal dominant genes, etc

3	2	1	2	1	1	2	2	2	2
17	11	8	6	5	25	2	2	20	20
32	22	16	12	10	25	25	10	25	20
3	2	1	2	1	1	2	2	2	2
17	11	8	6	5	25	2	2	20	20
32	22	16	12	10	25	25	30	25	20
0	2	1	2	1	1	2	2	2	2

0	11	8	6	5	25	2	10	20	20
0	22	16	12	10	25	25	10	25	20
0	0	1	2	1	1	2	2	0	2
0	0	8	6	15	25	2	30	0	20
0	0	16	12	30	25	25	30	0	20
0	0	0	2	0	0	0	0	0	0
0	0	0	6	0	0	0	0	0	0
0	0	0	12	0	0	0	0	0	0
2	3	4	5	4	4	4	4	3	4

The Breed.strategies character matrix specifies the breeding strategies in successive generations (assumes breeding will originate from F_2 to generate H_3, etc.). Each row corresponds to a strategy, with the format for each generation given in the successive column (starting from $F_2>H_3$). The admissible values are "inter" (intercross), "bcklo" (backcross to the low responder parent), or "bckhi" (backcross to the high-responder parent).

Values can be edited in the matrix before running the script. In the example here and which is used for the paper, strategy #1 (row 1) is a succession of backcross and intercross steps, strategy #2 has only backcrosses to the low-responder (allele 0) parent, strategy #3 is a succession of intercrosses, etc.).

bcklo	inter	bcklo	inter	bcklo	inter
bcklo	bcklo	bcklo	bcklo	bcklo	bcklo
inter	inter	inter	inter	inter	inter
inter	bcklo	inter	inter	inter	inter

inter	inter	bckl o	inter	inter	inter
inter	inter	bckl o	bckl o	inter	inter
inter	inter	bckl o	bckl o	bckl o	inter
inter	inter	bckl o	bckl o	bckl o	bckl o

B. Simulation and plotting scripts

The key setting are found on line 4 of the actual code, where the user sets the variables of the “contmat” numeric matrix that guides the running of each successive generation. Several strategies, genetic models, or breeding numbers can be specified; the program will iterate through all permutations.

“dep” is the depth of the pedigree (the number of generations through which the breeding runs).

“st” is the number of animals in the starting F_2 generation.

“br” is the number of breeding cages set up at each generation.

“li” is the number of littermates obtained and tested at each generation.

“gmd” is the genetic model, indicates which column of the Breed.Genemodel matrix are used, and thus how the phenotype will be calculated.

“sgy” indicates the breeding strategy which of the rows of the Breed.Strategy matrix are used.

