MCsim

Sarah I. Murphy, Samantha Lau, Timothy Lott, Aljosa Trmcic

1/28/2020

```
set.seed(42) #use a seed value for reproducibility
```

Utility Functions

- muAtNewTemp
  - Purpose: Calculate the new mu parameter at new temperature.
  - Params:
    - newTemp: the new temperature for which we calculate mu
    - oldMu: the previous mu value to adjust
    - oldTemp: the temperature corresponding to previous mu
    - T0: parameter used to calculate new mu

```
muAtNewTemp <- function(newTemp, oldMu, oldTemp = 6, T0 = -3.62) {
  numerator <- newTemp - T0
  denom <- oldTemp - T0
  newMu <- ((numerator / denom)^2) * oldMu

  return(newMu)
}
##adjustLag
#Purpose: Adjust the lag phase based on the Zwietering 1994 paper.
#Params:  t: the current timestep
#         oldLag: the lag time at the previous temperature
#         newLag: the lag time at the current temparature.
#         restartExp: If true then lag phase restarts even if already in
#                     exponential growth phase.
#         adjustmentConstant: The amount to adjust lag, the paper recommends 0.25
#Notes:
adjustLag <- function (t, oldLag, newLag, restartExp = T, adjustmentConstant = 0.25) {
  #determine the amount of lag phase completed
  remainingLag <- 1 - (t / oldLag)
  if(restartExp) {
  remainingLag <- ifelse(remainingLag < 0, 0, remainingLag)
  }
  else {
  adjustedLag <- ifelse(remainingLag <=0, oldLag,
                        t + remainingLag * newLag + adjustmentConstant*newLag)
  }

  adjustedLag <- t + remainingLag*newLag + adjustmentConstant*newLag
  return(adjustedLag)
}
##lagAtNewTemp
#Purpose: Calculate the new lag parameter at new temperature.
#Params:  newTemp: the new temperature for which we calculate lag
```

```
#         oldLag: the previous lag value to adjust
#         oldTemp: the temperature corresponding to previous lag
#         T0:    Parameter used to calculate new lag
lagAtNewTemp <- function (t, newTemp, oldLag, oldTemp = 6, T0 = -3.62) {
  numerator <- oldTemp -T0
  denom <- newTemp - T0
  newLag <- ( (numerator / denom)^2) * oldLag
  return(newLag)
}

getPrevRow <- function(df, sim_run, half_gallon, day) {
  old_temp <- df[df$BT == sim_run & df$half_gal == half_gallon & df$day==day-1,]
}

#Growth Models
buchanan_log10N = function(t,lag,mumax,LOG10N0,LOG10Nmax){
  ans <- LOG10N0 + (t >= lag) * (t <= (lag + (LOG10Nmax - LOG10N0) *     log(10)/mumax)) * mumax * (t -
  return(ans)
}
gompertz_log10N = function(t,lag,mumax,LOG10N0,LOG10Nmax) {
  ans <- LOG10N0 + (LOG10Nmax - LOG10N0) * exp(-exp(mumax * exp(1) *
                                                      (lag - t)/((LOG10Nmax - LOG10N0) * log(10)) + 1))
  return(ans)
}
baranyi_log10N = function(t,lag,mumax,LOG10N0,LOG10Nmax) {
  ans <- LOG10Nmax + log10((-1 + exp(mumax * lag) + exp(mumax *
                                                          t))/(exp(mumax * t) - 1 + exp(mumax * lag) * 

  return(ans)
}

#Function to calculate log10N
#Wwrapper function because it calls the proper model
#Purpose: This implements the growth model
log10N_func <- function(t, lag, mumax, LOG10N0, LOG10Nmax, model_name="buchanan") {
  if (model_name == "buchanan") {
    return(buchanan_log10N(t, lag, mumax, LOG10N0, LOG10Nmax) )
  }
  else if(model_name == 'baranyi') {
    return(baranyi_log10N(t, lag, mumax, LOG10N0, LOG10Nmax) )
  }
  else if(model_name == 'gompertz') {
    return(gompertz_log10N(t, lag, mumax, LOG10N0, LOG10Nmax) )
  }
  else {
    stop(paste0(model_name, " is not a valid model name. Must be one of buchanan, baranyi, gompertz"))
  }
}


# Data frame creation and setup   ----
#Set up data frame to store count at each day
#Size is for n_sim bulk tanks, n_half_gal half gallon lots, n_day days
```

```r
n_sim <-100    #1000 is for testing and exploring, experiments require at least 10k
n_halfgal <-10
n_day <- 24
start_day <- 1

#Repeat each element of the sequence 1..n_sim.Bulk tank data (MC runs)
BT <- rep(seq(1, n_sim), each = n_halfgal * n_day)
#Repeat the whole sequences times # of times
half_gal <- rep(seq(1, n_halfgal), times = n_day * n_sim)
#Vector of FALSE
AT <- vector(mode="logical", n_sim * n_halfgal * n_day)
#Repeat the days for each simulation run
day <- rep(rep(seq(start_day, start_day+n_day-1), each = n_halfgal), times = n_sim)
count <- vector(mode = "logical", n_sim * n_halfgal * n_day)

#matrix with columns:
#  BT   half_gal    AT    day   count
data <- data.frame(BT, half_gal, AT, day, count)

