Core Sims

Katie Lotterhos

December 13, 2015

In this document, I will make the dataframe for the core Nemo Simulations.

Fitness function

We are using a Gaussian (quadratic) function to describe stabilizing selection on the phenotype (z) of individual (i) in population (k) with phenotypic optimum Θ, and selection variance ω_k^2:

$$W_{z_{i,k}} = 1 - \frac{(z_{i,k} - \Theta_k)^2}{\omega_k^2}$$

As an R function:

```
setwd("/Users/katie/Desktop/CurrResearch/G1_TestTheTests/2015_12coresims")

w.zik <- function(zik, theta, omega.sq){
  1 - (zik - theta)^2/omega.sq
}
```

For the 2-patch model, we assume that each patch has an optimum of +1 and -1. We assume that for a given number of loci that affect the trait(`ntot`) at the lowest level of redundancy, their effect sizes (`alpha`) are $1/(2 * ntot)$. For this, we can use the supplemental equation from Yeaman 2015 Am Nat to calculate the critical migration rate m_c. At $m > m_c$, alleles are prone to swamping by migration.

```
theta.1 <- 1
theta.2 <- -1
omega.sq <- 25
ntot <- c(10,20,60,100,400)
alpha <- 1/(2*ntot)


N=1000
m_crit <- function(alpha, theta.1, theta.2, omega.sq, N){
  w1bb <- 1#w.zik(4*alpha,theta.1, omega.sq)/
    #w.zik(4*alpha, theta.1, omega.sq)
  w1Bb <- w.zik(2*alpha,theta.1, omega.sq)/
    w.zik(4*alpha, theta.1, omega.sq)
  w1BB <- w.zik(0, theta.1,omega.sq)/
    w.zik(4*alpha, theta.1, omega.sq)

  w2bb <- w.zik(4*alpha,theta.2, omega.sq)/
    w.zik(0, theta.2, omega.sq)
  w2Bb <- w.zik(2*alpha,theta.2, omega.sq)/
    w.zik(0, theta.2, omega.sq)
  w2BB <- 1#w.zik(0, theta.2,omega.sq)/
    #w.zik(0, theta.2, omega.sq)
```

```
  w1bb;w1Bb; w1BB
  w2bb;w2Bb; w2BB
  return(1/(w1Bb/(w1Bb-w1BB*(1+(1/(4*N))))-w2Bb/(w2BB*(1+(1/(4*N)))-w2Bb))
  )
  }
mc <- m_crit(alpha, theta.1, theta.2, omega.sq, N)
mc
```

```
## [1]  4.688562e-02  2.319798e-02  3.530533e-03  1.235666e-03 -3.840833e-05
```

Note that for the smallest effect size, alleles will always be prone to swamping by migration.

Levels

Now, we want to extend this base set to cover redundancy (1.5 ntot, 2 ntot), mutation rate, Ne (1000, 5000), environmental variance (0,2,4), and 9 migration rates (0.01mc, 0.1mc, 0.5mc, 0.9mc, 0.99mc, 1.01mc, 1.1mc, 1.5mc, 2mc) spanning the critical threshold, standing variation vs. new mutation, clustered vs. unclustered. This gives a maximum of :

```
length(ntot)* #levels polygenicity
  3* #levels redundancy
  2* #levels Ne
  3* #levels envi var
  9* #levels migration
  2*  #levels mutation
  2 # standing variation vs. new mutation
```

```
## [1] 3240
```

levels, although note that some levels will not be possible (e.g. $m_c = 0$ for ntot>500). We will do 3 replicates of each level.

Eventually we will layer demography, clustered vs. unclustered QTNs, and deleterious mutations onto this framework.

Genetic map

I've found that run time in Nemo increases with chromosome number (because crossing overs need to be computed for each one) and memory depends on chromosome length and resolution (because length determines the number of elements in the recombination lookup table (From `void GeneticMap::setLookupTable`: "The lookup table for a 1M map at the 0.01 cM scale will have 10,000 elements")). In humans (N_e=10000), 1Mb (1,000,000 bases) correponds to 1 cM and a recombination rate $r = 0.01$. In humans, the per-nucleotide mutation rate and recombination rate is assumed to be around 10^{-8} or $N_e\mu = N_e r = 10^{-4}$.

We will simulate on 21 linkage groups, each 10cM long. The 21st linkage groups will always have neutral loci.

Let's set the resolution of the map such that we capture recombination rates as low as 1e-06 (0.0001 cM). `prefix_genetic_map_resolution 0.0001` (i.e., a distance of 1 then corresponds to a recombination rate of 1e-06 between two loci).

At this resolution, 100,000 indexes in Nemo corresponds to a map length of 10 cM (in humans, this corresponds to an index as 100 bases apart or total map length of 10 Mb). This means that a the recombination among loci located at each end of the segment will be about $r = 0.1$.

