Run dmc

Sarah P. Flanagan

03 April, 2018

First I'll set up the working environment.

and load the necessary files

```
vcf<-parse.vcf("p4.upd.vcf") #this is the smaller dataset
```

```
vcf$SNP<-paste(vcf$`#CHROM`,vcf$POS,sep=".")
scaffs<-levels(as.factor(vcf[,1]))
scaffs[1:22]<-lgs
scaff.starts<-tapply(vcf$POS,vcf$`#CHROM`,max)
scaff.starts<-data.frame(rbind(cbind(names(scaff.starts),scaff.starts)),stringsAsFactors = F)
locus.info<-c(colnames(vcf[1:9]),"SNP")
```

Now I'll go through the steps outline in https://github.com/kristinmlee/dmc/blob/master/dmc_example.md

Calculate neutral F matrix

For this first step (calculating the neutral variance/covariance matrix, F), I'll use linkage groups that have no or few shared outlier loci. I need to (1) Calculate allele frequencies for each population, (2) specify a vector of sample sizes for each population, and (3) specify a string for filename for output. This step is baseline for all parameter sensitivity analyses.

```
calc.allFreqs<-function(vcf,pop.list, pop.labs){
  allFreqs<-do.call(rbind,lapply(pop.list, function(pop){
    this.vcf<-cbind(vcf[,1:9],vcf$SNP,vcf[,grep(pop,colnames(vcf))])
    afs<-do.call(rbind,apply(this.vcf,1,calc.afs.vcf))
    return(afs$RefFreq)
  }))
  colnames(allFreqs)<-vcf$SNP
  rownames(allFreqs)<-pop.labs
  return(allFreqs)
}
#calculate selected allele frequencies
LG4<-calc.allFreqs(vcf[vcf$`#CHROM` == "LG4" & vcf$POS < 120000,],pop.list,pop.labs)
saveRDS(LG4,"dmc/selectedRegionAlleleFreqs_p4LG4.RDS")
saveRDS(vcf[vcf$`#CHROM`=="LG4","POS"],"dmc/selectedRegionPositions_p4LG4.RDS")

#calculate neutral allele frequencies
allFreqs<-calc.allFreqs(vcf[vcf$`#CHROM` %in%
                              c("LG3","LG5","LG6","LG7","LG8","LG9","LG11","LG15","LG16","LG17","LG19",
                    pop.list,pop.labs)
saveRDS(allFreqs,"dmc/neutralAlleleFreqs_p4LG4.RDS")
```

```
allFreqs<-readRDS("dmc/neutralAlleleFreqs_p4LG4.RDS")
```

Once those are calculated, I can calculate the neutral F matrix, in addition to its determinant and inverse.

```
sampleSizes<-unlist(lapply(pop.list,function(pop){
  n<-length(grep(pop,colnames(vcf)))
```

```
  return(2*n) }))
numPops<-16

#calculate the neutral F matrix
saveRDS(sampleSizes,"dmc/sampleSizes.RDS")
neutralF_filename<-"dmc/neutralF_p4LG4_50" #for 50 positions
source("../programs/dmc-master/calcNeutralF.R")
F_estimate<-readRDS("dmc/neutralF_p4LG4_50.RDS")

#calculate the determinant and inverse of the neutral F matrix
M <- numPops
Tmatrix <- matrix(data = rep(-1 / M, (M - 1) * M), nrow = M - 1, ncol = M)
diag(Tmatrix) = (M - 1) / M
sampleErrorMatrix = diag(1/sampleSizes, nrow = numPops, ncol = numPops)
det_FOmegas_neutral = det(Tmatrix %*% (F_estimate + sampleErrorMatrix) %*% t(Tmatrix))
saveRDS(det_FOmegas_neutral, "dmc/det_FOmegas_neutral_p4LG4_50.RDS")

inv_FOmegas_neutral = ginv(Tmatrix %*% (F_estimate + sampleErrorMatrix) %*% t(Tmatrix))
saveRDS(inv_FOmegas_neutral, "dmc/inv_FOmegas_neutral_p4LG4_50.RDS")
```

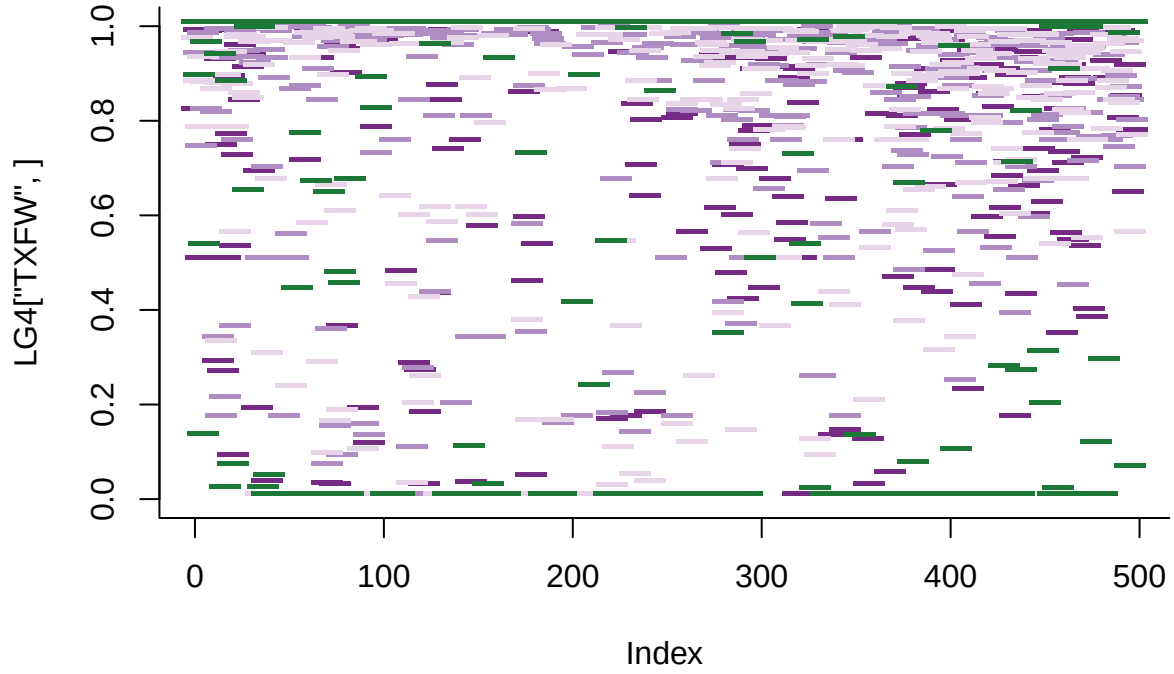
```
selSite = positions[seq(1, length(positions[positions<12000000]), length.out = 100)]
allFreqs<-readRDS("dmc/neutralAlleleFreqs_p4LG4.RDS")
#calculate the neutral F matrix
sampleSizes<-readRDS("dmc/sampleSizes.RDS")
neutralF_filename<-"dmc/neutralF_p4LG4_100" #for 50 positions
source("../programs/dmc-master/calcNeutralF.R")
F_estimate<-readRDS("dmc/neutralF_p4LG4_100.RDS")

#calculate the determinant and inverse of the neutral F matrix
M <- numPops
Tmatrix <- matrix(data = rep(-1 / M, (M - 1) * M), nrow = M - 1, ncol = M)
diag(Tmatrix) = (M - 1) / M
sampleErrorMatrix = diag(1/sampleSizes, nrow = numPops, ncol = numPops)
det_FOmegas_neutral = det(Tmatrix %*% (F_estimate + sampleErrorMatrix) %*% t(Tmatrix))
saveRDS(det_FOmegas_neutral, "dmc/det_FOmegas_neutral_p4LG4_100.RDS")

inv_FOmegas_neutral = ginv(Tmatrix %*% (F_estimate + sampleErrorMatrix) %*% t(Tmatrix))
saveRDS(inv_FOmegas_neutral, "dmc/inv_FOmegas_neutral_p4LG4_100.RDS")
```

Note: I had to use to allow read/write in the scripts.

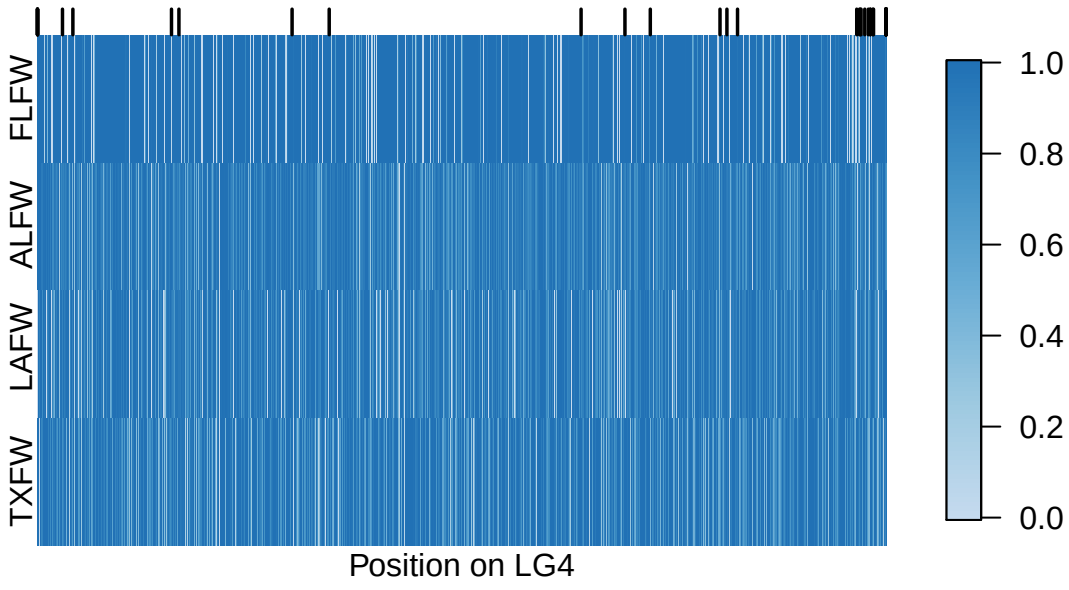
Let's look at the allele frequencies on LG4. This will help inform the hypotheses I test with dmc.

```
stacks.sig<-read.delim("p4.stacks.sig.snps.txt")
stacks.sig$SNP<-paste(stacks.sig$Chr,(stacks.sig$BP+1),sep=".")
LG4<-readRDS("dmc/selectedRegionAlleleFreqs_p4LG4.RDS")
allFreqs<-readRDS("dmc/neutralAlleleFreqs_p4LG4.RDS")
# LG4[rownames(LG4) %in% c("FLFW","ALFW","LAFW","TXFW"),colnames(LG4) %in% stacks.sig$SNP]
# LG4[rownames(LG4) %in% c("FLFW","ALFW","LAFW","TXFW"),
#          which(colnames(LG4) %in% stacks.sig$SNP)[1]:
#            which(colnames(LG4) %in% stacks.sig$SNP)[length(which(colnames(LG4) %in% stacks.sig$SNP))]]
grp.colors<-c('#762a83','#af8dc3','#e7d4e8','#d9f0d3','#7fbf7b','#1b7837')
plot(LG4["TXFW",],col=grp.colors[1],pch="-",bty="L",lwd=2,cex=2)
points(LG4["LAFW",],col=grp.colors[2],pch="-",bty="L",lwd=2,cex=2)
points(LG4["ALFW",],col=grp.colors[3],pch="-",bty="L",lwd=2,cex=2)
```

```
points(LG4["FLFW",],col=grp.colors[6],pch="-",bty="L",lwd=2,cex=2)
```




```
cols<-colorRampPalette(t(brewer.pal(9,name="Blues")[3:7]))(100)

fields::image.plot(t(allFreqs[rownames(allFreqs) %in% c("FLFW","ALFW","LAFW","TXFW"),]),col=cols,
      axes=FALSE)
text(x=-0.02,y=1,"FLFW",srt=90,xpd=TRUE)
text(x=-0.02,y=0.67,"ALFW",srt=90,xpd=TRUE)
text(x=-0.02,y=0.35,"LAFW",srt=90,xpd=TRUE)
text(x=-0.02,y=0,"TXFW",srt=90,xpd=TRUE)
mtext("Position on LG4",1)
axis(3,at =t(LG4[rownames(LG4) %in% c("FLFW","ALFW","LAFW","TXFW"),colnames(LG4) %in% stacks.sig$SNP]),
     labels = FALSE,lwd=0,lwd.ticks = 1.75,tck=-.05)
```



This shows that the FLFW site has somewhat different allele frequencies than the other sites. This suggests that I should test to see whether there was an independent mutation in the FLFW population and the other three have a single sweep.