“iter”is the number of iterated repeats through which the code should run.

```
#For K.O.'s paper, adapted from the DUAL script for high
   responder selection only,
#Designed to test best strategies and mouse numbers, as a
   function of gene model,
#and compare to a similar number of straight F2 mice.

#For selections of high responders only.
#Uses the same basic control matrix, which sets parameters.
#But takes the Breed.strategies matrix for breeding schemes
   (strategies in rows).
#Gene models in Breed.genemodels (models in columns, where
   consecutive triplets
#give the weight of the 0,1,2 genotypes for each gene).
#Performs successive breedings according to contmat
   settings
#(can go through different models, multiple iterations).
#Tracks five putatively controlling loci (as a function of
   the GeneModels asked for)
#as well as 100 dummy loci that segregate randomly.


AllHiMice<-matrix(,0,109)
AllFitH<-matrix(,0,111)
LocCount<-matrix(,0,9)

### USER: SET MAIN VARIABLES HERE ###
contmat<-
   d.contmat(dep=8,st=32,br=3,li=6,gmd=1:4,sgy=1:3,iter=1)

#Set the way the genotype/phenotype linear regression
   analysis will be performed.
#Can be cumulative (add the mice from each generation and
   perform on all pooled data)
# or stepwise (for each generation, the linear regression
   analysis is done of the mice of the F2 + that generation)
analysis.mode_"cumulative."
```

```
#Set whether you should calculate a control where the same
  incremental number of F2 mice are analyzed for
  comparison.
#Syntax is "mockbreed_T" (yes) or "mockbreed_F" (no)
mockbreed_T

#and choose also whether you want the expanded F2 analysis
  to be cumulative (all mice) or stepwise (starting F2 +
  current generation)
mockfit.mode_"cumulative."

#####################################
if  (analysis.mode!="stepwise"&analysis.mode!="cumulative")
  stop("Need to specify a valid analysis mode")
for (k in 1:(nrow(contmat)))      #Main loop that iterates
  through each row of the control matrix
{ param<-c(contmat[k,1:6])
  gen<-as.numeric(contmat[k,1])
  stn<-as.numeric(contmat[k,2])
  brd<-as.numeric(contmat[k,3])
  lit<-as.numeric(contmat[k,4])
  g<-Breed.genemodel[,contmat[k,6]]
  g1<-g[1:3]; g2<-g[4:6]; g3<-g[7:9]; g4<-g[10:12]; g5<-
  g[13:15]
  cntloci<-
  (1:5)[c(sum(g1)>0,sum(g2)>0,sum(g3)>0,sum(g4)>0,sum(g5)>0
  )]  #Control Loci (CL) which determine phenotype
  TotCL<-length(cntloci)
if (gen==2) {
        F2<-d.gencoh(st=stn)
        F2<-d.calcphen(mat= F2,av=g)
        HN<- F2 [order(-F2 [,"p1"]),]
        #AllHiMice<-rbind(AllHiMice, HN[,1:109])
    #Unblock if you want to save all mice (can get heavy
  with large iterations)
        AllCurrent<-HN                  #Resets      the
  current cohort when going back to starting population
        HF<-d.lm.lod(mat= F2)
        AllFitH_rbind(AllFitH,c(param,HF))
        LostCL<-
  sum(colSums(HN[,(cntloci+4)]==0)==nrow(HN))
        CleanNonCL<-
  sum(colSums(HN[,10:109]==0)==nrow(HN))

    LocCount_rbind(LocCount,c(param,TotCL,LostCL,CleanNonC
  L))
        }
```

```
if (gen>2) { e<-contmat[k,5]
          f<-(contmat[k,1]-2)
          strat<-Breed.strategies[e,f]
          H<-
  d.breed(inpop=HN,st=strat,br=brd,li=lit,genevals=g)
          HN<-H[[2]]
          HN<-HN[order(-HN[,"p1"]),]
          #AllHiMice<-rbind(AllHiMice, HN[,1:109])
     #Unblock if want to save all mice (can get heavy with
  large iterations)
          AllCurrent<-rbind(AllCurrent, HN)
          if        (analysis.mode=="stepwise")        HF<-
  d.lm.lod(mat=rbind(HN, F2))
          if       (analysis.mode=="cumulative")       HF<-
  d.lm.lod(AllCurrent)
          AllFitH_rbind(AllFitH,c(param,HF))
          LostCL<-
  sum(colSums(HN[,(cntloci+4)]==0)==nrow(HN))
          CleanNonCL<-
  sum(colSums(HN[,10:109]==0)==nrow(HN))

     LocCount_rbind(LocCount,c(param,TotCL,LostCL,CleanNonC
  L))
       }

}

  dimnames(LocCount)_list(NULL,c("Generation","stratn","brd
  n","litn","strategy","genemodel","TotCL","LostCL","CleanN
  onCL"))

#Set names for Gene models and strategies

  ModNms_cbind(1:7,c("2Add","3Add","4Add","5Add","4Add1Maj"
  ,"4Rec","4Dom"))
StratNms_cbind(1:3,c("Altern","Bcklo","Inter"))
strats_unique(contmat[,5])
gmods_unique(contmat[,6])
iters_unique(contmat[,7])

###To simulate a mock breeding, only if gene models are
  iterated (other parameters fixed)
#Calculate the values for a "strategy 0,"
#where the same number of F2 mice would have been analyzed
  (for each gene model)
if (mockbreed==T){
x_unique(contmat[,1]) #generations
```

```
mockfit<-matrix(,0,108)  #Gen,  Strat=0,  genemodel,  g1:g5,
   1:100)
if (mockfit.mode=="cumulative"){
CumNb_contmat[1:length(x),1:2]  #cumulative  nb  of  mice  at
   each generation
for     (k     in     2:nrow(CumNb))     CumNb[k,2]_CumNb[k-
   1,2]+(contmat[k-1,3]*contmat[k-1,4])
for (i in gmods){g<-Breed.genemodel[,gmods[i]]
     for                        (j                        in
   1:nrow(CumNb)){mockparam_c(CumNb[j,1],0,gmods[i])
                       x_d.gencoh(CumNb[j,2])
                       y_ F2<-d.calcphen(mat=x,av=g)
                       z<-d.lm.lod(mat=y)

     mockfit_rbind(mockfit,c(mockparam,z))}}}
if (mockfit.mode=="stepwise"){
CumNb_contmat[1:length(x),1:2]  #cumulative  nb  of  mice  at
   each generation
for             (k             in             2:nrow(CumNb))
   CumNb[k,2]_CumNb[1,2]+(contmat[1,3]*contmat[1,4]) #all
   equal
for (i in gmods){g<-Breed.genemodel[,gmods[i]]
     for                        (j                        in
   1:nrow(CumNb)){mockparam_c(CumNb[j,1],0,gmods[i])
                       x_d.gencoh(CumNb[j,2])
                       y_F2<-d.calcphen(mat=x,av=g)
                       z<-d.lm.lod(mat=y)

     mockfit_rbind(mockfit,c(mockparam,z))}}}
          } #end of mockbreed conditional
```