With 8,400 total neutral loci, we have 400 neutral loci per chromosome. We COULD model a neutral locus about every 250 indexes, corresponding to an average recombination rate of 0.00025. Instead, I propose to track groups of loci spread out along a chromosome, mimicking the data that we might acquire after sequence capture or RNAseq. I propose to place SNPs over 40 "locations" on each chromosome (each location approximately 100 indexes long and ~2500 indexes apart (r=0.0025 among locations)), getting 10 neutral SNPs per location.

Each location will have 1 QTN in the center (relative index = 50) and 5 neutral SNPs evenly spaced on either side (indexes 1, X, X, 40, 49,QTN, 51, 60, X, X, 100), correpsonding to recombination among neighboring SNP-QTL of r=1e-06 and among QTL-SNP pairs at the end of the segment of r=5e-05

In humans, 100 indexes corresponds to 10,000 bases or 10Kb or the average length of a gene.

```
  numseg <- 40#number of segments per chromosome
  seglength <- 100
  startseq <- 500 # starting location of first locus
  locs_start <- sort(round(seq(startseq, (100000-startseq-seglength), length.out=numseg)))
  locs_start
```

```
##  [1]   500  3036  5572  8108 10644 13179 15715 18251 20787 23323 25859
## [12] 28395 30931 33467 36003 38538 41074 43610 46146 48682 51218 53754
## [23] 56290 58826 61362 63897 66433 68969 71505 74041 76577 79113 81649
## [34] 84185 86721 89256 91792 94328 96864 99400
```

```
   all_locs <- lapply(locs_start, function(a) a:(a+seglength-1))
  locs_dist_between <- all_locs[[2]][1]-all_locs[[1]][seglength] # approx distance among locs
  locs_dist_between
```

```
## [1] 2437
```

```
  length(unlist(all_locs)) #possible sites on a chrom
```

```
## [1] 4000
```

```
  #head(all_locs)
  #tail(all_locs, 1)
  bp <- unlist(all_locs)
  #head(all_locs)
  #tail(all_locs)

  ### Set up the genetic map for the first chromosome
  locs.df <- data.frame(linkage.group=1, bp=bp, contig=rep(1:numseg, each=seglength), type="no")
  locs.df$type<-as.character(locs.df$type)

  ### QTN equally spaced in center of each segment
  QTN_loc <- locs_start + seglength/2
  QTN_loc
```

```
##  [1]   550  3086  5622  8158 10694 13229 15765 18301 20837 23373 25909
## [12] 28445 30981 33517 36053 38588 41124 43660 46196 48732 51268 53804
## [23] 56340 58876 61412 63947 66483 69019 71555 74091 76627 79163 81699
## [34] 84235 86771 89306 91842 94378 96914 99450
```

```
  locs.df[which(bp %in% QTN_loc),]
```

```
##      linkage.group    bp contig type
## 51               1   550      1   no
## 151              1  3086      2   no
## 251              1  5622      3   no
## 351              1  8158      4   no
## 451              1 10694      5   no
## 551              1 13229      6   no
## 651              1 15765      7   no
## 751              1 18301      8   no
## 851              1 20837      9   no
## 951              1 23373     10   no
## 1051             1 25909     11   no
## 1151             1 28445     12   no
## 1251             1 30981     13   no
## 1351             1 33517     14   no
## 1451             1 36053     15   no
## 1551             1 38588     16   no
## 1651             1 41124     17   no
## 1751             1 43660     18   no
## 1851             1 46196     19   no
## 1951             1 48732     20   no
## 2051             1 51268     21   no
## 2151             1 53804     22   no
## 2251             1 56340     23   no
## 2351             1 58876     24   no
## 2451             1 61412     25   no
## 2551             1 63947     26   no
## 2651             1 66483     27   no
## 2751             1 69019     28   no
## 2851             1 71555     29   no
## 2951             1 74091     30   no
## 3051             1 76627     31   no
## 3151             1 79163     32   no
## 3251             1 81699     33   no
## 3351             1 84235     34   no
## 3451             1 86771     35   no
## 3551             1 89306     36   no
## 3651             1 91842     37   no
## 3751             1 94378     38   no
## 3851             1 96914     39   no
## 3951             1 99450     40   no
```

```
  locs.df$type[bp %in% QTN_loc] <- "qtn"
  #locs.df[1:51,]

  ### neutral loci at increasing distance from QTN
  neut_dist_from_qtn <- c(1, 5, 10, 25, 49)
  neut1 <- sapply(QTN_loc, function(x)(x-neut_dist_from_qtn))
  neut2 <- sapply(QTN_loc, function(x)(x+neut_dist_from_qtn))
  locs.df$type[bp %in% c(neut1, neut2)] <- "neut"
  locs.df[1:101,]
```

```
##     linkage.group   bp contig type
## 1               1  500      1   no
## 2               1  501      1 neut
## 3               1  502      1   no
## 4               1  503      1   no
## 5               1  504      1   no
## 6               1  505      1   no
## 7               1  506      1   no
## 8               1  507      1   no
## 9               1  508      1   no
## 10              1  509      1   no
## 11              1  510      1   no
## 12              1  511      1   no
## 13              1  512      1   no
## 14              1  513      1   no
## 15              1  514      1   no
## 16              1  515      1   no
## 17              1  516      1   no
## 18              1  517      1   no
## 19              1  518      1   no
## 20              1  519      1   no
## 21              1  520      1   no
## 22              1  521      1   no
## 23              1  522      1   no
## 24              1  523      1   no
## 25              1  524      1   no
## 26              1  525      1 neut
## 27              1  526      1   no
## 28              1  527      1   no
## 29              1  528      1   no
## 30              1  529      1   no
## 31              1  530      1   no
## 32              1  531      1   no
## 33              1  532      1   no
## 34              1  533      1   no
## 35              1  534      1   no
## 36              1  535      1   no
## 37              1  536      1   no
## 38              1  537      1   no
## 39              1  538      1   no
## 40              1  539      1   no
## 41              1  540      1 neut
## 42              1  541      1   no
## 43              1  542      1   no
## 44              1  543      1   no
## 45              1  544      1   no
## 46              1  545      1 neut
## 47              1  546      1   no
## 48              1  547      1   no
## 49              1  548      1   no
## 50              1  549      1 neut
## 51              1  550      1  qtn
## 52              1  551      1 neut
## 53              1  552      1   no
```

```
## 54              1  553      1   no
## 55              1  554      1   no
## 56              1  555      1 neut
## 57              1  556      1   no
## 58              1  557      1   no
## 59              1  558      1   no
## 60              1  559      1   no
## 61              1  560      1 neut
## 62              1  561      1   no
## 63              1  562      1   no
## 64              1  563      1   no
## 65              1  564      1   no
## 66              1  565      1   no
## 67              1  566      1   no
## 68              1  567      1   no
## 69              1  568      1   no
## 70              1  569      1   no
## 71              1  570      1   no
## 72              1  571      1   no
## 73              1  572      1   no
## 74              1  573      1   no
## 75              1  574      1   no
## 76              1  575      1 neut
## 77              1  576      1   no
## 78              1  577      1   no
## 79              1  578      1   no
## 80              1  579      1   no
## 81              1  580      1   no
## 82              1  581      1   no
## 83              1  582      1   no
## 84              1  583      1   no
## 85              1  584      1   no
## 86              1  585      1   no
## 87              1  586      1   no
## 88              1  587      1   no
## 89              1  588      1   no
## 90              1  589      1   no
## 91              1  590      1   no
## 92              1  591      1   no
## 93              1  592      1   no
## 94              1  593      1   no
## 95              1  594      1   no
## 96              1  595      1   no
## 97              1  596      1   no
## 98              1  597      1   no
## 99              1  598      1   no
## 100             1  599      1 neut
## 101             1 3036      2   no
```

```
  locs.df.chr1 <- locs.df
  for (i in 2:21){ #21 linkage groups
    l2 <- locs.df.chr1
    l2$linkage.group[1:nrow(l2)] <- i
    locs.df <- rbind(locs.df,l2)
```

```
    }

head(locs.df, 101)
```

```