Calculate F(S) matrices

We generate matrices for the following five models: *1. All selected pops have independent mutations of beneficial allele* 2. All selected pops share beneficial allele via migration *3. Beneficial allele was standing the ancestor of all selected pops* 4. There was an independent mutation in FLFW and the other three share a sweep. *5. FLFW has an independent mutation and the others share a beneficial allele via standing variation

I've written a giant function to run all five models and calculate their composite likelihoods. The composite likelihoods include knowing how far away we are from the proposed selected site. Therefore, the first step for calculating them is to randomize with respect to the reference allele.

The inverses and determinants generated can be used to calculate the log-likelihood of a site a given distance away from a proposed selected site. To do this, we sum the log-likelihoods across all SNPs in the window to obtain a composite log-likelihood under a given convergence model with a set of parameters for a proposed selected site.

```
run.dmc<-function(F_estimate,out_name,positions, sampleSizes,selSite=NA,nselsites=50,rec =2.17*10^-8,
                  Ne = 8.3*10^6,selPops = c(3,5,7,16),numBins = 1000,numPops = 16,
                  sels = c(1e-4, 1e-3, 0.01, seq(0.02, 0.14, by = 0.01), seq(0.15, 0.3, by = 0.05),
                           seq(0.4, 0.6, by = 0.1)),
                  times = c(0, 5, 25, 50, 100, 500, 1000, 1e4, 1e6),
                  migs = c(10^-(seq(5, 1, by = -2)), 0.5, 1),mod4_sets=list(c(3,5,7),16),
                  mod1=TRUE,mod2=TRUE,mod3=TRUE,mod4=TRUE,mod5=TRUE,complike=TRUE,
                  neutral_det_name="dmc/det_FOmegas_neutral_p4LG4.RDS",
                  neutral_inv_name="dmc/inv_FOmegas_neutral_p4LG4.RDS"){
  #make all the parameters global
  F_estimate<<-F_estimate
  positions<<-positions
  sampleSizes<<-sampleSizes
  Ne<<-Ne
  rec<<-rec
  selPops<<-selPops
  numBins<<-numBins
  numPops<<-numPops
  sels<<-sels
  times<<-times
  migs<<-migs
  neutral_det_name<<-neutral_det_name
  neutral_inv_name<<-neutral_inv_name
  print(paste(neutral_det_name,neutral_inv_name,sep=","))
  #set up parameters
  M <<- numPops
  Tmatrix <<- matrix(data = rep(-1 / M, (M - 1) * M), nrow = M - 1, ncol = M)
  diag(Tmatrix) = (M - 1) / M
  gs<<-c(1/(2*Ne), 10^-(4:1))
  sampleErrorMatrix <<- diag(1/sampleSizes, nrow = numPops, ncol = numPops)
  if(is.na(selSite[1])) selSite=seq(min(positions), max(positions), length.out = nselsites)
  selSite<<-selSite
  sources <<- selPops
  sets<<-mod4_sets
```

```
#save params
params<-list(F_estimate,out_name,positions, sampleSizes,selSite,rec,
             Ne,selPops,numBins,numPops,sels,times,gs,migs,mod4_sets)
names(params)<-c("F_estimate","out_name","positions", "sampleSizes","selSite","rec",
                 "Ne","selPops","numBins","numPops","sels","times","gs","migs","mod4_sets")
print(params)
################ calculate F(S) ################
source("../programs/dmc-master/genSelMatrices_individualModes.R")

################ model 1 ################
if(mod1==TRUE){
  FOmegas_ind = lapply(sels, function(sel) {
    calcFOmegas_indSweeps(sel)
  })

  saveRDS(FOmegas_ind, paste("dmc/FOmegas_ind_",out_name,".RDS",sep=""))
  #model 1 determinant
  det_FOmegas_ind = lapply(FOmegas_ind, function(sel) {
    lapply(sel, function(dist) {
      det(dist)
    })
  })
  saveRDS(det_FOmegas_ind, paste("dmc/det_FOmegas_ind_",out_name,".RDS",sep=""))
  #model 1 inverse
  inv_FOmegas_ind = lapply(FOmegas_ind, function(sel) {
    lapply(sel, function(dist) {
      ginv(dist)
    })
  })
  saveRDS(inv_FOmegas_ind, paste("dmc/inv_FOmegas_ind_",out_name,".RDS",sep=""))
}
################ model 2 ################
if(mod2==TRUE){
  FOmegas_mig = lapply(sels ,function(sel) {
    lapply(migs, function(mig) {
      lapply(sources, function(my.source) {
        calcFOmegas_mig(sel, mig, my.source)
      })
    })
  })

  saveRDS(FOmegas_mig, paste("dmc/FOmegas_mig_",out_name,".RDS",sep=""))
  #determinant
  det_FOmegas_mig = lapply(FOmegas_mig, function(sel) {
    lapply(sel, function(mig) {
      lapply(mig, function(source) {
        lapply(source, function(dist) {
          det(dist)
        })
      })
    })
  })
  saveRDS(det_FOmegas_mig, paste("dmc/det_FOmegas_mig_",out_name,".RDS",sep=""))
```

```
  #inverse
  inv_FOmegas_mig = lapply(FOmegas_mig, function(sel) {
    lapply(sel, function(mig) {
      lapply(mig, function(source) {
        lapply(source, function(dist) {
          ginv(dist)
        })
      })
    })
  })
  saveRDS(inv_FOmegas_mig, paste("dmc/inv_FOmegas_mig_",out_name,".RDS",sep=""))
}
############### model 3 ###############
if(mod3==TRUE)
{
  FOmegas_sv = lapply(sels, function(sel) {
    lapply(gs, function(g) {
      lapply(times, function(time) {
        lapply(sources, function(my.source) {
          calcFOmegas_stdVar.source(sel, g, time, my.source)
        })
      })
    })
  })

  saveRDS(FOmegas_sv, paste("dmc/FOmegas_sv_",out_name,".RDS",sep=""))
  #determinant
  det_FOmegas_sv = lapply(FOmegas_sv, function(sel) {
    lapply(sel, function(g) {
      lapply(g, function(time) {
        lapply(time, function(my.source) {
          lapply(my.source, function(dist) {
            det(dist)
          })
        })
      })
    })
  })
  saveRDS(det_FOmegas_sv, paste("dmc/det_FOmegas_sv_",out_name,".RDS",sep=""))
  #inverse
  inv_FOmegas_sv = lapply(FOmegas_sv, function(sel) {
    lapply(sel, function(g) {
      lapply(g, function(time) {
        lapply(time, function(my.source) {
          lapply(my.source, function(dist) {
            ginv(dist)
          })
        })
      })
    })
  })
  saveRDS(inv_FOmegas_sv, paste("dmc/inv_FOmegas_sv_",out_name,".RDS",sep=""))
}
```

```
############### model 4 ###############
if(mod4==TRUE){

  source("../programs/dmc-master/genSelMatrices_multipleModes.R")

  my.modes_migInd=c("mig","ind")

  #the parameters time and g are not involved in the migration model so we only loop over
  ## the first element of these vectors
  FOmegas_mixed_migInd = lapply(sels ,function(sel) {
    lapply(gs[1], function(g) {
      lapply(times[1], function(time) {
        lapply(migs, function(mig) {
          lapply(sources, function(my.source) {
            calcFOmegas_mixed(sel, g, time, mig, my.source, my.modes_migInd)
          })
        })
      })
    })
  })

  saveRDS(FOmegas_mixed_migInd, paste("dmc/FOmegas_mixed_migInd_",out_name,".RDS",sep=""))

  detFOmegas_mixed_migInd = lapply(FOmegas_mixed_migInd, function(sel) {
    lapply(sel, function(g) {
      lapply(g, function(time) {
        lapply(time, function(mig) {
          lapply(mig, function(source) {
            lapply(source, function(dist) {
              det(dist)
            })
          })
        })
      })
    })
  })
  saveRDS(detFOmegas_mixed_migInd, paste("dmc/det_FOmegas_mixed_migInd_",out_name,".RDS",sep=""))

  invFOmegas_mixed_migInd = lapply(FOmegas_mixed_migInd, function(sel) {
    lapply(sel, function(g) {
      lapply(g, function(time) {
        lapply(time, function(mig) {
          lapply(mig, function(source) {
            lapply(source, function(dist) {
              ginv(dist)
            })
          })
        })
      })
    })
  })
  saveRDS(invFOmegas_mixed_migInd, paste("dmc/inv_FOmegas_mixed_migInd_",out_name,".RDS",sep=""))
}
```

```
############### model 5 ###############
if(mod5==TRUE){
  my.modes_svInd = c("sv", "ind")

  #the parameter mig is not involved in the standing variant model so we only loop over
  ## the first element of this vector
  FOmegas_mixed_svInd = lapply(sels ,function(sel) {
    lapply(gs, function(g) {
      lapply(times, function(time) {
        lapply(migs[1], function(mig) {
          lapply(sources, function(my.source) {
            calcFOmegas_mixed(sel, g, time, mig, my.source, my.modes_svInd)
          })
        })
      })
    })
  })

  saveRDS(FOmegas_mixed_svInd, paste("dmc/FOmegas_mixed_svInd_",out_name,".RDS",sep=""))

  detFOmegas_mixed_svInd = lapply(FOmegas_mixed_svInd, function(sel) {
    lapply(sel, function(g) {
      lapply(g, function(time) {
        lapply(time, function(mig) {
          lapply(mig, function(source) {
            lapply(source, function(dist) {
              det(dist)
            })
          })
        })
      })
    })
  })
  saveRDS(detFOmegas_mixed_svInd, paste("dmc/det_FOmegas_mixed_svInd_",out_name,".RDS",sep=""))

  invFOmegas_mixed_svInd = lapply(FOmegas_mixed_svInd, function(sel) {
    lapply(sel, function(g) {
      lapply(g, function(time) {
        lapply(time, function(mig) {
          lapply(mig, function(source) {
            lapply(source, function(dist) {
              ginv(dist)
            })
          })
        })
      })
    })
  })
  saveRDS(invFOmegas_mixed_svInd, paste("dmc/inv_FOmegas_mixed_svInd_",out_name,".RDS",sep=""))
}
############### calculate composite likelihoods ###############
if(complike==TRUE){
  #randomize allele freqs
```

```
freqs_notRand = readRDS("dmc/selectedRegionAlleleFreqs_p4LG4.RDS")
randFreqs = apply(freqs_notRand, 2, function(my.freqs) {
  if(runif(1) < 0.5) {
    my.freqs = 1 - my.freqs
  }
  my.freqs
})
saveRDS(randFreqs, paste("dmc/selectedRegionAlleleFreqsRand_",out_name,".RDS",sep=""))
freqs<<-randFreqs

#calc the likelihoods
source("../programs/dmc-master/calcCompositeLike.R")

## Neutral model

det_FOmegas_neutral = readRDS(neutral_det_name)
inv_FOmegas_neutral = readRDS(neutral_inv_name)
compLikelihood_neutral = lapply(1 : length(selSite), function(j) {
  calcCompLikelihood_neutral(j, det_FOmegas_neutral, inv_FOmegas_neutral)
})
saveRDS(compLikelihood_neutral, paste("dmc/compLikelihood_neutral_",out_name,".RDS",sep=""))

## Model 1
det_FOmegas_ind = readRDS(paste("dmc/det_FOmegas_ind_",out_name,".RDS",sep=""))
inv_FOmegas_ind = readRDS(paste("dmc/inv_FOmegas_ind_",out_name,".RDS",sep=""))
compLikelihood_ind = lapply(1 : length(selSite), function(j) {
  lapply(1 : length(sels), function(sel) calcCompLikelihood_1par(j, det_FOmegas_ind,
                                                                 inv_FOmegas_ind, sel))
})
saveRDS(compLikelihood_ind, paste("dmc/compLikelihood_ind_",out_name,".RDS",sep=""))

## Model 2
det_FOmegas_mig = readRDS(paste("dmc/det_FOmegas_mig_",out_name,".RDS",sep=""))
inv_FOmegas_mig = readRDS(paste("dmc/inv_FOmegas_mig_",out_name,".RDS",sep=""))
compLikelihood_mig = lapply(1 : length(selSite), function(j) {
  lapply(1 : length(sels), function(sel) {
    lapply(1 : length(migs), function(mig) {
      lapply(1 : length(sources), function(my.source) {
        calcCompLikelihood_3par(j, det_FOmegas_mig, inv_FOmegas_mig, sel, mig,
                                my.source)
      })
    })
  })
})
saveRDS(compLikelihood_mig, paste("dmc/compLikelihood_mig_",out_name,".RDS",sep=""))

## Model 3
det_FOmegas_sv = readRDS(paste("dmc/det_FOmegas_sv_",out_name,".RDS",sep=""))
inv_FOmegas_sv = readRDS(paste("dmc/inv_FOmegas_sv_",out_name,".RDS",sep=""))
compLikelihood_sv = lapply(1 : length(selSite), function(j) {
  lapply(1 : length(sels), function(sel) {
    lapply(1 : length(gs), function(g) {
      lapply(1 : length(times), function(t) {
```

```
        lapply(1: length(sources), function(my.source) {
          calcCompLikelihood_4par(j, det_FOmegas_sv, inv_FOmegas_sv, sel, g, t,
                                  my.source)
        })
      })
    })
  })
})
saveRDS(compLikelihood_sv, paste("dmc/compLikelihood_sv_",out_name,".RDS",sep=""))

## Model 4
det_FOmegas_mixed_migInd = readRDS(paste("dmc/det_FOmegas_mixed_migInd_",out_name,".RDS",sep=""))
inv_FOmegas_mixed_migInd = readRDS(paste("dmc/inv_FOmegas_mixed_migInd_",out_name,".RDS",sep=""))

# same trick as above (the parameters time and g are not involved in the migration
## model so we only loop over the first element of these vectors)
# now save lists for each proposed selected site (may want to do this for other
## models/more elegantly depending on density of parameter space)
for(j in 1 : length(selSite)) {
  compLikelihood_mixed_migInd = lapply(1 : length(sels), function(sel) {
    lapply(1 : length(gs[1]), function(g) {
      lapply(1 : length(times[1]), function(t) {
        lapply(1 : length(migs), function (mig) {
          lapply(1: length(sources), function(my.source) {
            calcCompLikelihood_5par(j, det_FOmegas_mixed_migInd,
                                    inv_FOmegas_mixed_migInd, sel, g, t, mig,
                                    my.source)
          })
        })
      })
    })
  })
  saveRDS(compLikelihood_mixed_migInd,
          paste("dmc/compLikelihood_mixed_migInd_",out_name,"_selSite", j, ".RDS",
                sep = ""))
}

## Model 5
det_FOmegas_mixed_svInd = readRDS(paste("dmc/det_FOmegas_mixed_svInd_",out_name,".RDS",sep=""))
inv_FOmegas_mixed_svInd = readRDS(paste("dmc/inv_FOmegas_mixed_svInd_",out_name,".RDS",sep=""))

#same trick as above (the parameter mig is not involved in the migration model so we
##only loop over the first element of this vector)
# now save lists for each proposed selected site (may want to do this for other
## models/more elegantly depending on density of parameter space)
for(j in 1 : length(selSite)) {
  compLikelihood_mixed_svInd = lapply(1 : length(sels), function(sel) {
    lapply(1 : length(gs), function(g) {
      lapply(1 : length(times), function(t) {
        lapply(1 : length(migs[1]), function (mig) {
          lapply(1: length(sources), function(my.source) {
            calcCompLikelihood_5par(j, det_FOmegas_mixed_svInd,
                                    inv_FOmegas_mixed_svInd, sel, g, t, mig,
```

```
                                        my.source)
            })
          })
        })
      })
    })
    saveRDS(compLikelihood_mixed_svInd,
            paste("dmc/compLikelihood_mixed_svInd_",out_name,"_selSite", j, ".RDS",
                  sep = ""))
  }

  #combine models 4 and 5 output

  ## Model 4
  compLikelihood_mixed_migInd_all = lapply(1: length(selSite), function(i) {
    readRDS(paste("dmc/compLikelihood_mixed_migInd_",out_name,"_selSite", i, ".RDS",
                  sep = ""))
  })

  saveRDS(compLikelihood_mixed_migInd_all, paste("dmc/compLikelihood_mixed_migInd_",out_name,".RDS",s

  ## Model 5
  compLikelihood_mixed_svInd_all = lapply(1: length(selSite), function(i) {
    readRDS(paste("dmc/compLikelihood_mixed_svInd_",out_name,"_selSite", i, ".RDS",
                  sep = ""))
  })

  saveRDS(compLikelihood_mixed_svInd_all, paste("dmc/compLikelihood_mixed_svInd_",out_name,".RDS",sep
 }
 return(params)
}
```

