# IWR summer school 2024 - process-based models tutorial

Peter Fransson and Julian Heidecke (Heidelberg University, IWR)

2024-09-26

## Contents



### Prerequisite

In this exercise we will be using the following software and packages:

- R
- tidyverse (R package)
- grDevices (R package)

### Introduction

This document contains exercises for Day 4 "Process based models" of the IWR Summer School on "Applied Modeling of Climate-Sensitive Infectious Diseases", September 26, 2024.

The aim of this tutorial is to give a hands-on experience of process-based models and Bayesian statistics.

We suggest that you write your solutions - both text and R code - in the R Markdown (the provided .Rmd file). Then, you have the questions and answers collected in one file. Often, code and output is given as part of the exercise, but you are of course welcome to play with the code and modify it according to your needs.

We start the tutorial with loading the tidyverse and grDevices package.

```
#load("...")
library(tidyverse)
library(grDevices)
```

## Instructions

## Part 1: *Modelling mosquito populations*

In this exercise we want to implement an ordinary differential equation (ODE) model representing the dynamics of a mosquito population. In the model we structure the mosquito population into different life stages. We consider mosquito eggs ($E$), mosquito juveniles ($J$), representing larvae and pupae, and female adult mosquitoes ($M$). The differential equations are a time-continuous model of how individuals die, transition between these life stages, and of the reproduction of new mosquito eggs by female adult mosquitoes. First, we write a function that evaluates the right-hand side of the differential equations and returns the result as a list. Furthermore, the function takes the following inputs:

- the time points t at which we want to get approximations to the solution of this model,
- y which is the state of the system,
- and parms which is a list of parameter values.

**Model description:** Female mosquitoes ($M$) produce $\phi$ eggs ($E$) per gonotrophic cycle. Female mosquitoes complete each gonotrophic cycle at the rate $a$. The average egg hatching time is $1/\delta_E$ days and juveniles ($J$) take on average $1/\delta_J$ days to develop into female adult mosquitoes. Eggs and juveniles are subject to a natural mortality rate of $\mu_E$ and $\mu_J$, respectively. Juveniles compete in their aquatic habitat for resources. This competition leads to an additional mortality rate $\alpha J$ that is proportional to juvenile density. A proportion $\omega$ of all adult mosquitoes emerging from the juvenile stage are female. Female adult mosquitoes have an average lifespan of $1/\mu_M$ days. Mathematically, this model is expressed by the following ODE (time unit: days):

$$\begin{aligned}\frac{dE}{dt} &= ... - \delta_E E - \mu_E E \\ \frac{dJ}{dt} &= \delta_E E - \alpha J^2 - ... \\ \frac{dM}{dt} &= ... - \mu_M M\end{aligned}$$

**Exercise 1.1:** Your task is to complete the equations describing the right-hand side of the ODE model in the function "RHS" based on the model description provided above. It may help you to first write down the missing parts of the equation on paper.

```
RHS = function(t, y, parms){
  # state variables
  E <- y[1]
  J <- y[2]
  M <- y[3]

  # right-hand side of ODE
  list(c(
    # eggs
     - delta_E*E - mu_E*E, #'*Complete line: (egg production missing)*
    # juveniles
    delta_E*E - alpha*J^2 - , #'*Complete line: (juvenile development and natural mortality missing)*
    # female adults
    - mu_M*M #'*Complete line: (emergence rate missing)*
  ))
}
```

## Part 2: *Simulating the mosquito model*

### Part 2a: *Implementing the Euler method for solving ODEs*

The Euler method is the simplest numerical method to solve ODEs. It is an iterative method that approximates the solution of an ODE by progressing step-by-step from an initial point $y_0$ at a initial time point $t_0$ to the final time point $t_{end}$. The method uses the derivative of the function at the current point to estimate the value at the next point. Specifically, let $\frac{dy}{dy} = f(t, y)$ denote our ODE with with initial condition $y(t_0) = y_0$ (**Note:** Here, both $y$ and $f(t, y)$ are vectors!). The Euler method estimates the next function value $y(t+h) \approx y_{n+1}$ at time $t+h$ from the previous function value $y(t) \approx y_n$, by adding the derivative at $y_n$, $dy = f(t, y_n)$, multiplied by the time step $h > 0$, i.e., $y_{n+1} = y_n + h \times dy$. The error of our approximation, the difference between the true solution $y(t)$ and the approximation $y_n$, will decrease with $h$, however the number of iteration necessary to complete the approximation increases proportionally to $h^{-1}$.