##     linkage.group   bp contig type
## 1               1  500      1   no
## 2               1  501      1 neut
## 3               1  502      1   no
## 4               1  503      1   no
## 5               1  504      1   no
## 6               1  505      1   no
## 7               1  506      1   no
## 8               1  507      1   no
## 9               1  508      1   no
## 10              1  509      1   no
## 11              1  510      1   no
## 12              1  511      1   no
## 13              1  512      1   no
## 14              1  513      1   no
## 15              1  514      1   no
## 16              1  515      1   no
## 17              1  516      1   no
## 18              1  517      1   no
## 19              1  518      1   no
## 20              1  519      1   no
## 21              1  520      1   no
## 22              1  521      1   no
## 23              1  522      1   no
## 24              1  523      1   no
## 25              1  524      1   no
## 26              1  525      1 neut
## 27              1  526      1   no
## 28              1  527      1   no
## 29              1  528      1   no
## 30              1  529      1   no
## 31              1  530      1   no
## 32              1  531      1   no
## 33              1  532      1   no
## 34              1  533      1   no
## 35              1  534      1   no
## 36              1  535      1   no
## 37              1  536      1   no
## 38              1  537      1   no
## 39              1  538      1   no
## 40              1  539      1   no
## 41              1  540      1 neut
## 42              1  541      1   no
## 43              1  542      1   no
## 44              1  543      1   no
## 45              1  544      1   no
## 46              1  545      1 neut
## 47              1  546      1   no
## 48              1  547      1   no
```

```
## 49              1  548      1   no
## 50              1  549      1 neut
## 51              1  550      1  qtn
## 52              1  551      1 neut
## 53              1  552      1   no
## 54              1  553      1   no
## 55              1  554      1   no
## 56              1  555      1 neut
## 57              1  556      1   no
## 58              1  557      1   no
## 59              1  558      1   no
## 60              1  559      1   no
## 61              1  560      1 neut
## 62              1  561      1   no
## 63              1  562      1   no
## 64              1  563      1   no
## 65              1  564      1   no
## 66              1  565      1   no
## 67              1  566      1   no
## 68              1  567      1   no
## 69              1  568      1   no
## 70              1  569      1   no
## 71              1  570      1   no
## 72              1  571      1   no
## 73              1  572      1   no
## 74              1  573      1   no
## 75              1  574      1   no
## 76              1  575      1 neut
## 77              1  576      1   no
## 78              1  577      1   no
## 79              1  578      1   no
## 80              1  579      1   no
## 81              1  580      1   no
## 82              1  581      1   no
## 83              1  582      1   no
## 84              1  583      1   no
## 85              1  584      1   no
## 86              1  585      1   no
## 87              1  586      1   no
## 88              1  587      1   no
## 89              1  588      1   no
## 90              1  589      1   no
## 91              1  590      1   no
## 92              1  591      1   no
## 93              1  592      1   no
## 94              1  593      1   no
## 95              1  594      1   no
## 96              1  595      1   no
## 97              1  596      1   no
## 98              1  597      1   no
## 99              1  598      1   no
## 100             1  599      1 neut
## 101             1 3036      2   no
```

```
  tail(locs.df, 101)
```

```
##       linkage.group    bp contig type
## 83900            21 96963     39 neut
## 83901            21 99400     40   no
## 83902            21 99401     40 neut
## 83903            21 99402     40   no
## 83904            21 99403     40   no
## 83905            21 99404     40   no
## 83906            21 99405     40   no
## 83907            21 99406     40   no
## 83908            21 99407     40   no
## 83909            21 99408     40   no
## 83910            21 99409     40   no
## 83911            21 99410     40   no
## 83912            21 99411     40   no
## 83913            21 99412     40   no
## 83914            21 99413     40   no
## 83915            21 99414     40   no
## 83916            21 99415     40   no
## 83917            21 99416     40   no
## 83918            21 99417     40   no
## 83919            21 99418     40   no
## 83920            21 99419     40   no
## 83921            21 99420     40   no
## 83922            21 99421     40   no
## 83923            21 99422     40   no
## 83924            21 99423     40   no
## 83925            21 99424     40   no
## 83926            21 99425     40 neut
## 83927            21 99426     40   no
## 83928            21 99427     40   no
## 83929            21 99428     40   no
## 83930            21 99429     40   no
## 83931            21 99430     40   no
## 83932            21 99431     40   no
## 83933            21 99432     40   no
## 83934            21 99433     40   no
## 83935            21 99434     40   no
## 83936            21 99435     40   no
## 83937            21 99436     40   no
## 83938            21 99437     40   no
## 83939            21 99438     40   no
## 83940            21 99439     40   no
## 83941            21 99440     40 neut
## 83942            21 99441     40   no
## 83943            21 99442     40   no
## 83944            21 99443     40   no
## 83945            21 99444     40   no
## 83946            21 99445     40 neut
## 83947            21 99446     40   no
## 83948            21 99447     40   no
## 83949            21 99448     40   no
```

```
## 83950           21 99449     40 neut
## 83951           21 99450     40  qtn
## 83952           21 99451     40 neut
## 83953           21 99452     40   no
## 83954           21 99453     40   no
## 83955           21 99454     40   no
## 83956           21 99455     40 neut
## 83957           21 99456     40   no
## 83958           21 99457     40   no
## 83959           21 99458     40   no
## 83960           21 99459     40   no
## 83961           21 99460     40 neut
## 83962           21 99461     40   no
## 83963           21 99462     40   no
## 83964           21 99463     40   no
## 83965           21 99464     40   no
## 83966           21 99465     40   no
## 83967           21 99466     40   no
## 83968           21 99467     40   no
## 83969           21 99468     40   no
## 83970           21 99469     40   no
## 83971           21 99470     40   no
## 83972           21 99471     40   no
## 83973           21 99472     40   no
## 83974           21 99473     40   no
## 83975           21 99474     40   no
## 83976           21 99475     40 neut
## 83977           21 99476     40   no
## 83978           21 99477     40   no
## 83979           21 99478     40   no
## 83980           21 99479     40   no
## 83981           21 99480     40   no
## 83982           21 99481     40   no
## 83983           21 99482     40   no
## 83984           21 99483     40   no
## 83985           21 99484     40   no
## 83986           21 99485     40   no
## 83987           21 99486     40   no
## 83988           21 99487     40   no
## 83989           21 99488     40   no
## 83990           21 99489     40   no
## 83991           21 99490     40   no
## 83992           21 99491     40   no
## 83993           21 99492     40   no
## 83994           21 99493     40   no
## 83995           21 99494     40   no
## 83996           21 99495     40   no
## 83997           21 99496     40   no
## 83998           21 99497     40   no
## 83999           21 99498     40   no
## 84000           21 99499     40 neut
```

```
#   set.seed(4210)
#   newseed <- runif(21*20,min = 1000, max=99999)
#   k <- 0
#   locs.df$type <- "no"
#   for (chrom in 1:21){
#     for (seg in 1:20){
#       k <- k + 1
#       set.seed(newseed[k])
#       rows <- as.numeric(sample(rownames(locs.df)[locs.df$chr==chrom & locs.df$seqID==seg], size = 20
#       locs.df$type[rows] = "neut"
#     }
#   }
#
#   head(locs.df, 10)
#   tail(locs.df, 10)
#   head(locs.df[locs.df$type=="neut",], 50)
```

```
  num.neut <- sum(locs.df$type=="neut") ## total number neutral loci
  num.neut
```