C. Functions used in the computation

Simple functions required for the main script to run; they need to be entered into the S-Plus database by running the code once.

```
#D.CONTMAT    Set   up   the   overall   matrix   that   allows
   progression through different values of arguments
#Modified for the Mega set
d.contmat<-function(iter = 1,
dep = 6,
st = 32,
br = 3,
li = 6,
sgy=1,
gmd=3)
{gen<-2:dep
it<-1:iter
contmat <- matrix(,,7)
x<- list(gen,st, br, li, sgy, gmd,it)
for (k in 1:7) {y<-x[[k]]
            contmat[,k]<-y[1]
            if (length(y)>1) {cm<-contmat
                        for(i  in  2:length(y))  {cm[,k]<-
  y[i]; contmat<-rbind(contmat,cm)}}}
hd                                                    <-
  c("Generation","startn","brdn","litn","strategy","genemod
  el","iteration")
dimnames(contmat)<-list(NULL,hd)
return(contmat)}


#D.CALCPHEN  Can be used to calculate the phenotypes in any
  matrix of the kilobreed format
#where "p1" is the phenotypic index
#g1 to g5 are the five gene control loci, whose
d.calcphen<-function                                    (
  mat=famille,noisevar=10,av=c(2,10,20,2,10,20,2,10,20,2,10
  ,20,0,0,0))
{g1<-av[1:3]; g2<-av[4:6]; g3<-av[7:9]; g4<-av[10:12]; g5<-
  av[13:15]
for  (k  in  1:nrow(mat))  {mat[k,"p1"]<-  g1[(mat[k,"g1"]+1)]
  +g2[(mat[k,"g2"]+1)]                 +g3[(mat[k,"g3"]+1)]
  +g4[(mat[k,"g4"]+1)]+g5[(mat[k,"g5"]+1)]+noisevar*rnorm(1
  )}
mat[,"p1"]_ceiling(mat[,"p1"])
#mat<-mat[order(mat[,"p1"]),]  ####  output  not  ordered  in
  this version
return(mat)}



#D.GENCOH Generates input matrix with F2 distribution, up
  to 5 contloci, 100 blanks
d.gencoh<-function (st=32)
{cohin_matrix(,st,109)
hds<-
  c("p1","Gen","pID","Sex","g1","g2","g3","g4","g5",1:100)
dimnames(cohin)_list(NULL, hds)
cohin[,"Gen"]_2
cohin[,"pID"]_1
```

```
cohin[,"Sex"]_round(runif(st,min=0,max=1),digits=0)
a_st*105
cohin[,5:109]_round(runif(a,min=0,max=2),digits=0)
return(cohin)}


#D.KW.FIT  Calculates the fit of each locus (contloci as
   well as blanks) in the LN cohort, +/- other mice
#Uses the Kruskal test, outputs -log(10) of P value
   (straight P values in B summary)
#Input is a genotype matrix, same columns as fastbreed
   matrix, phenotype in 1, genotypes in 5:109.
d.kw<-function(mat=LN)
{  MM<-mat
   t<-matrix(,0,105) ; AA<-c()
        for (j in 5:109) {  M<-MM[is.number(MM[,j]),]
                           fit<-kruskal.test(M[,1],M[,j])
                           AA<-c(AA,fit$p.value)}
A<-replace(AA,is.na(AA),1)
L<- -(log(A,10))
L<-round(L,digits=2)
     return(L)
   }


#D.LM.LOD  Another calculation of correlation between
   individual genotypes and the phenotype
#Calculates the linear regression fit of each locus
   (contloci as well as blanks) in the input cohort