#Now import the data from our input files and begin filling in our data frames
#input files
frequency_file <- "Frequency.csv"
growth_file <- "GrowthParameters.csv"
init_file <- "InitialCountsMPN.csv"

#Import frequency data and get the rpoB allelic type
freq_import <- read.csv(frequency_file, stringsAsFactors = FALSE, header = TRUE)
freq_data = freq_import$rpoB.allelic.type

#Import growth parameter data
growth_import <-read.csv(growth_file, stringsAsFactors = FALSE)

#Import initial count logMPN data
initialcount_import <- read.csv(init_file, stringsAsFactors = FALSE)
#MPN Column
initialcount_data = initialcount_import[,3]
#LOG MPN Column
initialcountlog_data = initialcount_import[,4]

# Calculate samples used in the monte carlo   ----

#Now sample the MPN distributions and the temperature distribution
#Sample logMPN from normal distribution
logMPN_mean <- c(-0.7226627)
logMPN_sd <- c(.9901429)
logMPN_samp = rnorm(n_sim, logMPN_mean, logMPN_sd)
MPN_samp = 10^logMPN_samp
MPN_samp_halfgal = MPN_samp * 1900 #MPN per half gallon (1892.71 mL in half gallon)

#Temperature data
stages <- read.csv("temp_stages.csv", stringsAsFactors = F, comment.char = "#")
```

```
## Warning in read.table(file = file, header = header, sep = sep,
```

```
## quote = quote, : incomplete final line found by readTableHeader on
## 'temp_stages.csv'
```

```r
#Generate initial MPN for each half gallon from Poisson distribution
#Also sample AT for each half gallon
MPN_init<-vector()
allele <- vector()
temps <- vector()
for (i in 1:n_sim){
  MPN_init_samp <-rep(rpois(n_halfgal, MPN_samp_halfgal[i]), times = n_day)
  MPN_init<-c(MPN_init, MPN_init_samp)
  allele_samp <- rep(sample(freq_data, n_halfgal, replace = T), times = n_day)
  allele <- c(allele, allele_samp)
  #now calculate temp
  for (j in 1:nrow(stages)){
    stage_row <- stages[j, ]
    n_times <- stage_row$endTime - stage_row$beginTime + 1
    params <- as.numeric(unlist(strsplit(stage_row$parameters, " ")))
    temp_mean <- params[[1]]
    temp_sd <- params[[2]]
    temp_sample <- rep(rnorm(n_halfgal, temp_mean, temp_sd), times = n_times)
    temps <- c(temps, temp_sample)
  }
}
#add in temperature
data$temp <- temps

#Convert MPN_init from half-gallon to mLs
MPN_init_mL <- MPN_init / 1900
#remove 0's from the data and replace with detection limit
MPN_init_mL[MPN_init_mL == 0] <- 0.01;

#Now we add in those calculations to our original dataframe
data$logMPN_init <- log10(MPN_init_mL) #Add initial logMPN to data frame
data$AT<-allele #Add in AT data

data$newTemp <- vector(mode="logical", n_sim * n_halfgal * n_day)
data$newMu<- vector(mode="logical", n_sim * n_halfgal * n_day)
data$newLag <- vector(mode="logical", n_sim * n_halfgal * n_day)


##Now we will calculate the log10N for each row in the data frame
##Get the AT and day from the data frame, get growth parameters depending on the AT
# Simulation   ----
for (i in 1:(n_sim *n_halfgal * n_day)){
  #Find row in growth parameter data that corresponds to allele sample
  allele_index <- which(growth_import$rpoBAT == data$AT[i])
  row <- data[i, ]
  prev_row <- getPrevRow(data, row$BT, row$half_gal, row$day)
  update <- ifelse(nrow(prev_row) == 0 || row$temp != prev_row$temp, T, F)
  #calculate the new growth parameters using the square root model and our
  #sampled temperature

    newT <- row$temp
```

```
    newLag <- ifelse(update,
                     lagAtNewTemp(row$day, newT, growth_import$lag[allele_index]),
                     prev_row$newLag)
     old_lag <- ifelse(row$day==1, newLag, prev_row$newLag )
     # newLag <- ifelse(update & row$day <= old_lag & row$day > 1,
     #                  adjustLag(row$day, old_lag, newLag),
     #                  old_lag)
     newLag <- ifelse(update & row$day > 1,
                      adjustLag(row$day, old_lag, newLag, restartExp = F),
                      old_lag)
    newMu <-  ifelse(update,
                     muAtNewTemp(newT, growth_import$mumax[allele_index]),
                     prev_row$newMu)
  data$newTemp[i] <- newT
  data$newLag[i] <- newLag
  data$newMu[i] <- newMu

  #Calculate the log10N count using our new growth parameters
  newCount <- log10N_func(row$day, newLag, newMu,data$logMPN_init[i],growth_import$LOG10Nmax[allele_ind
  oldCount <- log10N_func(row$day-1, newLag, newMu,data$logMPN_init[i],growth_import$LOG10Nmax[allele_i
  data$count[i] <- ifelse(row$day==1, newCount,
                          prev_row$count + (newCount-oldCount))
}
```