The following code randomly assigns neutral loci and is not evaluated

```
## [1] 8400
```

```
  num.qtn <- sum(locs.df$type=="qtn")
  num.qtn
```

```
## [1] 840
```

```
  #### Write neutral map to Nemo format ###
  coreMapNeutralFile <- "coreMapNeutral.txt"
  write("ntrl_genetic_map \\ ", coreMapNeutralFile)
  for (i in 1:21){
    if(i==1){towrite <- paste("{{",paste(locs.df$bp[locs.df$linkage.group==i & locs.df$type=="neut"], co
    if(i==21){towrite <- paste("{",paste(locs.df$bp[locs.df$linkage.group==i & locs.df$type=="neut"], co
    if(i>1 & i<21){
    towrite <- paste("{",paste(locs.df$bp[locs.df$linkage.group==i & locs.df$type=="neut"], collapse=",
    }
    write(towrite, coreMapNeutralFile, append=TRUE)
  }#end loop


  #### Write qtl map to Nemo format ###
  coreMapQuantiFile <- "coreMapQuanti.txt"
  write("quanti_genetic_map \\\ ", coreMapQuantiFile)
  for (i in 1:21){
    if(i==1){towrite <- paste("{{",paste(locs.df$bp[locs.df$linkage.group==i & locs.df$type=="qtn"], col
    if(i==21){towrite <- paste("{",paste(locs.df$bp[locs.df$linkage.group==i & locs.df$type=="qtn"], col
    if(i>1 & i<21){
```