**Exercise 2a.1:** Below is the code for Euler's method. We have removed the lines where we calculate the next approximation $y_{n+1}$ and update time $t$ in the for loop, see comments in the code. Please complete the missing code lines.

```
Euler <- function(f,t0,tend,y0,parms,h){
  system_size <- length(y0)
  var_names = names(y0)
  N <- (tend-t0)/h #Number of steps

  y<-matrix(0,N+1,system_size) #Container for the solution y.
  colnames(y)<-var_names
  y[1,] <- y0 #Initial condition

  t <- numeric(length = N+1) #Container for the time line
  t[1] <- t0

  for (n in 1:N){
    dy <- f(t[n],y[n,],parms) #Derivative

    y[n+1,] <- #'*Complete line:Estimate next point.*
    t[n+1] <- #'*Complete line:Take a time step.*
  }

  solution <- as.data.frame(y)
  solution$t <- t

  return(solution)
}
```

Now we want to use the Euler algorithm to derive an approximate solution to our mosquito model described by "RHS". We will solve the equations on the time interval $[t_0 = 0, t_{end} = 365]$ with a stepsize $h$ and initial conditions $y_0$.

**Exercise 2a.2:** The list of parameters below is missing values for the parameters $a$, $\omega$, and $\mu_M$. Please assign values to these parameters based on the following (assumed) information:

- A female adult mosquito lays on average one egg raft every 5th day.
- A female adult mosquito has an average lifespan of 10 days.
- About 50% of all adult mosquitoes emerging from the juvenile compartment are female.

We store the approximate solution of the ODE system in the variable "sol".

```
t0 <- 0
tend <- 365
```

```r
h <- 0.1
# we define parameters globally for now, later we will utilize the parms argument of RHS
a <- #'*Complete line*
phi <- 120
delta_E <- 0.6
mu_E <- 0.15
alpha <- 0.003
delta_J <- 0.08
mu_J <- 0.05
omega <- #'*Complete line*
mu_M <- #'*Complete line:*

#define initial conditions with a !named! vector and solve the model
y0 <- c(E = 10, J = 0, M = 0)
sol <- Euler(RHS, t0, tend, y0, parms = c(), h) # for now we don't need parms
```

**Exercise 2a.3:** After storing the solution of the ODE system in the variable "sol", make plots of the solution for all three state variables $E$, $J$ ,$M$.

```r
# Set up the plotting area to have 2 rows and 2 columns
{
par(mfrow = c(2, 2), mar = c(4, 4, 2, 1), oma = c(1, 1, 2, 1))

# Plot E vs t
plot(, , type = "l", col = "red", lwd = 2, #'*Complete line:plot E vs t.*
     main = "Number of eggs over time", xlab = "Time (days)",
     ylab = "Mosquito eggs")

# Plot J vs t
plot(, , type = "l", col = "blue", lwd = 2, #'*Complete line: plot J vs t.*
     main = "Number of juveniles over time", xlab = "Time (days)",
     ylab = "Mosquito juveniles")

# Plot M vs t
plot(, , type = "l", col = "black", lwd = 2, #'*Complete line: plot M vs t.*
     main = "Number of female adults over time", xlab = "Time (days)",
     ylab = "Female mosquito adults")

# Add an overall title for the entire plot
title("Time Series Plots", outer = TRUE)
}
```

**Exercise 2a.4:** Describe the qualitative behavior of the solution and discuss why the system behaves like this with one of the tutors.

**Note:** Various more advanced algorithms have been developed to solve ODEs that provide better approximations and are numerically more stable than the Euler method. In practice, researchers often prefer to use these more advanced algorithms. Luckily, many implementations of algorithms are readily available in R packages and we do not have to code them from scratch. For instance, a commonly used R package is the deSolve Package. You can check it out for your own research: https://cran.r-project.org/web/packages/deSolve/index.html

### Part 2b: *Temperature-sensitive parameters*

So far we have assumed that the parameters in our model are constant over the simulation period. These parameters represent so called mosquito "life-history traits". For example, the parameter $\delta_J$ can be interpreted

as the reciprocal of the average time it takes for a juvenile mosquito to develop from egg hatching to adult mosquito emergence. The assumption that mosquito life-history traits are constant over time is not applicable in real-world scenarios. For example, mosquito life-history traits are highly sensitive to environmental conditions. One of the most prominent examples is the impact of ambient temperature on the mosquito life cycle, which results from the fact that mosquitoes are ectotherms (this means that they cannot regulate their body temperature). Therefore, their whole metabolism including developmental progress, reproduction, and mortality are impacted by changes in temperature. Based on experimental studies researchers have described how mosquito traits typically change in response to temperature. Although these representations often involve simplifying assumptions, they help tremendously to model changes in mosquito populations in response to changes in environmental conditions.

Here we want to incorporate temperature sensitivity into the ODE model and observe how the dynamics of the mosquito population change in response.