I have to set some parameters before running the function, including the sets of parameters I want to try different settings for.

```
################ Set parameters ################
positions<-readRDS("dmc/selectedRegionPositions_p4LG4.RDS")
F_estimate<-readRDS("dmc/neutralF_p4LG4_50.RDS")
sampleSizes<-readRDS("dmc/sampleSizes.RDS")

numPops = 16
numPops <- 16
selPops <- c(3,5,7,16)
numBins <- 1000

selSite = positions[seq(1, length(positions), length.out = 50)]
sels = c(1e-4, 1e-3, 0.01, seq(0.02, 0.14, by = 0.01), seq(0.15, 0.3, by = 0.05),
         seq(0.4, 0.6, by = 0.1))
times = c(0, 5, 25, 50, 100, 500, 1000, 1e4, 1e6)
migs = c(10^-(seq(5, 1, by = -2)), 0.5, 1)
mod4_sets=list(c(3,5,7),16)
```

And I'm going to run it with a variety of effective population sizes and recombination rates to see how robust the models are to that parameter.

```
############### Run the program ###############
Nes <- c(8.3*10^3,8.3*10^4,8.3*10^5,8.3*10^6)
rs<-c(2.17*10^-8,2*10^-7,6*10^-6,5.6*10^-6)

dmc.out<-lapply(rs,function(r){
  rec<-r
  rname<-gsub("(\\d).*(\\d)$","\\1_\\2",as.character(r))
  lapply(Nes, function(ne){
    out_name<-paste("p4LG4_",ne,"_",rname,sep="")
    dir<-getwd()
    print(paste("running",out_name,"in",dir,sep=" "))
    p<-run.dmc(F_estimate = F_estimate,out_name = out_name,positions = positions,sampleSizes = sampleSi
               selSite=selSite,rec =rec,
               Ne = ne,selPops = selPops,numBins = numBins,numPops = numPops,
               sels = sels, times = times,
               migs = migs,mod4_sets=mod4_sets,
               neutral_det_name="dmc/det_FOmegas_neutral_p4LG4_50.RDS",
               neutral_inv_name = "dmc/inv_FOmegas_neutral_p4LG4_50.RDS")
    return(p)
  })
})
print("Done running dmc")
```

I'm running this one on abies using `nohup Rscript ~/Projects/popgen/scripts/002_run_dmc.R >> ~/Projects/popgen/dmc_ne.log 2>&1 &`.

Estimate recombination rates

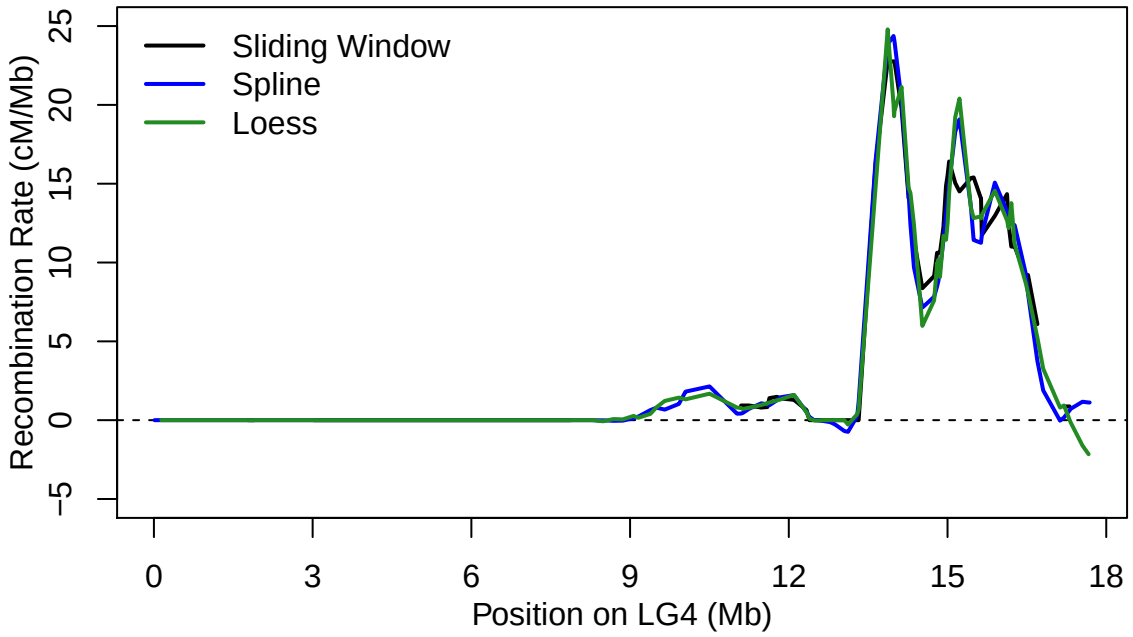
```
ssc.map<-read.delim("../../scovelli_genome/SSC_linkage_map_20160701.txt",header=T)
colnames(ssc.map)<-c("map","mkr","gen","scaffold","scaffold_orientation","scaffold.bp","phys","node")
ssc.map$set<-"Syngnathus.scovelli"
map<-ssc.map[,c("set","map","mkr","phys","gen")]
write.table(map,"../../scovelli_genome/ssc_map.txt",sep=" ",row.names=FALSE, col.names = TRUE)
library(MareyMap)
startMareyMapGUI()
#using the GUI, I ran all three types of recombination rate estimators and saved the MapSet
```

```
ssc.map<-read.delim("../../scovelli_genome/ssc_MareyMap.txt",comment.char = "#",sep=" ")
#genome-wide
r<-data.frame(Genome_wide=c(mean(ssc.map$slidingwindow,na.rm = TRUE)/1000000,
                            mean(ssc.map$spline,na.rm = TRUE)/1000000,
                            mean(ssc.map$loess,na.rm = TRUE)/1000000),
              LG4=c(mean(ssc.map$slidingwindow[ssc.map$map=="LG4"],na.rm = TRUE)/1000000,
                            mean(ssc.map$spline[ssc.map$map=="LG4"],na.rm = TRUE)/1000000,
                            mean(ssc.map$loess[ssc.map$map=="LG4"],na.rm = TRUE)/1000000),
              LG4_first12Mb=c(mean(ssc.map$slidingwindow[ssc.map$map=="LG4" & ssc.map$phys<12*10^6],na.
                            mean(ssc.map$spline[ssc.map$map=="LG4"& ssc.map$phys<12*10^6],na.rm = TRUE),
                            mean(ssc.map$loess[ssc.map$map=="LG4"& ssc.map$phys<12*10^6],na.rm = TRUE)/
              LG4_first10Mb=c(mean(ssc.map$slidingwindow[ssc.map$map=="LG4" & ssc.map$phys<10*10^6],na.
                            mean(ssc.map$spline[ssc.map$map=="LG4"& ssc.map$phys<10*10^6],na.rm = TRUE),
                            mean(ssc.map$loess[ssc.map$map=="LG4"& ssc.map$phys<10*10^6],na.rm = TRUE)/
              row.names = c("Sliding Window","Spline","Loess"))
kable(r,caption="Recombination rate estimates")
```

Table 1: Recombination rate estimates

	Genome_wide	LG4	LG4_first12Mb	LG4_first10Mb
Sliding Window	5.8e-06	3.7e-06	2e-07	0
Spline	5.6e-06	3.0e-06	2e-07	0
Loess	5.6e-06	3.0e-06	2e-07	0

```
load("../../scovelli_genome/ssc_map.rda")
#plot LG4
plot(set[["LG4"]]@interpolations$slidingwindow@physicalPositions,
     set[["LG4"]]@interpolations$slidingwindow@rates,type="l",
     ylim=c(-5,25),xlab="",ylab="",xaxt='n',lwd=2)
abline(h=0,lty=2)
axis(1,at=seq(0,1.8*10^7,3*10^6),labels = seq(0,18,3))
mtext("Position on LG4 (Mb)",1,line=2)
mtext("Recombination Rate (cM/Mb)",2,line=2)
lines(set[["LG4"]]@interpolations$spline@physicalPositions,
      set[["LG4"]]@interpolations$spline@rates,type="l",col="blue",lwd=2)
lines(set[["LG4"]]@interpolations$loess@physicalPositions,
      set[["LG4"]]@interpolations$loess@rates,type="l",lwd=2,col="forestgreen")
legend("topleft",bty='n',col=c("black","blue","forestgreen"),lwd=2,c("Sliding Window","Spline","Loess"))
```