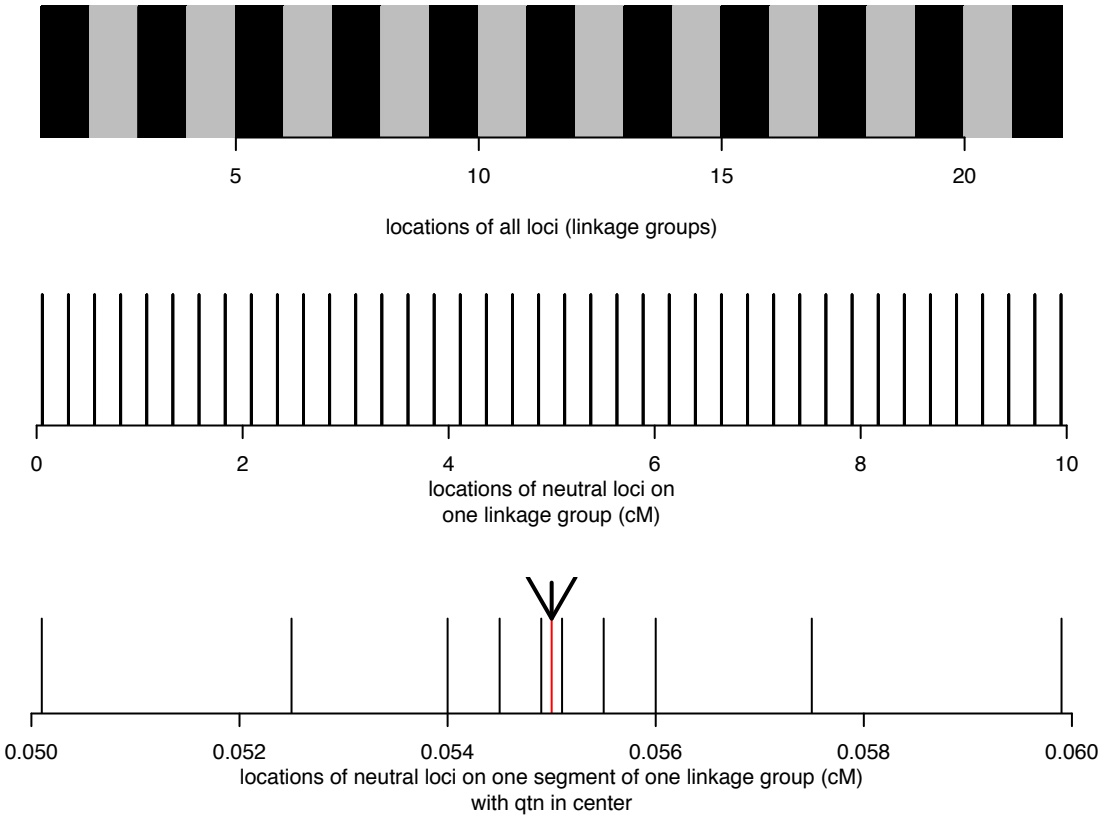
```
    towrite <- paste("{",paste(locs.df$bp[locs.df$linkage.group==i & locs.df$type=="qtn"], collapse=",")
    }
    write(towrite, coreMapQuantiFile, append=TRUE)
  }#end loop

  ### Write the locus dataframe to R format ###
  locs.df$pos <- (locs.df$linkage.group + (locs.df$bp)/100000)
  locs.df$col <- "black"
  locs.df$col[locs.df$linkage.group%%2==0]<-"grey"
  write.table(locs.df, "CoreSetGeneticMapFULL.txt", row.names=FALSE)
  write.table(locs.df[locs.df$type!="no",], "CoreSetGeneticMapREDUCED.txt", row.names=FALSE)
```

```
par(mfrow=c(3,1), mar=c(5,1,1,1))
    plot(locs.df$pos, rep(1, length(locs.df$pos)), type="h", xlab= "locations of all loci (linkage group

  cond <- which(locs.df$type=="neut" & locs.df$pos<2)
    plot(locs.df$bp[cond]/10000, rep(1, length(cond)), type="h", xlab= "locations of neutral loci on\nor

  cond <- which(locs.df$type!="no" & locs.df$pos<1.03)
    plot(locs.df$bp[cond]/10000, rep(1, length(cond)), type="h", xlab= "locations of neutral loci on one
  arrows(0.055,1.2, 0.055,1, lwd=2)
```



