Simulation study to test vivax relatedness model

```
## Loading required package: dplyr
## Warning: package 'dplyr' was built under R version 3.4.4
##
## Attaching package: 'dplyr'
## The following objects are masked from 'package:stats':
##
##     filter, lag
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
## Loading required package: Matrix
## Warning: package 'Matrix' was built under R version 3.4.4
## Loading required package: gtools
## Warning: package 'gtools' was built under R version 3.4.4
## Loading required package: tictoc
## Loading required package: doParallel
## Warning: package 'doParallel' was built under R version 3.4.4
## Loading required package: foreach
## Loading required package: iterators
## Warning: package 'iterators' was built under R version 3.4.4
## Loading required package: parallel
## Loading required package: igraph
## Warning: package 'igraph' was built under R version 3.4.4
##
## Attaching package: 'igraph'
## The following object is masked from 'package:gtools':
##
##     permute
## The following objects are masked from 'package:dplyr':
##
##     as_data_frame, groups, union
## The following objects are masked from 'package:stats':
##
##     decompose, spectrum
## The following object is masked from 'package:base':
##
##     union
## Loading required package: matrixStats
## Warning: package 'matrixStats' was built under R version 3.4.4
```

```
##
## Attaching package: 'matrixStats'

## The following object is masked from 'package:dplyr':
##
##     count

## Loading required package: RColorBrewer
```

Determines whether to run the full suite of simulations (takes a long time to run).

```
RUN_MODELS = F
PLOT_RESULTS = T
```

Simulation 1: Effective Complexity of Infection

We want to know how adding extra 'noise' parasites into the data will affect the estimator of the probability of relatedness between parasites. We also explore how this is a function of the number of alleles measured.

Outline of simulation is as follows: * Simulate data for N individuals, with M markers which are polyallelic for a given number (this controls complexity of problem). We do this for two episodes where there are underlying clonal or sibling relationships between episodes. * Compute resulting theta estimate * Plot theta as a function of the problem complexity, and the effective MOI

```
# Setup simulation study parameters
N_alleles = 10 # constant heterozygosity for all markers
K_indivs = 1000
Ms = seq(3,15, by = 3) # number of markers

K_poly_markers = 3

MOI_1_max = 2
MOI_2_max = 2

settings = expand.grid(1:MOI_1_max, 1:MOI_2_max, Ms)
names(settings)= c('MOI_1', 'MOI_2', 'M')
settings = settings[settings$MOI_1+settings$MOI_2<4,]
```