Evaluate the output of the initial dmc runs

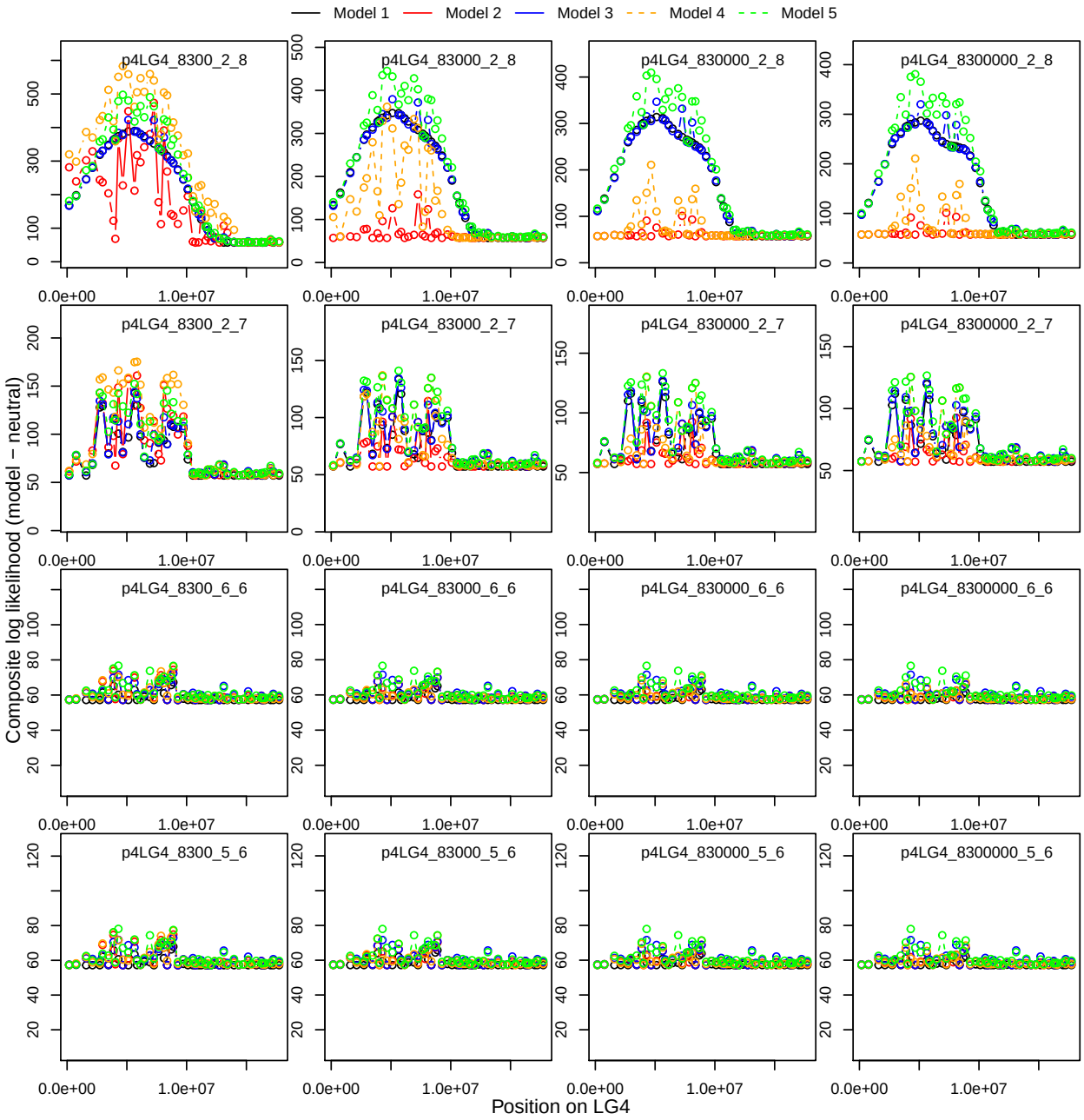
Plot maximum composite likelihood ratios for models over proposed selected sites

```
calc.max.complike<-function(pattern,neutral_name=NA){
  #read in composite likelihood files and calculate max for all proposed selected sites
  if(is.na(neutral_name)){
```

```
    compLikelihood_neutral = readRDS(paste("dmc/compLikelihood_neutral_",pattern,".RDS",sep=""))
  }else{
    compLikelihood_neutral = readRDS(neutral_name)
  }

  compLikelihood_neutral_site = sapply(1 : length(compLikelihood_neutral), function(i) {
    max(unlist(compLikelihood_neutral[[i]]))
  })

  compLikelihood_ind = readRDS(paste("dmc/compLikelihood_ind_",pattern,".RDS",sep=""))
  compLikelihood_ind_site = sapply(1 : length(compLikelihood_ind), function(i) {
    max(unlist(compLikelihood_ind[[i]]))
  })

  compLikelihood_mig = readRDS(paste("dmc/compLikelihood_mig_",pattern,".RDS",sep=""))
  compLikelihood_mig_site = sapply(1 : length(compLikelihood_mig), function(i) {
    max(unlist(compLikelihood_mig[[i]]))
  })

  compLikelihood_sv = readRDS(paste("dmc/compLikelihood_sv_",pattern,".RDS",sep=""))
  compLikelihood_sv_site = sapply(1 : length( compLikelihood_sv), function(i) {
    max(unlist(compLikelihood_sv[[i]]))
  })

  compLikelihood_mixed_migInd = readRDS(paste("dmc/compLikelihood_mixed_migInd_",pattern,".RDS",sep=""))
  compLikelihood_mixed_migInd_site = sapply(1 : length(compLikelihood_mixed_migInd), function(i) {
    max(unlist(compLikelihood_mixed_migInd[[i]]))
  })

  compLikelihood_mixed_svInd = readRDS(paste("dmc/compLikelihood_mixed_svInd_",pattern,".RDS",sep=""))
  compLikelihood_mixed_svInd_site = sapply(1 : length(compLikelihood_mixed_svInd), function(i) {
    max(unlist(compLikelihood_mixed_svInd[[i]]))
  })
  return(list(max.likes=c(ind=(compLikelihood_ind_site - compLikelihood_neutral_site),
           mig=(compLikelihood_mig_site - compLikelihood_neutral_site),
           sv=(compLikelihood_sv_site - compLikelihood_neutral_site),
           migInd=(compLikelihood_mixed_migInd_site - compLikelihood_neutral_site),
           svInd=(compLikelihood_mixed_svInd_site - compLikelihood_neutral_site)),
         max.complikes=c(neutral=compLikelihood_neutral,ind=compLikelihood_ind,
           mig=compLikelihood_mig,
           sv=compLikelihood_sv,
           migInd=compLikelihood_mixed_migInd,
           svInd=compLikelihood_mixed_svInd)))
}

plot.complike<-function(pattern,selSite,leg=TRUE,lab=TRUE){
  max.complikes<-calc.max.complike(pattern)
  max.likes<-max.complikes[[1]]
  plot_range = range(max.likes)

  if(lab==TRUE){
    x.lab<-"Proposed position selected site"
    y.lab<-"Composite log-likelihood (model - neutral)"
```

```
  } else{
    x.lab<-y.lab<-""
  }

  plot(selSite, max.likes[grep("ind",names(max.likes))], type = "b",
       ylim = c(plot_range[1] - 50, plot_range[2] + 50),
       xlab = x.lab,
       ylab = y.lab)
  lines(selSite, max.likes[grep("mig\\d+",names(max.likes))], col = "red",
        type = "b")
  lines(selSite, max.likes[grep("sv\\d+",names(max.likes))], col = "blue",
        type = "b")
  lines(selSite, max.likes[grep("migInd",names(max.likes))],
        col = "orange", lty = 2, type = "b")
  lines(selSite, max.likes[grep("svInd",names(max.likes))],
        col = "green", lty = 2, type = "b")
  legend("top",bty='n',legend = pattern)
  if(isTRUE(leg)){
    legend("topright", col = c("black", "red", "blue", "orange", "green"),
           lty = c(rep(1, 3), rep(2, 2)), sapply(1 : 5, function(i) paste("Model", i)))
  }
  return(list(max.complikes[[2]],max.likes,pattern))
}
```

```
patterns<-c("p4LG4_8300_2_8","p4LG4_83000_2_8","p4LG4_830000_2_8","p4LG4_8300000_2_8",
            "p4LG4_8300_2_7","p4LG4_83000_2_7","p4LG4_830000_2_7","p4LG4_8300000_2_7",
            "p4LG4_8300_6_6","p4LG4_83000_6_6","p4LG4_830000_6_6","p4LG4_8300000_6_6",
            "p4LG4_8300_5_6","p4LG4_83000_5_6","p4LG4_830000_5_6","p4LG4_8300000_5_6")
selSite = positions[seq(1, length(positions), length.out = 50)]
par(mfrow=c(4,4),mar=c(1.5,1.5,0.5,0.5),oma=c(2,2,2,2))
ne.complikes<-lapply(patterns,plot.complike,leg=FALSE,lab=FALSE,selSite=selSite)
mtext("Position on LG4",1,outer=TRUE,cex=0.8,line=.5)
mtext("Composite log likelihood (model - neutral)",2,outer=T,cex=0.8,line=.5)
par(fig=c(0,1,0,1),oma=c(0,0,0,0),mar=c(0,0,0,0),new=TRUE)
plot(0,0,type='n',bty='n',xaxt='n',yaxt='n')
legend("top", col = c("black", "red", "blue", "orange", "green"),ncol=5,bty='n',
          lty = c(rep(1, 3), rep(2, 2)), sapply(1 : 5, function(i) paste("Model", i)))
```



Get maximum composite likelihood estimates

```
#These won't work if the parameters aren't specified as above
get.mcle<-function(max.complikes,Ne,rec,selSite= positions[seq(1, length(positions), length.out = 50)],
                   sels = c(1e-4, 1e-3, 0.01, seq(0.02, 0.14, by = 0.01), seq(0.15, 0.3, by = 0.05),
                            seq(0.4, 0.6, by = 0.1)),
                   times = c(0, 5, 25, 50, 100, 500, 1000, 1e4, 1e6),
                   migs = c(10^-(seq(5, 1, by = -2)), 0.5, 1),sources=c(3,5,7,16)){
  Ne<<-Ne
  rec<<-rec
  gs = c(1/(2*Ne), 10^-(4:1))
```

```
  source("../programs/dmc-master/getMCLE.R")

  ## Model 1
  m1<-cbind(getMCLEind(max.complikes[grep("ind",names(max.complikes))], selSite, sels),
            maxG=NA,maxTime=NA,maxSource=NA)

  ## Model 2
  m2<-cbind(getMCLEmig(max.complikes[grep("mig\\d+",names(max.complikes))], selSite, sels, migs, sources
            maxG=NA)

  ## Model 3
  m3<-getMCLEsv_source(max.complikes[grep("sv\\d+",names(max.complikes))], selSite, sels, gs, times, sou

  ## Model 4
  m4<-getMCLEmixed(max.complikes[grep("migInd",names(max.complikes))], selSite, sels, gs[1], times[1], m

  ## Model 5
  m5<-getMCLEmixed(max.complikes[grep("svInd",names(max.complikes))], selSite, sels, gs, times, migs[1]
  mcles<-rbind(m1,m2,m3,m4,m5)
  rownames(mcles)<-c("Model1","Model2","Model3","Model4","Model5")
  return(mcles)
}
```

```
rs<-c("2_8"=2.17*10^-8,"2_7"=2*10^-7,"6_6"=6*10^-6,"5_6"=5.6*10^-6)
ne.mcles<-lapply(ne.complikes, function(m.complikes){
  pattern<-m.complikes[[3]]
  print(pattern)
  Ne<-as.numeric(gsub("p4LG4_(\\d+)_.*","\\1",pattern))
  r<-gsub("p4LG4_\\d+_(\\d_\\d)","\\1",pattern)
  rec<-rs[r]
  max.comp<-m.complikes[[1]]
  mcle<-get.mcle(max.comp,Ne=Ne,rec=rec)
  return(list(mcle,Ne,rec))
})
```

```
## [1] "p4LG4_8300_2_8"
## [1] "p4LG4_83000_2_8"
## [1] "p4LG4_830000_2_8"
## [1] "p4LG4_8300000_2_8"
## [1] "p4LG4_8300_2_7"
## [1] "p4LG4_83000_2_7"
## [1] "p4LG4_830000_2_7"
## [1] "p4LG4_8300000_2_7"
## [1] "p4LG4_8300_6_6"
## [1] "p4LG4_83000_6_6"
## [1] "p4LG4_830000_6_6"
## [1] "p4LG4_8300000_6_6"
## [1] "p4LG4_8300_5_6"
## [1] "p4LG4_83000_5_6"
## [1] "p4LG4_830000_5_6"
## [1] "p4LG4_8300000_5_6"
```

```r
names(ne.mcles)<-patterns
lapply(ne.mcles,function(mcle){ kable(mcle[[1]]) })
```

```
## $p4LG4_8300_2_8
##
##
##            maxLoc   maxSel       maxG   maxTime   maxSource
## -------  --------  -------  ---------  --------  ----------
## Model1    5641459      0.6         NA        NA          NA
## Model2    7263580      0.6   1.00e-05         3          NA
## Model3    5122070      0.6   1.00e-04        50           5
## Model4    4665911      0.6   6.02e-05         0           5
## Model5    4665911      0.6   1.00e-04         5           3
##
## $p4LG4_83000_2_8
##
##
##            maxLoc   maxSel    maxG   maxTime   maxSource
## -------  --------  -------  ------  --------  ----------
## Model1    5122070      0.6      NA        NA          NA
## Model2    7263580      0.4   1e-05         3          NA
## Model3    5122070      0.6   6e-06        50           5
## Model4    4665911      0.6   6e-06         0           3
## Model5    4665911      0.6   6e-06         5           3
##
## $p4LG4_830000_2_8
##
##
##            maxLoc   maxSel    maxG   maxTime   maxSource
## -------  --------  -------  ------  --------  ----------
## Model1    5122070     0.60      NA        NA          NA
## Model2    7263580     0.09   1e+00         5          NA
## Model3    5122070     0.60   6e-07        50           5
## Model4    4665911     0.30   6e-07         0           7
## Model5    4665911     0.60   6e-07        25           3
##
## $p4LG4_8300000_2_8
##
##
##            maxLoc   maxSel    maxG   maxTime   maxSource
## -------  --------  -------  ------  --------  ----------
## Model1    5122070      0.6      NA        NA          NA
## Model2    7263580      0.1   1e+00         5          NA
## Model3    5122070      0.6   1e-07        50           5
## Model4    4665911      0.3   1e-07         0           7
## Model5    4665911      0.6   1e-07        25           3
##
## $p4LG4_8300_2_7
##
##
##            maxLoc   maxSel       maxG   maxTime   maxSource
## -------  --------  -------  ---------  --------  ----------
## Model1    5641459      0.6         NA        NA          NA
## Model2    5846714      0.6   1.00e-05         5          NA
```