Initialization

Burnin for 4Ne generations with no selection until mutation-migration-drift balance.

After equil. is reached, we may then re-initialize qtl. In one case, we can re-initialize them to have 0 frequency to mimic selection from new mutation. In another case, we can keep the frequencies of qtl at the end of burnin to study selection from standing genetic variation. NEED TO COMARE THESE TWO CASES.

Then simulate again until 4Ne mutation-migration-drift-selection balance.

An alternative appraoch: From Vatsiou et al. 2015 MEC: For all scenarios, we used an initialization procedure that samples allele frequencies from an island model at migration–mutation–drift equilibrium. More precisely, all loci were initialized at the beginning of the simula- tions, t0 = 0, by sampling the allele frequencies of each locus from a beta distribution with parameters $a = 4Nem * p$ and $b = 4Nem * (1 - p)$, where p is the fre- quency in a migrant pool, which was derived from real human SNP data from noncoding regions, m is the migration rate, and Ne is the effective population size (Wright)

Mutation and Ne levels

For mutation, let's assume we want to at least capture $N_e\mu$=1e-04, which would approximate humans. This gives μ=1e-07 when $N_e = 1000$ or μ=1e-08 when $N_e = 10000$. Assume our artifical increase in mutation is related to the map resolution (100 bp/ index).

- Ne= 1000; μ=1e-06
- Ne= 1000; μ=1e-05
- Ne= 10000; μ=1e-06
- Ne= 10000; μ=1e-05

From Thornton et al. 2013 (Using a simulator similar to SLiM for a 100kB region) We simulated a population of N=20,000 diploids with a neutral mutation rate of u=0.00125 per gamete per generation, and a recombination rate of r=0.00125 per diploid per generation. These values correspond to the scaled parameters $\theta = 4N\mu = 100$ and $\rho = 4Nr = 100$, and thus correspond to a "typical" 100 kilobase region of the human genome. The mutation rate to causative (deleterious) mutations was $\mu_d = 0.1\mu$ per gamete per generation.

(Here, the logic is 0.00125/100,000 bases ~ 10^{-8})

From Caballero et al. 2013 The per-nucleotide mutation rate u and recombination rate r were assumed to be equal to 10^{-7}, implying values of $N_e u = N_e r = 10^{-4}$, which are appropriate for human populations (Li and Sadler 1991; Kong et al. 2002). Thus, because we used Ne = 1000 in the simulations and effective sizes for human populations are an order ofmagnitude larger (see, e.g., Charlesworth 2009), we increased the mutation and recombi- nation rates by an order ofmagnitude to simulate the genetic variation corresponding to a population that is 10 times larger. The scaled recombination rate is consistent with an average value of 1 cM/Mb in the genome.