#Additive model, heteros and homos are split
#Input is a matrix of the breed format, phenotype in 1,
   genotypes in 5:109
#Returns the vector of -log10 of the pvalues for the linear
   regression
d.lm.lod<-function(mat=LN)
{x_nrow(mat)
   L_c()
   mat1_mat2_mat
   for (j in 1:x)
   {for(k in 5:109) { if(mat[j,k]>0) mat1[j,k]<-1 else
   mat1[j,k]<-0
                    if (mat[j,k]==2) mat2[j,k]<-1 else
   mat2[j,k]<-0 }}
   for            (i            in            5:109)
   {fit_lm(mat1[,1]~mat1[,i]+mat2[,i],singular.ok=T)
     t_summary(fit)$fstatistic
     t_1-pf(t[1],t[2],t[3])
     t_round(-log10(t),digits=2)
            L_c(L,t)}
     names(L)<-c("g1","g2","g3","g4","g5",1:100)
     return(L)}

#D.SUMMARY: Returns the fit values for all markers, after
  elimination of those whose
# Dif value is below a given quantile, combining both high
  and low selection
#RES is the listing for each generation, giving fit values
  for the control loci (g1 to g5)
#the number of contloci, how many of those lost, how many
  dummy loci retained/lost and min/max
d.summary_function(Thrhld=0.5)
{
N_nrow(AllFitH)
atn_matrix(,0,105)
RES_matrix(,0,17)
L_list()
for (i in 1:N) {g<-Breed.genemodel[,AllFit[i,6]]
  g1<-g[1:3];  g2<-g[4:6];  g3<-g[7:9];  g4<-g[10:12];  g5<-
  g[13:15]
  cntloci<-
  sum(sum(g1)>0,sum(g2)>0,sum(g3)>0,sum(g4)>0,sum(g5)>0)
  if (AllFit[i,1]==2) at_AllFit[i,7:111]
  if (AllFit[i,1]>2) {
  u_AllDifH[i,7:111]
  v_AllDifL[i,7:111]
  w_AllFit[i,7:111]
  u_u>quantile(u,Thrhld)
  v_v>quantile(v,Thrhld)
  at_u*v*w}
  L[[length(L)+1]]_at
  lcl_cntloci-sum(at[1:cntloci]>0)
  nrlv_at[(cntloci+1):105]
  outDL_sum(nrlv==0)
  nrlvl_nrlv[nrlv>0]
  inDL_length(nrlvl)
  minDL_min(nrlvl)
  maxDL_max(nrlvl)

     res_c(AllFit[i,1:6],at[1:5],cntloci,lcl,inDL,outDL,min
  DL,maxDL)
  RES_rbind(RES,res)}
  hd_c("Generation","startn","brdn","litn","strategy","gene
   model","g1","g2","g3","g4","g5","contloci","lostCL","inDL
   ","outDL","minDL","maxDL")
   dimnames(RES)_list(NULL,hd)
   RL_list(RES,L)
   return(RL)
}



#D.BREED Breed routine  for D.breed, converts an ordered
   inpop to cohout (not ordered)
#returns list of cohout and delta.p matrices
#genotypes in inpop must be in cols 5:9
d.breed<-function (inpop = F2, st = "bckhi" , br = 3 , li =
   6, genevals=g)
{inmat<-inpop
VH<-
   sum(genevals[3],genevals[6],genevals[9],genevals[12],gene
   vals[15])
VL<-
   sum(genevals[1],genevals[4],genevals[7],genevals[10],gene
   vals[13])
famille<-matrix(,li+1,109)
delta<-matrix(,li,109)
t<-matrix(,br+1,109)
hds<-
   c("p1","Gen","pID","Sex","g1","g2","g3","g4","g5",1:100)
dimnames(famille)<-list(NULL, hds)
dimnames(delta)<-list(NULL, hds)
cohout<-matrix(,0,109)
dimnames(cohout)<-list(NULL, hds)
ldp<-list()
length(ldp)<-105