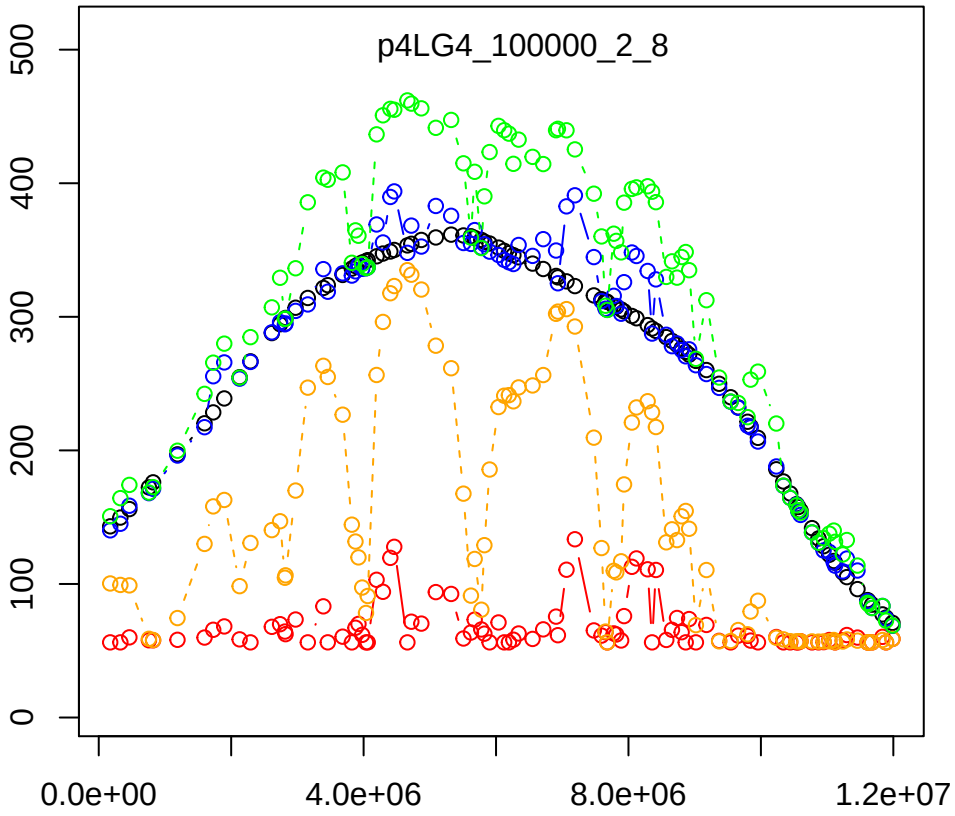
```
## Model3    5641459      0.6   1.00e-04       100           3
## Model4    5846714      0.6   6.02e-05         0           5
## Model5    5641459      0.6   1.00e-04        25           5
##
## $p4LG4_83000_2_7
##
##
##            maxLoc   maxSel    maxG   maxTime   maxSource
## -------  --------  -------  ------  --------  ----------
## Model1    5641459      0.6      NA        NA          NA
## Model2    8097311      0.6   1e-05         5          NA
## Model3    5641459      0.6   6e-06       100           3
## Model4    4291898      0.6   6e-06         0           7
## Model5    5641459      0.6   6e-06        50           5
##
## $p4LG4_830000_2_7
##
##
##            maxLoc   maxSel    maxG   maxTime   maxSource
## -------  --------  -------  ------  --------  ----------
## Model1    5641459      0.6      NA        NA          NA
## Model2    8097311      0.5   1e+00         5          NA
## Model3    5641459      0.6   6e-07       100           3
## Model4    4291898      0.6   6e-07         0           7
## Model5    5641459      0.6   6e-07        50           5
##
## $p4LG4_8300000_2_7
##
##
##            maxLoc   maxSel    maxG   maxTime   maxSource
## -------  --------  -------  ------  --------  ----------
## Model1    5641459      0.6      NA        NA          NA
## Model2    8097311      0.6   1e+00         5          NA
## Model3    5641459      0.6   1e-07       100           3
## Model4    4291898      0.6   1e-07         0           7
## Model5    5641459      0.6   1e-07        50           5
##
## $p4LG4_8300_6_6
##
##
##            maxLoc   maxSel       maxG   maxTime   maxSource
## -------  --------  -------  ---------  --------  ----------
## Model1    7856956      0.6         NA        NA          NA
## Model2    8889185      0.6   1.00e-05         5          NA
## Model3    8889185      0.6   1.00e-04         5           3
## Model4    8889185      0.6   6.02e-05         0           3
## Model5    4291898      0.6   1.00e-01         0           5
##
## $p4LG4_83000_6_6
##
##
##            maxLoc   maxSel    maxG   maxTime   maxSource
## -------  --------  -------  ------  --------  ----------
## Model1    7856956      0.6      NA        NA          NA
```

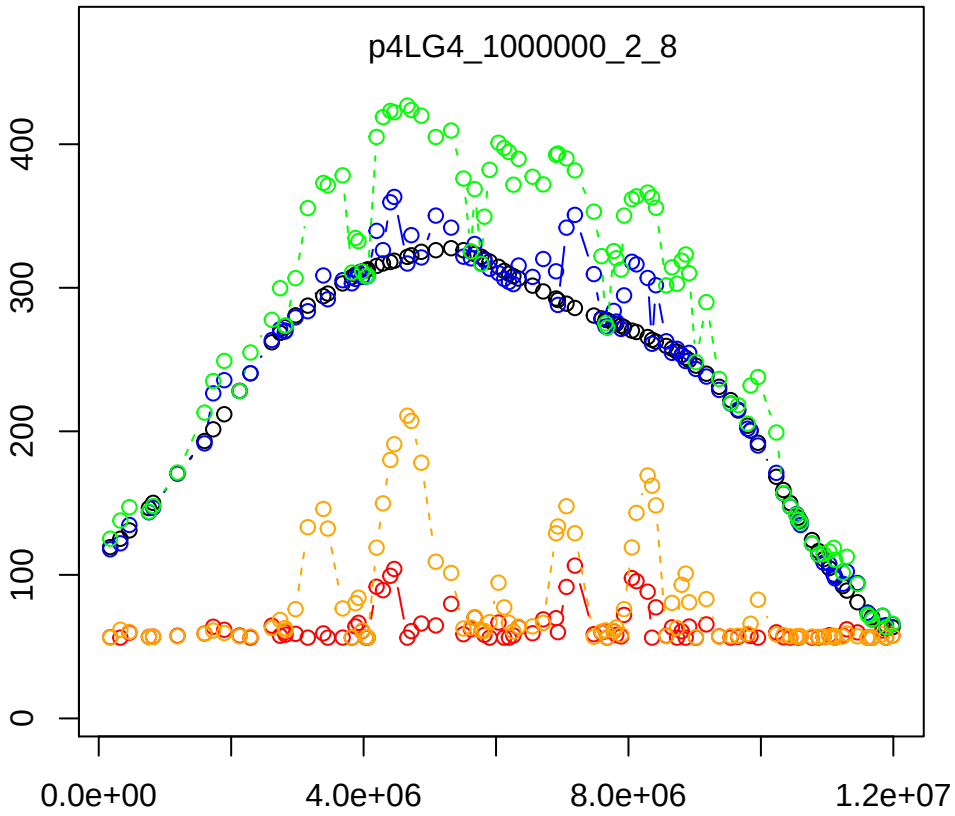
```
## Model2    8889185      0.6   1e+00         3          NA
## Model3    4291898      0.6   1e-01         0           5
## Model4    8889185      0.6   6e-06         0           3
## Model5    4291898      0.6   1e-01         0           5
##
## $p4LG4_830000_6_6
##
##
##            maxLoc   maxSel    maxG   maxTime   maxSource
## -------  --------  -------  ------  --------  ----------
## Model1    7856956      0.6      NA        NA          NA
## Model2    8889185      0.6   1e+00         3          NA
## Model3    4291898      0.6   1e-01         0           5
## Model4    8889185      0.6   6e-07         0           3
## Model5    4291898      0.6   1e-01         0           5
##
## $p4LG4_8300000_6_6
##
##
##            maxLoc   maxSel    maxG   maxTime   maxSource
## -------  --------  -------  ------  --------  ----------
## Model1    7856956      0.6      NA        NA          NA
## Model2    8889185      0.6   1e+00         3          NA
## Model3    4291898      0.6   1e-01         0           5
## Model4    8889185      0.6   1e-07         0           3
## Model5    4291898      0.6   1e-01         0           5
##
## $p4LG4_8300_5_6
##
##
##            maxLoc   maxSel       maxG   maxTime   maxSource
## -------  --------  -------  ---------  --------  ----------
## Model1    7856956      0.6         NA        NA          NA
## Model2    8889185      0.6   1.00e-05         5          NA
## Model3    8889185      0.6   1.00e-04         5           3
## Model4    8889185      0.6   6.02e-05         0           3
## Model5    4291898      0.6   1.00e-01         0           5
##
## $p4LG4_83000_5_6
##
##
##            maxLoc   maxSel    maxG   maxTime   maxSource
## -------  --------  -------  ------  --------  ----------
## Model1    7856956      0.6      NA        NA          NA
## Model2    8889185      0.6   1e+00         3          NA
## Model3    4291898      0.5   1e-01         0           5
## Model4    8889185      0.6   6e-06         0           3
## Model5    4291898      0.6   1e-01         0           5
##
## $p4LG4_830000_5_6
##
##
##            maxLoc   maxSel    maxG   maxTime   maxSource
## -------  --------  -------  ------  --------  ----------
```

```
## Model1    7856956      0.6      NA        NA          NA
## Model2    8889185      0.6   1e+00         3          NA
## Model3    4291898      0.5   1e-01         0           5
## Model4    8889185      0.6   6e-07         0           3
## Model5    4291898      0.6   1e-01         0           5
##
## $p4LG4_8300000_5_6
##
##
##             maxLoc   maxSel    maxG   maxTime   maxSource
## -------  --------  -------  ------  --------  ----------
## Model1    7856956      0.6      NA        NA          NA
## Model2    8889185      0.6   1e+00         3          NA
## Model3    4291898      0.5   1e-01         0           5
## Model4    8889185      0.6   1e-07         0           3
## Model5    4291898      0.6   1e-01         0           5
```

Model with chosen parameters

```
selSite = positions[seq(1, length(positions[positions<12000000]), length.out = 100)]
par(mfrow=c(2,1),oma=c(2,2,2,2),mar=c(2,2,2,2))
max.complikes1<-plot.complike("p4LG4_100000_2_8",leg=FALSE,selSite = selSite)
max.complikes2<-plot.complike("p4LG4_1000000_2_8",leg=FALSE,selSite = selSite)
```


p4LG4_100000_2_8
0
100
200
300
400
500
0.0e+00
4.0e+06
8.0e+06
1.2e+07


p4LG4_1000000_2_8
0
100
200
300
400
0.0e+00
4.0e+06
8.0e+06
1.2e+07

```
mcle1<-get.mcle(max.complikes1[[1]],Ne=100000,rec=2*10^-8,selSite = selSite)
kable(mcle1)
```

	maxLoc	maxSel	maxG	maxTime	maxSource
Model1	5322346	0.60	NA	NA	NA
Model2	7191007	0.25	1e-05	3	NA
Model3	4462715	0.60	5e-06	50	5
Model4	4661258	0.60	5e-06	0	7
Model5	4661258	0.60	5e-06	25	5

```
mcle2<-get.mcle(max.complikes2[[1]],Ne=1000000,rec=2*10^-8,selSite = selSite)
kable(mcle2)
```