A constant unstructured population of size N = Ne = 1000 individuals was run for 10,000 generations. This burn-in period ensured that allele frequencies were close to mutation-selection equilibrium. In the final burn-in generation, the population was expanded to 10,000 individuals to simulate a frequency distribution of genetic variants corresponding to an unscaled population size that was 10 times larger

Clustered vs. unclustered QTNs.

To address reviewer comments about clustering, propose to compare 2 simulations:

- 1 QTN in each QTL with mu = 1e-06
- 100 QTNs in each QTL with mu = 1e-08

Should give similar phenotypic evolution

```
Ne <- c(1000, 10000)
mu <- c(1e-07, 1e-06, 1e-05) # levels??
Ne[1]*mu; Ne[2]*mu
```

```
## [1] 1e-04 1e-03 1e-02
```

```
## [1] 0.001 0.010 0.100
```

```
resolution <- 0.0001
envi_var <- c(0,3)
redundancy <- c(1, 1.5, 2)
#set.seed(198)
#seed <- sample(9999:99999, 3)
seed <- c(1,2,3)
expand.grid(Ne=Ne, envi_var=envi_var,
                   mu=mu,
                   resolution=resolution,
            redundancy=redundancy)
```

```
##       Ne envi_var    mu resolution redundancy
## 1   1000        0 1e-07      1e-04        1.0
## 2  10000        0 1e-07      1e-04        1.0
## 3   1000        3 1e-07      1e-04        1.0
## 4  10000        3 1e-07      1e-04        1.0
## 5   1000        0 1e-06      1e-04        1.0
## 6  10000        0 1e-06      1e-04        1.0
## 7   1000        3 1e-06      1e-04        1.0
## 8  10000        3 1e-06      1e-04        1.0
## 9   1000        0 1e-05      1e-04        1.0
## 10 10000        0 1e-05      1e-04        1.0
## 11  1000        3 1e-05      1e-04        1.0
## 12 10000        3 1e-05      1e-04        1.0
## 13  1000        0 1e-07      1e-04        1.5
## 14 10000        0 1e-07      1e-04        1.5
## 15  1000        3 1e-07      1e-04        1.5
## 16 10000        3 1e-07      1e-04        1.5
## 17  1000        0 1e-06      1e-04        1.5
## 18 10000        0 1e-06      1e-04        1.5
## 19  1000        3 1e-06      1e-04        1.5
## 20 10000        3 1e-06      1e-04        1.5
## 21  1000        0 1e-05      1e-04        1.5
## 22 10000        0 1e-05      1e-04        1.5
## 23  1000        3 1e-05      1e-04        1.5
## 24 10000        3 1e-05      1e-04        1.5
## 25  1000        0 1e-07      1e-04        2.0
## 26 10000        0 1e-07      1e-04        2.0
## 27  1000        3 1e-07      1e-04        2.0
## 28 10000        3 1e-07      1e-04        2.0
## 29  1000        0 1e-06      1e-04        2.0
## 30 10000        0 1e-06      1e-04        2.0
## 31  1000        3 1e-06      1e-04        2.0
## 32 10000        3 1e-06      1e-04        2.0
```

```
## 33  1000        0 1e-05      1e-04        2.0
## 34 10000        0 1e-05      1e-04        2.0
## 35  1000        3 1e-05      1e-04        2.0
## 36 10000        3 1e-05      1e-04        2.0
```

For each of the above levels, we have multiple levels of allele effect sizes and multiple levels of migration rates.

The following code is not evaluated.

```
mig_levels <- c(0.01, 0.1, 0.5, 0.9, 0.99, 1.01, 1.1, 2, 10, 100)

i=0;params <- c()
for (a in seq_along(ntot)){
  for (b in seq_along(Ne)){
    mc<- m_crit(alpha[a], theta.1, theta.2, omega.sq, Ne[b])
    if (mc<=0){
      mc <- c(0.0001)
    }
    for (c in seq_along(mig_levels)){
      i <- i+1; #print(i)
      m=mc*mig_levels[c]
      if (m>0 & m<0.5){
        out <- expand.grid(alpha=alpha[a], ntot=ntot[a]*redundancy,
                           Ne=Ne[b], mc=mc, m=mc*mig_levels[c],
                           envi_var=envi_var, mu=mu, rep=seed,
                            resolution=resolution, demog="IM2")
        out <- data.frame(out, redundancy);out
      if (i==1){
      params <- out
      }else{
        params <- rbind(params, out)
      }#end ifelse
      }#end if m
    }# end loop c
  }  # end loop b
}# end loop a

dim(params)
params$NeMu <- params$Ne*params$mu
tail(head(params,180), 20)
tail(params,220)
```

Standing genetic variation vs. new mutation

I want to make sure we are making good decisions about how to parameterize/initialize the traits. I also want to set this up so that in the future, when we want to run non-eq situations, we have a system in place for creating variability in the population.