breeders_cohout
cohlo<-matrix(,br,109)
cohhi<-matrix(,br,109)
cohlo[,5:109]<-0   #All alleles from strain a are "0"
cohhi[,5:109]<-2   #All alleles from strain b are "2"
#select breeders for intercross
if   (st=="inter")   {coh0<-inmat[inmat[,4]==0,];   coh1<-
   inmat[inmat[,4]==1,]}
if (st=="bcklo") {coh0<-inmat[1:br,1:109]; coh1<-cohlo}
if (st=="bckhi") {coh0<-inmat[1:br,1:109]; coh1<-cohhi}
j<-li+3
#start breeding
for (h in 1:br){famille<-rbind(coh0[h,],coh1[h,],famille)
```

```
            for     (i     in     5:109)     {famille[3,i]<-
   famille[1,i]*famille[2,i]+famille[1,i]+famille[2,i]
                        if                 (famille[3,i]==0)
   famille[4:j,i]<-0
                        if                 (famille[3,i]==1)
   famille[4:j,i]<-round(runif(li,min=0,max=1),digits=0)
                        if                 (famille[3,i]==2)
   famille[4:j,i]<-1
                        if                 (famille[3,i]==3)
   famille[4:j,i]<-round(runif(li,min=0,max=2),digits=0)
                        if                 (famille[3,i]==5)
   famille[4:j,i]<-round(runif(li,min=1,max=2),digits=0)
                        if                 (famille[3,i]==8)
   famille[4:j,i]<-2
                        }
a_famille[1:3,]
b_famille[4:j,]
b_d.calcphen(mat=b,av=genevals)
b[,2]<- (a[1,2]+1)
b[,3]<-h
b[,4]<-round(runif(li,min=0,max=1),digits=0)
cohout<-rbind(cohout,b)
breeders<-rbind(breeders,a)
if (st=="bckhi") {v<-matrix(rep(a[1,],li),li,byrow=T)
     #diff from parent (bck)
   delta<- b-v
   for (i in 5:109) {
   d1_cbind(delta[,1],delta[,2],delta[,i])
   x_matrix(,,3)
   if (a[1,i]==a[2,i]) d1[,3]<-NA
   x_rbind(d1)
   x_x[is.number(x[,3])==T,]
   if (is.matrix(x)==T) {x[,2]<- (famille[1,2]+1)
   ldp[[(i-4)]]<-rbind(ldp[[(i-4)]],x)}}}
if (st=="bcklo") {v<-matrix(rep(a[1,],li),li,byrow=T)
     #diff from parent (bck)
   delta<- b-v
   for (i in 5:109) {
   d1_cbind(delta[,1],delta[,2],delta[,i])
   x_matrix(,,3)
     if (a[1,i]==a[2,i]) d1[,3]<-NA
   x_rbind(d1)
   x_x[is.number(x[,3])==T,]
   if (is.matrix(x)==T) {x[,2]<- (famille[1,2]+1)
   ldp[[(i-4)]]<-rbind(ldp[[(i-4)]],x)}}}
if (st=="inter") {v<-matrix(rep(a[1,],li),li,byrow=T)
     #diff from parent (inter)
   w<-matrix(rep(a[2,],li),li,byrow=T)
   delta1<- b-v
   delta2<- b-w
   for (i in 5:109) {
   if (a[1,i]==0 | a[1,i]==2) {delta1[,i]<-NA}
   if (a[2,i]==0 | a[2,i]==2) {delta2[,i]<-NA}
   d1_cbind(delta1[,1],delta1[,2],delta1[,i])
   d2_cbind(delta2[,1],delta2[,2],delta2[,i])
   x_matrix(,,3)
   x_rbind(d1,d2)
   x_x[is.number(x[,3])==T,]
   if (is.matrix(x)==T) {x[,2]<- (famille[1,2]+1)
   ldp[[(i-4)]]<-rbind(ldp[[(i-4)]],x)}}}
famille<-matrix(NA,li+1,109)
}
   #delta.p<-delta.p[order(delta.p[,"p1"]),]
   L<-list(breeders,cohout,ldp)
   return(L)}
```