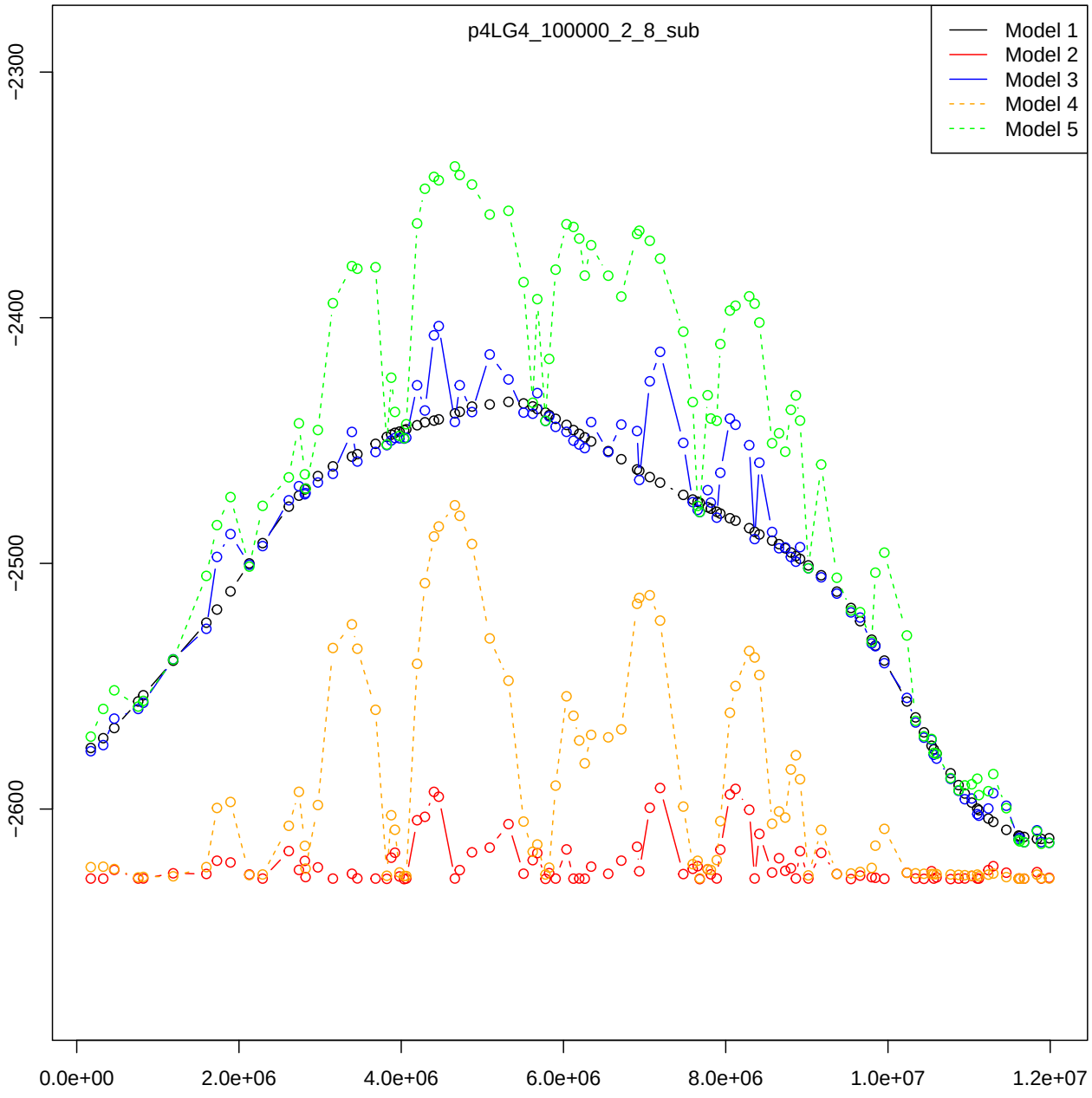
	maxLoc	maxSel	maxG	maxTime	maxSource
Model1	5322346	0.60	NA	NA	NA
Model2	7191007	0.08	1e+00	5	NA
Model3	4462715	0.60	5e-07	50	3
Model4	4661258	0.25	5e-07	0	7
Model5	4661258	0.60	5e-07	25	3

The selection values are still the smallest ones. Kristin Lee indicated that this might occur if there aren't enough neutral SNPs - so I'll try re-calculating neutral SNPS

```
#calculate selected allele frequencies
LG4.sub<-calc.allFreqs(vcf[vcf$`#CHROM` == "LG4" & vcf$POS < 12*10^6,],pop.list,pop.labs)
saveRDS(LG4.sub,"dmc/selectedRegionAlleleFreqs_p4LG4_subset.RDS")
saveRDS(vcf[vcf$`#CHROM`=="LG4"& vcf$POS < 12*10^6,"POS"],"dmc/selectedRegionPositions_p4LG4_subset.RDS

#calculate neutral allele frequencies
all.out<-read.delim("all_outliers.txt",sep='\t',header=TRUE)
allFreqs<-calc.allFreqs(vcf[!vcf$`#CHROM` %in% "LG4" & !vcf$SNP %in% all.out$SNP,],
                  pop.list,pop.labs)
saveRDS(allFreqs,"dmc/neutralAlleleFreqs_p4LG4_notSNPs.RDS")

neutralF_filename<-"dmc/neutralF_p4LG4_subset"
source("../programs/dmc-master/calcNeutralF.R")
F_estimate<-readRDS("dmc/neutralF_p4LG4_subset.RDS")

#calculate the determinant and inverse of the neutral F matrix
M <- numPops
Tmatrix <- matrix(data = rep(-1 / M, (M - 1) * M), nrow = M - 1, ncol = M)
diag(Tmatrix) = (M - 1) / M
sampleErrorMatrix = diag(1/sampleSizes, nrow = numPops, ncol = numPops)
det_FOmegas_neutral = det(Tmatrix %*% (F_estimate + sampleErrorMatrix) %*% t(Tmatrix))
saveRDS(det_FOmegas_neutral, "dmc/det_FOmegas_neutral_p4LG4_subset.RDS")

inv_FOmegas_neutral = ginv(Tmatrix %*% (F_estimate + sampleErrorMatrix) %*% t(Tmatrix))
saveRDS(inv_FOmegas_neutral, "dmc/inv_FOmegas_neutral_p4LG4_subset.RDS")
```

```
positions<-readRDS("dmc/selectedRegionPositions_p4LG4_subset.RDS")
F_estimate<-readRDS("dmc/neutralF_p4LG4_subset.RDS")
```

```
sampleSizes<-readRDS("dmc/sampleSizes.RDS")
neutral_det_name = "dmc/det_FOmegas_neutral_p4LG4_subset.RDS"
neutral_inv_name = "dmc/inv_FOmegas_neutral_p4LG4_subset.RDS"
selSite = positions[seq(1, length(positions[positions<12000000]), length.out = 100)]
par(mfrow=c(1,1),oma=c(2,2,2,2),mar=c(2,2,2,2))
max.complikes1<-plot.complike("p4LG4_100000_2_8_sub",leg=TRUE,selSite = selSite)
```



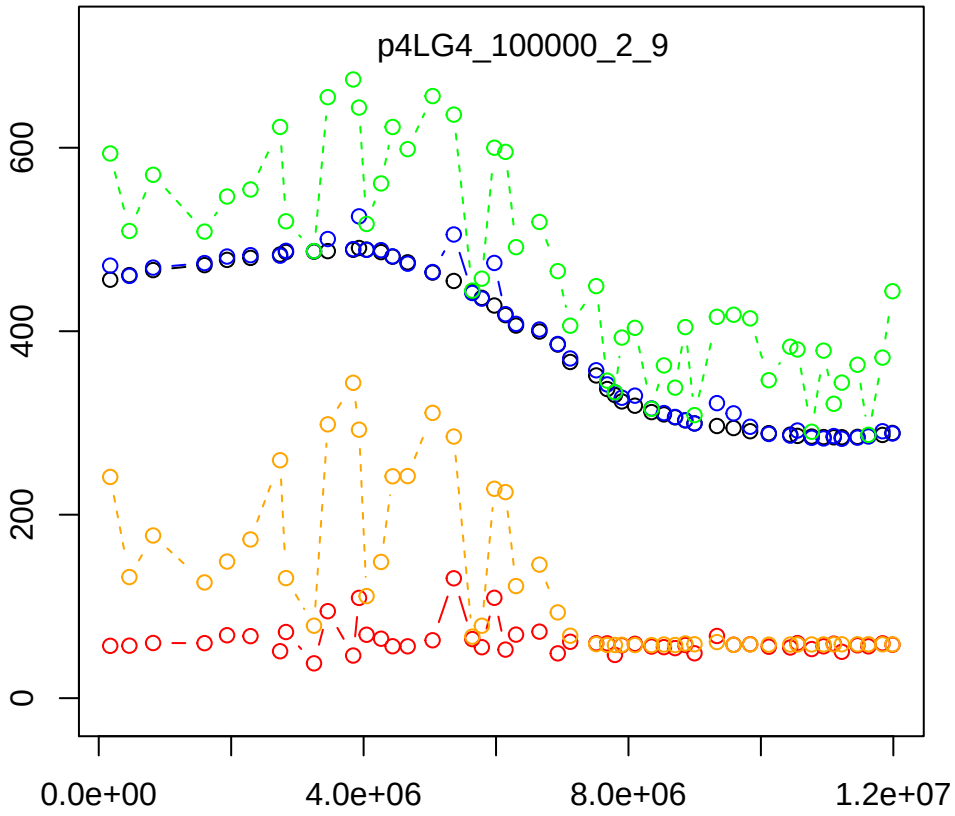

```
mcle.sub<-get.mcle(max.complikes1[[1]],Ne=100000,rec=2*10^-8,
                   selSite=selSite)
kable(mcle.sub)
```

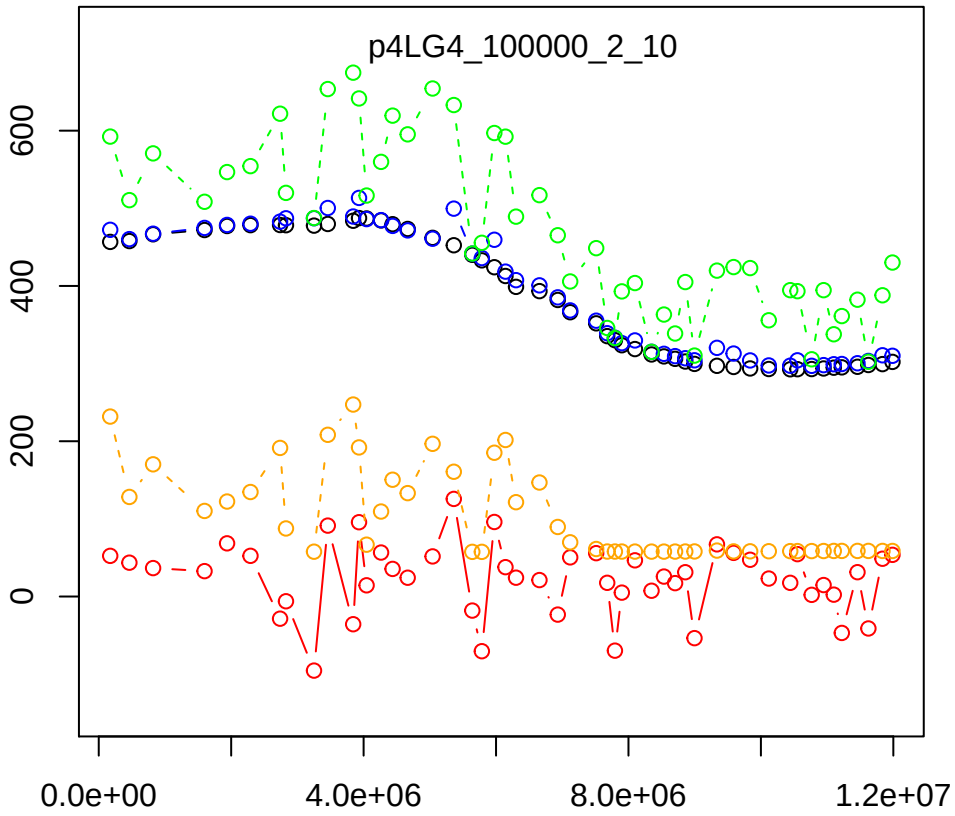
	maxLoc	maxSel	maxG	maxTime	maxSource
Model1	5322346	0.60	NA	NA	NA
Model2	7191007	0.09	1e-05	5	NA

	maxLoc	maxSel	maxG	maxTime	maxSource
Model3	4462715	0.60	5e-06	50	5
Model4	4661258	0.50	5e-06	0	7
Model5	4661258	0.60	5e-06	25	5