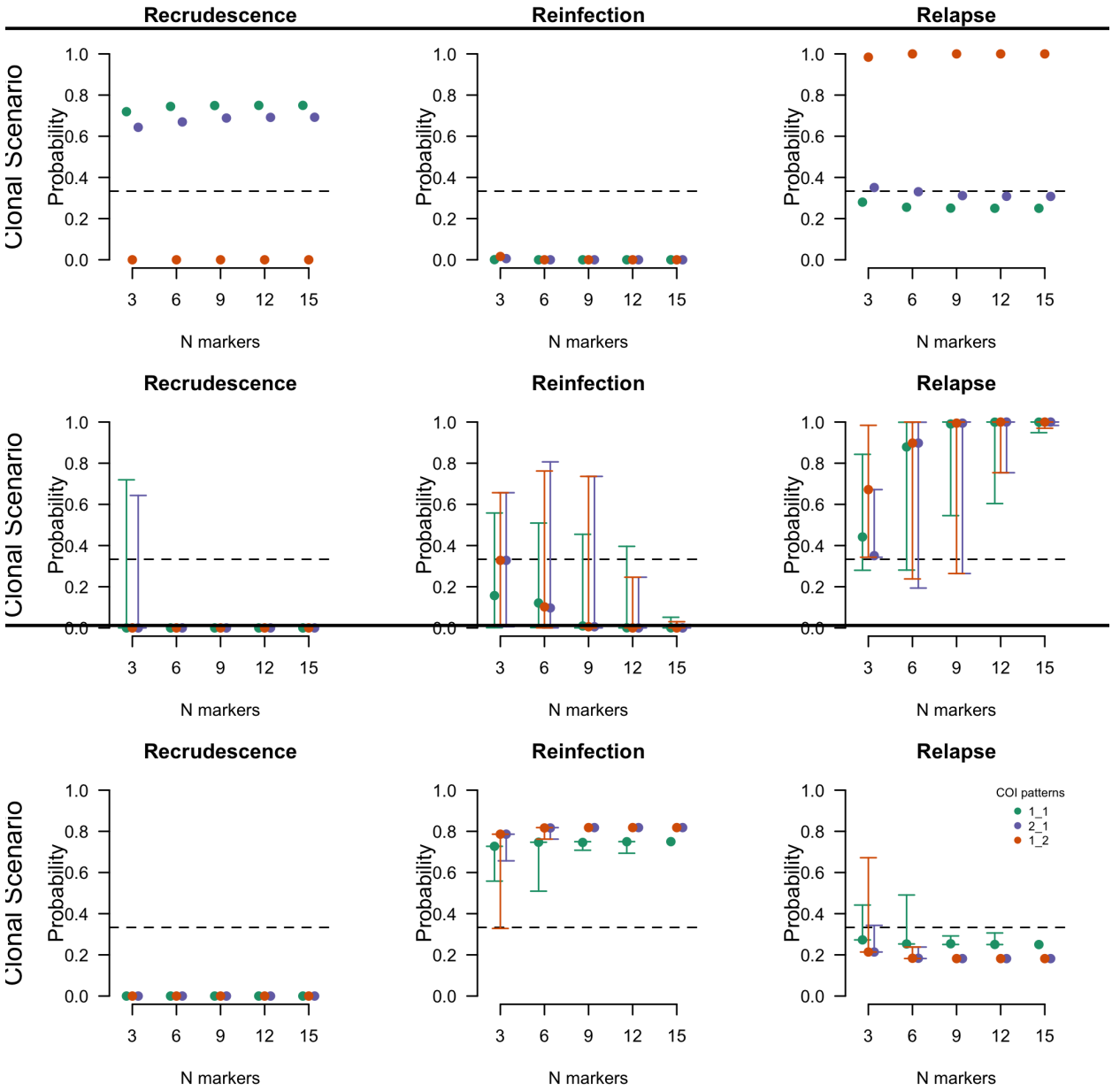
```
# All simulation parameter settings possible
if(RUN_MODELS){
  # iterate over types of data
  for(related_type in c('Sibling','Stranger','Clone')){
    # iterate over parameter settings
    JOBS = nrow(settings)
    thetas_all = foreach(s = 1:JOBS, .combine = rbind,
                         .packages = c('dplyr','Matrix','gtools',
                                       'igraph','matrixStats','doParallel')
    ) %do% { # parallisation happening inside the function

      MOI_1 = settings$MOI_1[s]
      MOI_2 = settings$MOI_2[s]
      M = settings$M[s]

      MS_markers = sapply(1:M, function(x) paste0('MS',x))
      FS = lapply(MS_markers, function(x) table(1:N_alleles)/N_alleles)
      # Uniform distribution on the alleles
```

```
      names(FS) = MS_markers

      ###################################
      # create a dataframe to store the simulated MS data
      ###################################

      MS_data = BuildSimData(Tn = 2, MOIs = c(MOI_1, MOI_2),
                             M = M, N = K_indivs,
                             N_alleles = N_alleles,
                             K_poly_markers = K_poly_markers,
                             relatedness = related_type)
      eps_ids = unique(MS_data$Episode_Identifier[MS_data$Episode>1])
      P_matrix = data.frame(Episode_Identifier=eps_ids,
                            C = rep(1/3,K_indivs),
                            L = rep(1/3,K_indivs),
                            I = rep(1/3,K_indivs))
      ###################################
      # Run the model on the data
      ###################################

      TH = post_prob_CLI(MS_data = MS_data, cores = 7,
                         Fs = FS, verbose = F, p = P_matrix)
      # Add setting number as key for plotting
      TH$setting = s
      TH # return results
    }
    writeLines(paste0('********** Done for ',related_type,' **********'))
    fname = paste0('SimulationOutputs/Posterior_Probs_',related_type,'_EffectMOI.RData')
    save(thetas_all, file = fname)
  }
}
```