Data Analysis

A. Generation of a simulated dataset reproducing the actual pedigree

The simulated pedigree reproduces the experimental pedigree structure (the number of breeding cages set up at each generation, the breeding strategy, and the number of offspring obtained are specified). The number of independent genetic markers segregating in the crosses is also specified. The virtual animals generated are given the same individual numbers as the actual ones (taken from the BxS.gen matrix). Outputs the "BxS.SimGen" numeric matrix used in the significance computation below.

```
#Simulates the BalbxSJL selected breeding (F2 founder
   generation and subsequent Hx generations).
#Generates a matrix with N genes segregating independently
   and randomly
#The breeding numbers are entered in the first ###USER###
   segment of the code,
#the breeding strategy in the second #### USER### part
#Output is the BxS.SimGen numeric matrix

##### USER #####
#Set N, the number of genes tracked
```

```
N_10000

#Set the number of mice in the starting F2 cohort
f2.sz_32

#Now set the descriptors of the cross. For each generation,
  starting with F2,
#specify the number of cages bred from a generation
  (hx.brd) and the number of offspring from each cage
  (hx.product)
#Additional pairs of hx.brd and hx.product could be entered
  if needed
f2.brd_3
f2.product_c(2,4,8)
h3.brd_3
h3.product_c(1,2,5)
h4.brd_4
h4.product_c(3,7,8,10)
h5.brd_2
h5.product_c(1,7)
h6.brd_3
h6.product_c(0,4,5)

#### Functions ####
#sim.offspring generates a set of offspring from two
  parents
#Only genotypes are tracked, offspring have the same number
  of genes as parent
#and are a random combination of genotypes.
#Returns a off.mat matrix of N rows, as many columns as
  there are genes
sim.offspring<- function(p1,p2,n){
m_length(p1)
off1_matrix(rep(p1,n),n,m,byrow=T)
off2_matrix(rep(p2,n),n,m,byrow=T)
x_off1==1
off1[x]_round(runif(sum(x)),digits=0)
x_off1==2
off1[x]_1
x_off2==1
off2[x]_round(runif(sum(x)),digits=0)
x_off2==2
off2[x]_1
off.tot_off1+off2
return(off.tot)}

#make.gen functions
```

```
#Function that makes a generation from the previous one.
   Arguments are the parent generation (matrix)
#, the number of breedings, and the number per family
   (typically the nn.brd and nn.product parameters)
make.gen.inter<- function(par.mat, nb.brd, nb.product){
   x_sample(1:nrow(par.mat),size=(2*nb.brd))
   for (i in 1:nb.brd){k_x[(2*(i-1))+1]
        l_x[(2*(i-1))+2]
        k_par.mat[k,]
        l_par.mat[l,]
        m_nb.product[i]
        off.mat_sim.offspring(k,l,m)
        if (i==1) next.gen_off.mat
        if (i>1) next.gen_rbind(next.gen,off.mat)}
          return(next.gen)}
#       L_list(next.gen,par.mat[x,])
#       return(L)}
make.gen.bck<- function(par.mat,nb.brd,nb.product){
   x_sample(1:nrow(par.mat),size=(nb.brd))
   for (i in 1:nb.brd){
        p1_par.mat[x[i],]
        p2_sjl
        off.mat_sim.offspring(p1,p2,n=nb.product[i])
        if (i==1) next.gen_off.mat
        if (i>1) next.gen_rbind(next.gen,off.mat)}
          return(next.gen)}

##
runif(5)       #reset random seed
sjl_rep(0,N)
#Set the F2 generation. Size of a is F2 size x N

   a_sample(c(0,1,2),size=(f2.sz*N),prob=c(0.25,0.5,0.25),re
   place=T)
F2_matrix(a,f2.sz,N)
#Then run the subsequent generations

#### USER ####
#Specify the actual function used to generate each
   generation from the previous one
#(as make.gen.inter for an intercross, or "make.gen.bck"
   for a backcross)
#If a backcross, the parent will be assumed to be SJL,
   genotype 0 at all loci

H3_make.gen.inter(F2,f2.brd,f2.product)
H4_make.gen.bck(H3,h3.brd,h3.product)
```

```
H5_make.gen.inter(H4,h4.brd,h4.product)
H6_make.gen.inter(H5,h5.brd,h5.product)
H7_make.gen.bck(H6,h6.brd,h6.product)
#########################

F2_cbind(rep(2,f2.sz), F2)
x_rep(3+((1:f2.brd)*0.1), f2.product)
H3_cbind(x,H3)
x_rep(4+((1:h3.brd)*0.1), h3.product)
H4_cbind(x,H4)
x_rep(5+((1:h4.brd)*0.1), h4.product)
H5_cbind(x,H5)
x_rep(6+((1:h5.brd)*0.1), h5.product)
H6_cbind(x,H6)
BxS.SimGen_rbind(F2,H3,H4,H5,H6)
#Add true individual nb
BxS.SimGen_cbind(BxS.gen[-1,1],BxS.SimGen)
#Add bogus marker Ids
BxS.SimGen_rbind(c(99,99,1:(ncol(BxS.SimGen)-
   2)),BxS.SimGen)
```

B. Significance analysis of locus-wise phenotype correlation and selection throughout the pedigree

Used to calculate the impact of individual markers on the phenotype during the pedigree. The input data are in the formar of a BxS.gen, BxS.phen, and BxS.mrks matrices, as well as the BxS.SimGen simulated data generated in section A.