Let's try lower recombination rates

```
selSite = positions[seq(1, length(positions), length.out = 50)]
F_estimate<-readRDS("dmc/neutralF_p4LG4_50.RDS")
sampleSizes<-readRDS("dmc/sampleSizes.RDS")
par(mfrow=c(2,1),oma=c(2,2,2,2),mar=c(2,2,2,2))
max.complikes1<-plot.complike("p4LG4_100000_2_9",leg=FALSE,selSite = selSite)
max.complikes2<-plot.complike("p4LG4_100000_2_10",leg=FALSE,selSite = selSite)
```


p4LG4_100000_2_9
0
200
400
600
0.0e+00
4.0e+06
8.0e+06
1.2e+07


p4LG4_100000_2_10
0
200
400
600
0.0e+00
4.0e+06
8.0e+06
1.2e+07

```
mcle1<-get.mcle(max.complikes1[[1]],Ne=100000,rec=2*10^-9)
kable(mcle1)
```

	maxLoc	maxSel	maxG	maxTime	maxSource
Model1	3930682	0.15	NA	NA	NA
Model2	5362182	0.02	1e-05	3	NA
Model3	3930682	0.15	1e-04	500	5
Model4	3842610	0.06	5e-06	0	7
Model5	3842610	0.20	5e-06	50	3

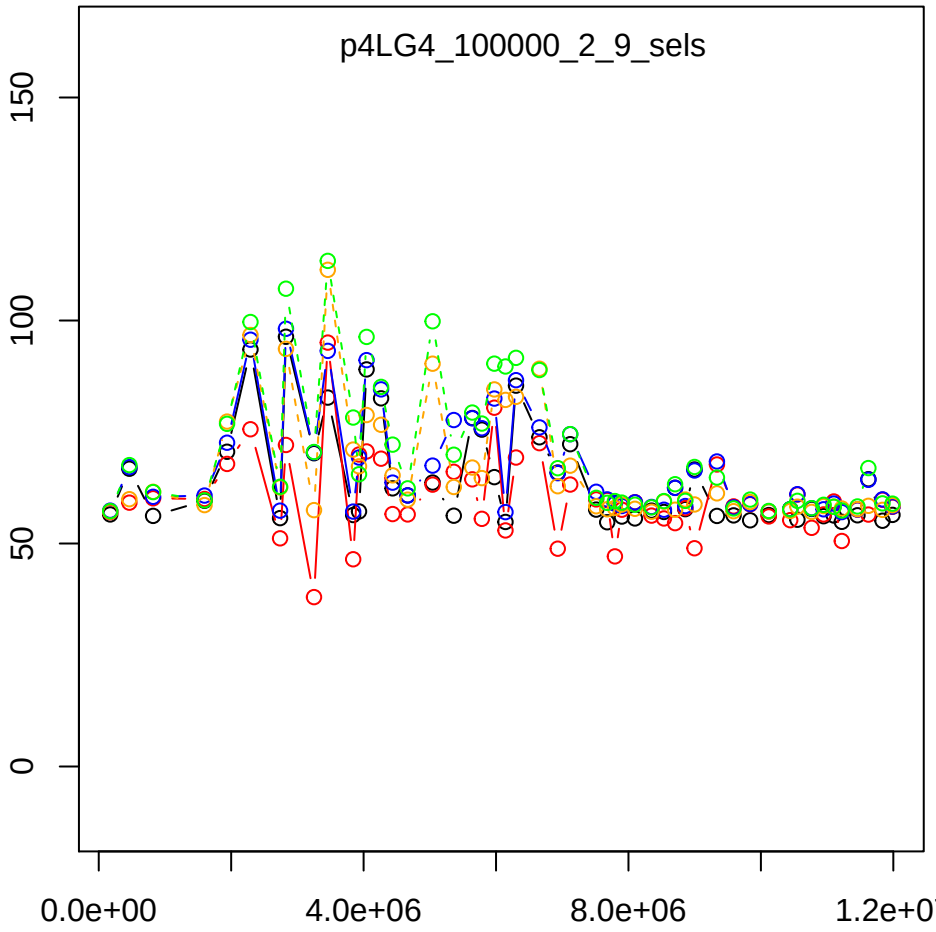
```
mcle2<-get.mcle(max.complikes2[[1]],Ne=100000,rec=2*10^-10)
kable(mcle2)
```

	maxLoc	maxSel	maxG	maxTime	maxSource
Model1	3930682	0.010	NA	NA	NA
Model2	5362182	0.001	1e-05	3	NA
Model3	3930682	0.020	5e-06	10000	5
Model4	3842610	0.010	5e-06	0	7
Model5	3842610	0.020	5e-06	500	3

Different selection parameters

Going below 1e-5 (i.e. to 1e-6) caused infinite values to occur

```
sels = c(9e-4,7.5e-4,5e-4, 2.5e-4,1e-4)
selSite = positions[seq(1, length(positions[positions<12000000]), length.out = 50)]
F_estimate<-readRDS("dmc/neutralF_p4LG4_50.RDS")
par(mfrow=c(1,1),oma=c(2,2,2,2),mar=c(2,2,2,2))
max.complikes.sel<-plot.complike("p4LG4_100000_2_9_sels",leg=FALSE,selSite = selSite)
```



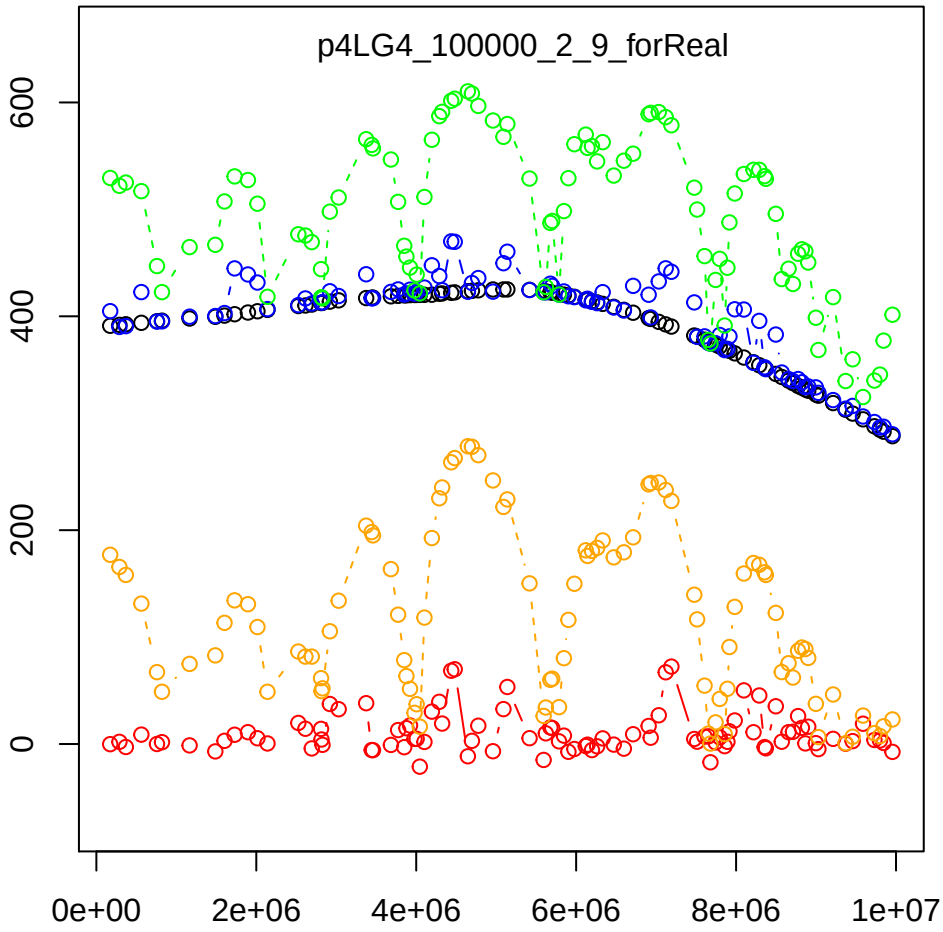

```
mcle.sel<-get.mcle(max.complikes.sel[[1]],Ne=100000,rec=2*10^-9,sels=sels)
kable(mcle.sel)
```

	maxLoc	maxSel	maxG	maxTime	maxSource
Model1	2827629	0.00090	NA	NA	NA
Model2	3459260	0.00075	1e-05	7	NA
Model3	2827629	0.00090	1e-02	10000	5
Model4	3459260	0.00090	5e-06	0	7
Model5	3459260	0.00090	1e-01	100	5

Chosen parameter runs

Using recombination rate of 2e-9, Ne= 100000, with a wider range of selection values that go below 1e-4, and 100 selection sites from 0 to 10Mb.

```
selSite = positions[seq(1, length(positions[positions<10000000]), length.out = 100)]
F_estimate<-readRDS("dmc/neutralF_p4LG4_100_10MB.RDS")
    sels = c(1e-4, 1e-3, 0.01, seq(0.02, 0.14, by = 0.01), seq(0.15, 0.3, by = 0.05),
             seq(0.4, 0.6, by = 0.1))
    par(mfrow=c(1,1),oma=c(2,2,2,2),mar=c(2,2,2,2))
max.complikes<-plot.complike("p4LG4_100000_2_9_forReal",leg=FALSE,selSite = selSite)
```




```
mcle<-get.mcle(max.complikes[[1]],Ne=100000,rec=2*10^-9,sels=sels,selSite=selSite)
kable(mcle)
```

	maxLoc	maxSel	maxG	maxTime	maxSource
Model1	5139485	0.15	NA	NA	NA
Model2	7191007	0.02	1e-05	3	NA
Model3	4437840	0.20	5e-06	500	5
Model4	4645842	0.06	5e-06	0	7
Model5	4645842	0.20	5e-06	50	3

Plot profile likelihood surfaces

Plot the profile likelihood surfaces for Models 3 and 5, where standing variation is present in all or one population, which have the highest composite log-likelihoods.

```
selSite = positions[seq(1, length(positions[positions<10000000]), length.out = 100)]
gs<-c(1/(2*Ne), 10^-(4:1))
Ne=100000
rec=2*10^-9
sources <- c(3,5,7,16)
times = c(0, 5, 25, 50, 100, 500, 1000, 1e4, 1e6)
migs = c(10^-(seq(5, 1, by = -2)), 0.5, 1)
    sels = c(1e-4, 1e-3, 0.01, seq(0.02, 0.14, by = 0.01), seq(0.15, 0.3, by = 0.05),
```

```
            seq(0.4, 0.6, by = 0.1))
par(mfrow=c(2,1),oma=c(2,2,2,2),mar=c(2,2,2,2))
## Model 3
compLikelihood_sv<-readRDS("dmc/compLikelihood_sv_p4LG4_100000_2_9_forReal.RDS")
names(compLikelihood_sv)<-selSite
compLikelihood_sv_site = sapply(1 : length(selSite), function(i) {
  max(unlist(compLikelihood_sv[[i]]))
})
mcle_sv = getMCLEsv_source(compLikelihood_sv,selSite, sels, gs, times, sources)
mcle_sv_index<-which(mcle_sv[1]==selSite)
compLike_sv_byTime = lapply(1 : length(times), function(time) {
  sapply(1: length(sels), function(sel) {
    sapply(1 : length(gs), function(g) {
      compLikelihood_sv[[mcle_sv_index]][[sel]][[g]][[time]]
    })
  })
})

profileLike_time_sv = sapply(1: length(compLike_sv_byTime), function(i) {
  max(unlist(compLike_sv_byTime[[i]]))
})

plot(times, profileLike_time_sv, type = "b", xlab = "Time",
     ylab = "Profile composite log-likelihood", main = "Figure 2: Model 3")
abline(v = mcle_sv[4], lty = 2, col = "red")

## Model 5
compLikelihood_mixed_svInd<-readRDS("dmc/compLikelihood_mixed_svInd_p4LG4_100000_2_9_forReal.RDS")
mcle_mixed_svInd = getMCLEmixed(compLikelihood_mixed_svInd, selSite, sels,
                                gs, times, migs[1], sources)
mcle_mixed_svInd_index<-which(mcle_sv[1]==selSite)
compLike_mixed_svInd_byTime = lapply(1 : length(times), function(time) {
  sapply(1: length(sels), function(sel) {
    sapply(1 : length(gs), function(g) {
      compLikelihood_mixed_svInd[[mcle_mixed_svInd_index]][[sel]][[g]][[time]]
    })
  })
})

profileLike_time_mixed_svInd = sapply(1: length(compLike_mixed_svInd_byTime), function(i) {
  max(unlist(compLike_mixed_svInd_byTime[[i]]))
})

plot(times, profileLike_time_mixed_svInd, type = "b", xlab = "Time",
     ylab = "Profile composite log-likelihood", main = "Figure 3: Model 5")
abline(v = mcle_mixed_svInd[4], lty = 2, col = "red")
```