Plot results

```
State_names = list(L='Relapse',I='Reinfection',C='Recrudescence')
if(PLOT_RESULTS){
  epsilon=0.4
  mycols = brewer.pal(MOI_1_max+MOI_2_max, name = 'Dark2')
  par(mfcol=c(3,3), las=1, bty='n', mar=c(4,5,3,2))
  # Iterate over the recurrence states
  for(State in c('C','I','L')){

    # iterate over types of data
    for(related_type in c('Clone','Sibling','Stranger')){

      fname = paste0('SimulationOutputs/Posterior_Probs_',related_type,'_EffectMOI.RData')
      load(fname)
      plot(NA,NA,
           xlim = c(2,16),ylim=c(0,1),
           xlab='N markers',
           ylab = '', xaxt='n',
           main = '')#paste(related_type, ':', State))
```

```
      mtext(text = 'Probability',side = 2,line = 2,las=3, cex=.8)
      if(State=='C') mtext(text = 'Clonal Scenario',side = 2,line = 4,las=3, cex=1)
      title(State_names[[State]])
      axis(1, at = Ms)
      abline(h=1/3, lty=2)
      settings$MOI_total = apply(settings[,1:2],1,sum)
      settings$MOI_pattern = as.factor(apply(settings[,1:2],1,paste, collapse = '_'))
      settings$Plot_offset = as.numeric(settings$MOI_pattern)
      Max_adj = MOI_1_max+MOI_2_max

      for(m in unique(settings$M)){
        for(MOI_pattern in unique(settings$MOI_pattern)){
          sub_settings = which(settings$M==m & settings$MOI_pattern==MOI_pattern)
          adj_x = settings$Plot_offset[sub_settings]
          sub_results = filter(thetas_all, setting==sub_settings)
          points(m + epsilon*adj_x - epsilon*Max_adj/2,
                 quantile(sub_results[, State], probs = 0.5),
                 pch = 19, col = mycols[adj_x])
          x1 = quantile(sub_results[,State], probs = 0.1)
          x2 = quantile(sub_results[,State],probs = .9)
          if( (x1 + 0.01 ) < x2){
            arrows(m + epsilon*adj_x - epsilon*Max_adj/2,
                   x1,m + epsilon*adj_x - epsilon*Max_adj/2,
                   x2,length=0.05, angle=90, code=3,
                   col = mycols[adj_x])
          }
        }
      }



    }
  }
  legend('topright',pch=19, col=mycols[unique(settings$Plot_offset)],
         legend = unique(settings$MOI_pattern),cex=.7,title='COI patterns',
         bty='n')

   x.tmp <- grconvertX(25, to='ndc')
        y.tmp <- grconvertY(1.8, to='ndc')
        par(xpd=NA)
        segments( -1000, grconvertY(y.tmp, from='ndc'),
                  grconvertX(x.tmp,  from='ndc'), grconvertY(y.tmp, from='ndc'),
                  col='black',
                  lwd = 2)

         x.tmp <- grconvertX(25, to='ndc')
        y.tmp <- grconvertY(4.7, to='ndc')
        par(xpd=NA)
        segments( -1000, grconvertY(y.tmp, from='ndc'),
                  grconvertX(x.tmp,  from='ndc'), grconvertY(y.tmp, from='ndc'),
                  col='black',
                  lwd = 2)
}
```


Recrudescence
Reinfection
Relapse
Clonal Scenario
Probability
N markers
COI patterns
1_1
2_1
1_2