The key output is the AllFitH vector, which gives the
-log10(pvalue) for the distribution (computed from a combination of genotype/phenotype correlation in the F2 and H5 generations and the remaining count of Balb alleles in the H6).

```
#BxS calculation
#Designed 11-03 for linkage analysis up to the H6
   generation, rev 0204, 0604
#Input (either true or simulated)
# BxS.gen matrix: Indivs in rows. First column has IndivNb,
   second has generation ID
#           (if from the simulation, family indicator also)
#           then one column for each marker. First row has
  marker Nb
# BxS.phen matrix: Indivs in rows. First column has
  IndivNb, second has phenotype. First row 0,0 to match
  with gen
# BxS.mrks matrix: First row has marker Nb (as per gen),
  second chromosome, third Mb position
#Calculates:   - The linear regression P value in all F2s,
  phen vs each marker
#         - The Pearson correlation coefficient in all F2s,
  phen vs each marker
#         - The Spearman correlation coefficient in H5
  mice, phen vs each marker (OK for non-normal
  distributions)
#         - The genotype delta between hi and low
  responders at H5, phen vs each marker


#### USER ####
# Choose the matrices to use
#mat.g is the genotype matrix. Choose the real data
  (BxS.gen) or the simulated one (BxS.SimGen)
mat.sim_BxS.SimGen
mat.real_BxS.gen

#mat.p is the phenotype matrix. By default, should use
  BxS.phen
mat.p_BxS.phen


#######################
if (nrow(mat.g)!=nrow(mat.p)) stop("Data doesn't
  fit,different row numbers in genotype and phenotype
  matrices")

for (k in 1:2){
  if (k==1) mat.g_mat.real
  if (k==2) mat.g_mat.sim
#Calculate Pearson cor coeff on F2
f2.pears_rep(0,ncol(mat.g)-2)
f2_mat.g[,2]>=2&mat.g[,2]<3 #To use the decimal generation
  identifiers from BxS.SimGen
y_mat.p[f2,2]
for (i in 3:ncol(mat.g)){
x_mat.g[f2,i]
fit_cor.test(x,y)
f2.pears[i-2]_round(fit$estimate,digits=3)}
#Calculate Spearman correlation coeff on all
all.spear_rep(0,ncol(mat.g)-2)
y_mat.p[-1,2]
for (i in 3:ncol(mat.g)){
x_mat.g[-1,i]
fit_cor.test(x,y,method="spearman")
all.spear[i-2]_round(fit$estimate,digits=3)}

#Calculate Spearman cor coeff on H5
h5.spear_rep(0,ncol(mat.g)-2)
h5_mat.g[,2]>=5&mat.g[,2]<6
y_mat.p[h5,2]
for (i in 3:ncol(mat.g)){
x_mat.g[h5,i]
if (var(x,na.method="omit")==0) {h5.spear[i-2]_0
            next}
fit_cor.test(x,y,method="spearman")
h5.spear[i-2]_round(fit$estimate,digits=3)}

#Calculate H5 skew: the allele imbalance in the H5
h5hi_mat.g[,2]>=5&mat.g[,2]<6&mat.p[,2]>30
h5lo_mat.g[,2]>=5&mat.g[,2]<6&mat.p[,2]<30
h5.skew_colSums(mat.g[h5hi,],na.rm=T)-
  colSums(mat.g[h5lo,],na.rm=T) #To show skewing of the
  distribution, simply subtract the sum of genotypes in hi-
  lo responder
h5.skew_h5.skew[-(1:2)]

#Calculate H6 count: The frequency of "1" alleles remaining
  at H6
h6_mat.g[,2]>=6&mat.g[,2]<7
h6.cnt_(colSums(mat.g[h6,-
  (1:2)]==1,na.rm=T)+2*colSums(mat.g[h6,-
  (1:2)]==2,na.rm=T))/(sum(h6)*2)

if (k==1){f2.pearsT_f2.pears
       h5.spearT_h5.spear
       h5.skewT_h5.skew
       h6.cntT_h6.cnt
       all.spearT_all.spear}
if (k==2){f2.pearsS_f2.pears