Figure 2: Model 3

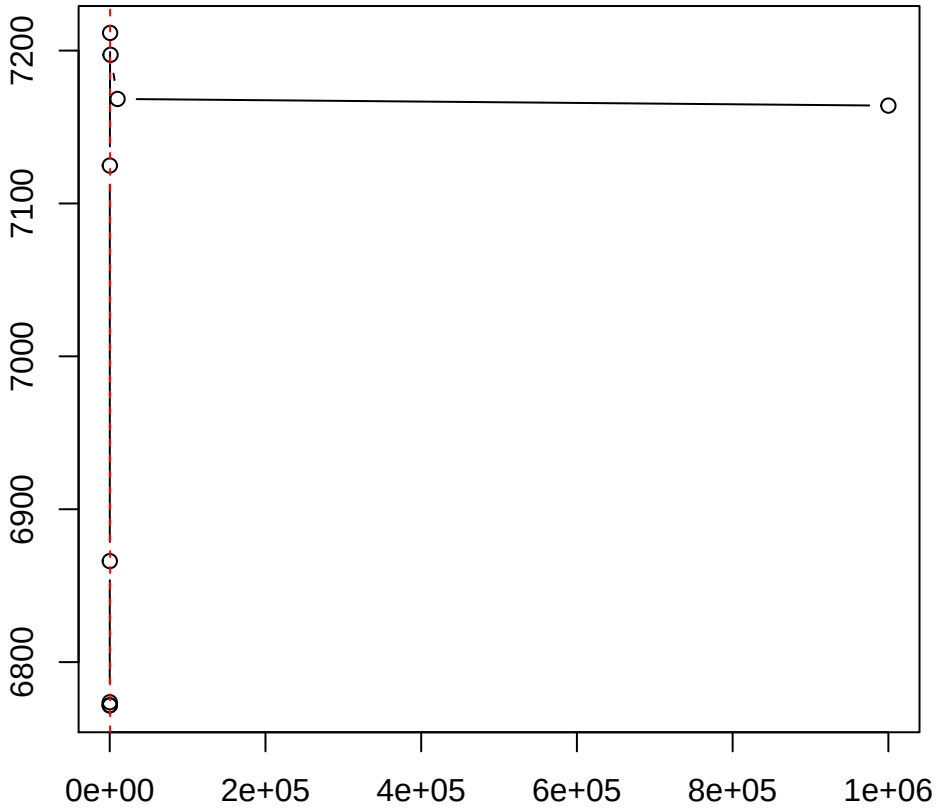
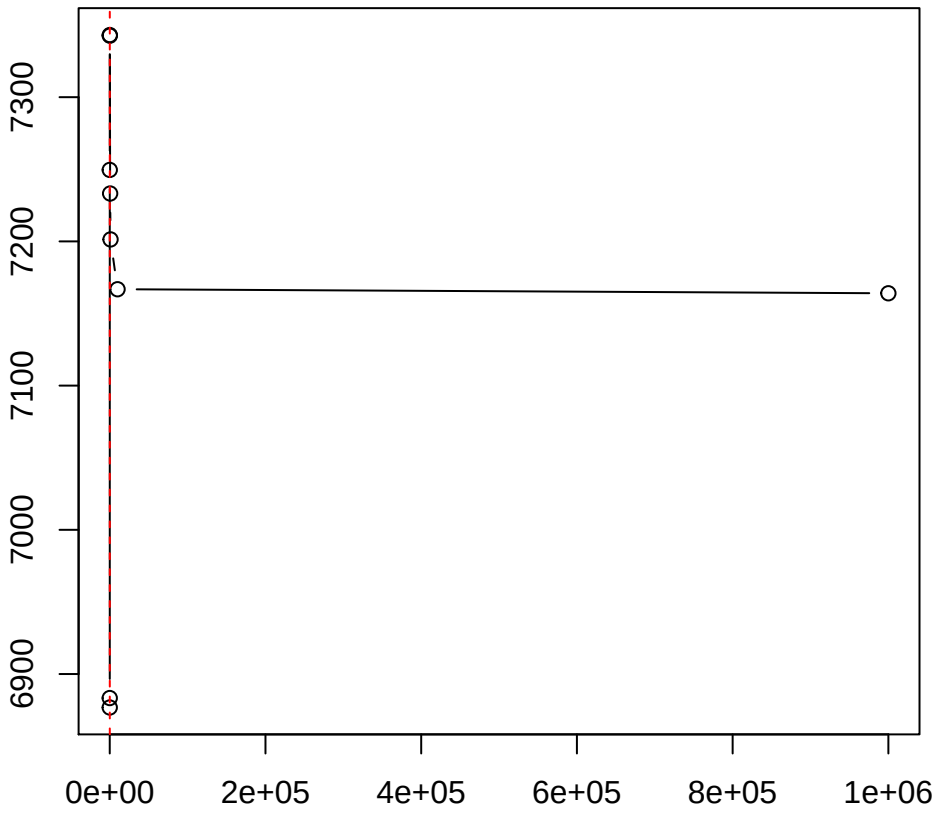


Figure 3: Model 5

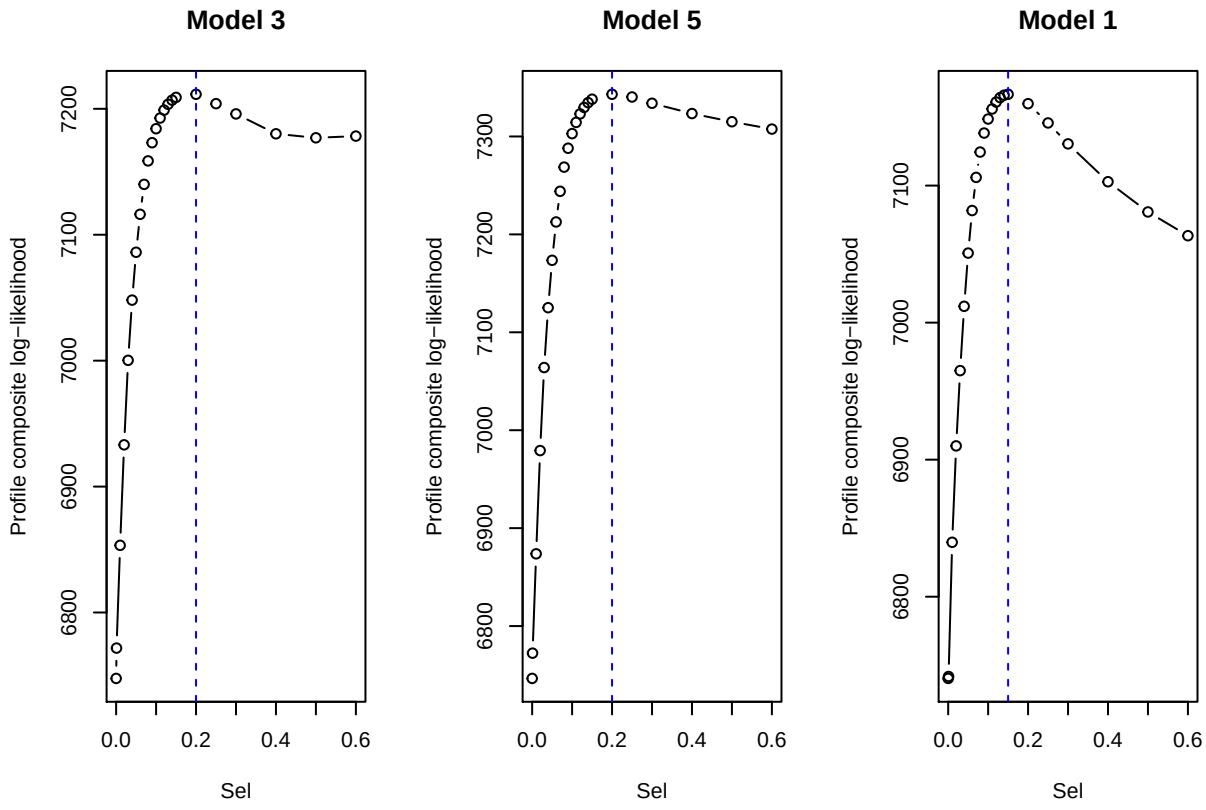


We see little difference in the composite log-likelihoods of models 3 and 5, provides further evidence that independent mutations (or selection very old standing variation) in FLFW generated the patterns observed in the data. The point estimates of MCLE of t are marked by red lines in Figures 2 and 3.

MCLE for the strength of selection

The best fitting models are 5 (green), 3 (blue), and 1 (black). Models 3 and 5 estimate selection at 0.2 and model 1 estimates it at 0.15.

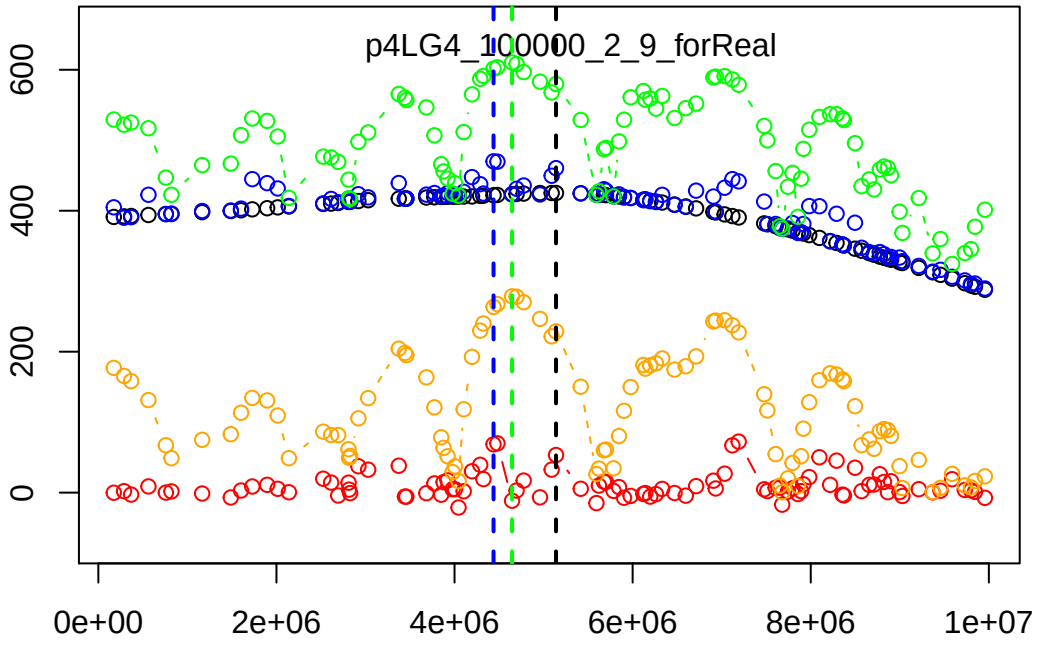
```
## Model 1
compLikelihood_ind<-readRDS("dmc/compLikelihood_ind_p4LG4_100000_2_9_forReal.RDS")
mcle_ind = getMCLEind(compLikelihood_ind, selSite, sels)
mcle_ind_index<-which(mcle_ind[1]==selSite)
profileLike_sel_ind = sapply(1: length(sels), function(i) {
  max(unlist(compLikelihood_ind[[mcle_ind_index]][[i]]))
})

## Model 3
profileLike_sel_sv = sapply(1: length(sels), function(i) {
  max(unlist(compLikelihood_sv[[mcle_sv_index]][[i]]))
})

## Model 5
profileLike_sel_mixed_svInd = sapply(1: length(sels), function(i) {
  max(unlist(compLikelihood_mixed_svInd[[mcle_mixed_svInd_index]][[i]]))
})

par(mfrow = c(1, 3))
plot(sels, profileLike_sel_sv, type = "b", xlab = "Sel",
     ylab = "Profile composite log-likelihood", main = "Model 3")
abline(v = mcle_sv[2], lty = 2, col = "blue")

plot(sels, profileLike_sel_mixed_svInd, type = "b", xlab = "Sel",
     ylab = "Profile composite log-likelihood", main = "Model 5")
abline(v = mcle_mixed_svInd[2], lty = 2, col = "blue")

plot(sels, profileLike_sel_ind, type = "b", xlab = "Sel",
     ylab = "Profile composite log-likelihood", main = "Model 1")
abline(v = mcle_ind[2], lty = 2, col = "blue")
```



Where is the selected site?

The selected site is somewhere between 4.43784×10^6 and 5.139485×10^6, if we take the maximum estimates from models 1, 3, and 5.

```
par(mfrow=c(1,1),oma=c(2,2,2,2),mar=c(2,2,2,2))
max.complikes<-plot.complike("p4LG4_100000_2_9_forReal",leg=FALSE,selSite = selSite)
abline(v=mcle_ind[1],lty=2,lwd=2,col="black")
abline(v=mcle_sv[1],lty=2,lwd=2,col="blue")
abline(v=mcle_mixed_svInd[1],lty=2,lwd=2,col="green")
```


p4LG4_100000_2_9_forReal
600
400
200
0
0e+00
2e+06
4e+06
6e+06
8e+06
1e+07