Limitations to keep in mind:

- Certain arguments are temporal, but traits and the arguments we would want to manipulate are not. NOT POSSIBLE: selection_local_optima (@g0 {{0,0}} @g1000{{-1,1}}) Also we can't set the values of quantitative traits at different points in time (i.e. making them neutral for a while with quanti_allele_effect=0 and then having quanti_allele_effect=0.1).

- It does not seem to be possible to save a population with quanti and then reload the quanti trait. Nemo requires that quanti is initialized, and if I don't initialize it, then Nemo initializes it by default.

2) There are two ways to initialize quanti

- quanti_init_trait_values {{0}{0}} (same as quanti_init 0) If I initialize by quanti_init_trait_values, it appears Nemo randomly assigns alleles to individuals so that they have a 0 trait value (selection from standing genetic variation, but in this case the variation is assigned somewhat randomly among loci). As a result, all alleles at loci have about a 50% frequency.
- quanti_init_freq {{x,x,x,x,}} User can specify frequency for each allele. These can be made to be patch-specific (1 row for each patch). If the number of rows is lower than the number of patches, values will be recycled.

3) It seems to me, for a quantitative trait, evolution from new mutation vs. from standing genetic variation would be simulated this way:

new mutation: Simulate ntrl and quanti with 0 effect-size for 4Ne generations and then output datafiles. Load population file into Nemo, and start with:

- quanti_init_freq {{x,x,x,x,}} drawn from exactly 50% "0"s and 50% "1"s
- selection_local_optima {{-1,1}}

standing genetic variation: Simulate ntrl and quanti with REAL effect-size for 4Ne generations under stabilizing selection and selection_local_optima {{0,0}} and output datafiles, then Load population file into Nemo, but change: * selection_local_optima {{-1,1}} * quanti_init_freq {{x,x,x,x,}} based on observed quanti frequencies from 1st simulation Note that because Nemo doesn't appear to be able to load a quanti trait, we would have to calculate these ourselves after the simulation is done, write them to the new .ini file, and then output.

```
for (i in seq_along(nrow(params))){
  newfilename <- paste("Coreset2popIM_alpha=",sprintf("%f", params$alpha[i]),
                       "_ntot=", sprintf("%03d",params$ntot[i]),
                       "_Ne=",sprintf("%05d",params$Ne[i]),
                       "_mc=",round(params$mc[i],5),
                       "_m=",round(params$m[i],5),
                       "_envi_var=",params$envi_var[i],
                       "_mu=",params$mu[i],
                       "_redun=", params$redundancy[i],
                       "_rep=", params$rep[i],
                       ".ini",
                       sep="")
  newfnpath <- paste("ini_Core2patchIM/", newfilename,sep="")
  system(paste("cp IM_2patch_base.ini",newfnpath ))

 # write(paste("random_seed", paste(seed, collapse=" ")), newfnpath, append=TRUE)
  write(paste("filename", newfilename), newfnpath, append=TRUE)
  write(paste("generations", 4*params$Ne[i]+50000), newfnpath, append=TRUE)
  write(paste("patch_capacity", params$Ne[i]), newfnpath, append=TRUE)
  write(paste("breed_disperse_rate", sprintf("%f",params$m[i])), newfnpath, append=TRUE)
  write(paste("quanti_loci 840"), newfnpath, append=TRUE)
  write(paste("quanti_mutation rate", params$mu[i]), newfnpath, append=TRUE)
```

```
  write(paste("ntrl_mutation_rate", params$mu[i]), newfnpath, append=TRUE)
  logtime <- paste(4*params$Ne[i] + c(1, 5000, 10000, 50000), collapse=" ")
  write(paste("ntrl_output_logtime", logtime), newfnpath, append=TRUE)
  write(paste("quanti_output_logtime", logtime), newfnpath, append=TRUE)
  write(paste("quanti_environmental_variance", params$envi_var[i]), newfnpath, append=TRUE)

  ### Assign qtns
  ### Want to sample chromosomes equally
  locs.df.reduced <- locs.df[locs.df$type!="no",]
  head(locs.df[locs.df$type=="qtn",])

  nqtn <- params$ntot[i]
  chr.qtn <- sort(rep(1:20, length.out = nqtn)) #randomly assign qtns to chromosome
  locs.df.reduced$effect <- 0
  #loop through chromosomes and randomly assign qtn position
    for (j in seq_along(levels(factor(chr.qtn)))){
      num <- sum(chr.qtn==chr.qtn[j])
      possible.locs <- which(locs.df.reduced$type=="qtn" & locs.df.reduced$linkage.group==chr.qtn[j])
      locs <- sample(possible.locs,num)
      locs.df.reduced$effect[locs] <- params$alpha[i]
    } # end loop through j

   write(paste("quanti_allele_value {",
               paste(locs.df.reduced$effect[locs.df.reduced$type=="qtn"], collapse=","),
               "}"), newfnpath, append=TRUE)
    write.table(locs.df.reduced, file = paste(newfnpath,".GeneticMap", sep=""), row.names=FALSE)
}
```