       h5.spearS_h5.spear
       h5.skewS_h5.skew
       h6.cntS_h6.cnt
       all.spearS_all.spear}
}
```

```
#Calculate probabilities:
#Calculate, in the simulated dataset, the proportion of
  markers that show the combination
#of F2 Pearson correlation and H5 skew of the true results
pvals.2and5_rep(1,length(f2.pearsT))
N_length(f2.pearsS)
for (i in 1:length(f2.pearsT)){
  x_f2.pearsS>=f2.pearsT[i]
  y_h5.skewS>=h5.skewT[i]
  z_x&y
  pvals.2and5[i]_(-(log10((sum(z)+1)/N)))   #add 1 to
  avoid log0 and to give the limit, given the number of
  iterations
  pvals.2and5_round(pvals.2and5,digits=2)}

#Calculate the proportion of markers in the simulated
  dataset that show the same Spearman correlation
  coefficient over the whole mouse set
N_length(all.spearS)
n_length(all.spearT)
pvals.all_rep(1,n)
for (i in 1:n){
  x_all.spearS>=all.spearT[i]
  pvals.all[i]_(-(log10((sum(x)+1)/N)))   #add 1 to avoid
  log0 and to give the limit, given the number of
  iterations
  pvals.all_round(pvals.all,digits=2)}

#Calculate, in the simulated dataset, the proportion of
  markers that show the combination
#of F2 Pearson correlation and H5 Spearman, and H6 allele
  count of the true results
pos.pv.256_rep(1,length(f2.pearsT))
N_length(f2.pearsS)
for (i in 1:length(f2.pearsT)){
  x_f2.pearsS>=f2.pearsT[i]
  y_h5.skewS>=h5.skewT[i]
  w_h6.cntS>=h6.cntT[i]
  z_x&y&w
  pos.pv.256[i]_(-(log10((sum(z)+1)/N)))    #add 1 to
  avoid log0 and to give the limit, given the number of
  iterations
  pos.pv.256_round(pos.pv.256,digits=2)}
```

```
#Calculate, in the simulated dataset, the proportion of
   markers that show the combination
#of negative F2 Pearson correlation and H5 Spearman, and H6
   allele count of the true results
neg.pv.256_rep(1,length(f2.pearsT))
N_length(f2.pearsS)
for (i in 1:length(f2.pearsT)){
   x_f2.pearsS<=f2.pearsT[i]
   y_h5.skewS<=h5.skewT[i]
   w_h6.cntS<=h6.cntT[i]
   z_x&y&w
   neg.pv.256[i]_(-(log10((sum(z)+1)/N)))}    #add 1 to
   avoid log0 and to give the limit, given the number of
   iterations


par(mfrow=c(2,2))
hist(f2.pearsS,nclass=15,xlim=c(-0.6,0.6))
hist(h5.spearS,nclass=15,xlim=c(-0.6,0.6))
hist(f2.pearsT,nclass=15,xlim=c(-0.6,0.6))
hist(h5.spearT,nclass=25,xlim=c(-0.6,0.6))
hist(all.spearT,nclass=15,xlim=c(-0.8,0.8))
hist(all.spearS,nclass=15,xlim=c(-0.8,0.8))
plot(f2.pearsT,h5.spearT,xlim=c(-0.8,0.8),ylim=c(-0.8,0.8))
plot(f2.pearsS,h5.spearS,xlim=c(-0.8,0.8),ylim=c(-0.8,0.8))
plot(f2.pearsT,h5.skewT,xlim=c(-0.6,0.6),ylim=c(-12,12))
plot(f2.pearsS,h5.skewS,xlim=c(-0.6,0.6),ylim=c(-12,12))
plot(f2.pearsT,h6.cntT,xlim=c(-0.6,0.6),ylim=c(-0.1,1))
plot(f2.pearsS,h6.cntS,xlim=c(-0.6,0.6),ylim=c(-0.1,1))
plot(1:length(f2.pearsT),pvals.2and5,xlab=paste("Marker
   Nb"),ylab=paste("LOD score, F2 and H5"))
plot(1:length(f2.pearsT),pvals.all,xlab=paste("Marker
   Nb"),ylab=paste("LOD score, all generations"))
plot(1:length(f2.pearsT),pos.pv.256,xlab=paste("Marker
   Nb"),ylab=paste("Combined LOD, F2,H5,H6"))
plot(1:length(f2.pearsT),neg.pv.256,xlab=paste("Marker
   Nb"),ylab=paste("Combined Low resp LOD, combined
   F2,H5,H6"))
```