Assume that the juvenile mosquito development rate $\delta_J$ changes with respect to temperature $tmp$ as given by a Briere function $\delta_J(tmp) = q * tmp * (tmp - T_{min}) * \sqrt{(T_{max} - tmp)}$ with parameters $q = 4.06/100000, T_{min} = 2, T_{max} = 41$.

Moreover, assume that researches conducted laboratory experiments and found that the adult female mosquito mortality rate $\mu_M$ changes with temperature $tmp$ according to the inverse of a quadratic function: $1/(q * (tmp - T_{min}) * (T_{max} - tmp) + 0.1)$ with parameters $q = 1.43, T_{min} = 13, T_{max} = 32$ (We add 0.1 to the denominator to avoid division by zero. This will limit the mortality rate to a maximum of 10 which is close to immediate death). Additionally, assume that at all temperatures the mosquito mortality rate in the field is $\kappa$ times larger than in the laboratory.

**Exercise 2b.1:** Complete the implementation of temperature-dependent function "delta_J". Visualize the output of both functions, "delta_J" and "mu_M", along the temperature range 0-45°C.

```
delta_J <- function(tmp, q, Tmin, Tmax){
  ifelse(tmp>max(0,Tmin) & tmp<Tmax, , 0) #'*Complete line:*
  #'*calculate Briere function.*
}

mu_M <- function(tmp, q, Tmin, Tmax, kappa){
  ifelse(tmp>Tmin & tmp<Tmax, 1/((1/kappa*q*(tmp - Tmin)*(Tmax - tmp)) + 0.1), 10)
  #calculate inverse quadratic function
}

Tmp <- seq(0, 45, 0.1)

# Set up the plotting area to have 1 row and 2 columns
{
par(mfrow = c(1, 2), mar = c(4, 4, 2, 1), oma = c(1, 1, 2, 1))

# Plot Tmp vs delta_J
plot(Tmp, delta_J(Tmp, 4.06/100000, 2, 41), type = "l", col = "red", lwd = 2,
     main = expression(delta[J]), xlab = "Temperature (°C)",
     ylab = "Juvenile development rate")

# Plot Tmp vs mu_M
plot(Tmp, mu_M(Tmp, 1.43, 13, 32, 10), type = "l", col = "blue", lwd = 2,
     main = expression(mu[M]), xlab = "Temperature (°C)",
     ylab = "Mosquito adult mortality rate")

# Add an overall title for the entire plot
title("Temperature sensitive parameters", outer = TRUE)
```

```
}
```

We will now solve our model along the timespan of one year using these temperature-sensitive parameters. For simplicity, we assume that the daily temperature follows a sinusoidal curve over the year with an average yearly temperature "avg" and neglect temperature variations within a day. We assume that $t_0 = 0$ represents the first day of the year and $t = 364$ the last day of the year.

**Exercise 2b.2:** Incorporate the temperature sensitivity of $\delta_J$ and $\mu_M$ into the updated function "RHS_tmp", solve the model, and visualize the solution for the time series of adult female mosquitoes.

**Note**: In preparation of the following exercises we will now make use of the parms argument of the function "RHS_tmp" and provide values for the parameters $\alpha$, $\kappa$, and the average temperature "avg" when calling the function RHS_tmp. This will allow us to easily call the function using different values for these parameters.

```
tmp <- function(day, avg){ # function returning the temperature on a specific day in the year
  avg + 10*cos((2*pi*(day-182))/(365))
}

RHS_tmp = function(t, y, parms){
  # unpack parameter vector
  alpha <- parms["alpha"]
  kappa <- parms["kappa"]
  avg <- parms["avg"]

  delta_J <-  #'*Complete line: parameterize delta_J.*
  mu_M <- mu_M(tmp(t, avg), 1.43, 13, 32, kappa) #parameterize mu_M

  # state variables
  E <- y[1]
  J <- y[2]
  M <- y[3]

  # right-hand side of ODE
  list(c(
    # eggs
    a*phi*M - delta_E*E - mu_E*E,
    # juvenile
    delta_E*E - alpha*J^2 - mu_J*J - delta_J*J,
    # female adults
    omega*delta_J*J - mu_M*M
  ))
}

t0 <- 0
tend <- 365
h <- 0.01

parms <- c(alpha = 0.003, kappa = 5, avg = 15)
sol_tmp <- Euler(RHS_tmp, t0, tend, y0, parms, h)

{
par(mfrow = c(1, 2), mar = c(4, 4, 2, 1), oma = c(1, 1, 2, 1))

# Plot Tmp vs t
plot(sol_tmp$t, tmp(sol_tmp$t,15), type = "l", col = "black", lwd = 2,
```

```
     main = "", xlab = "Time (days)",
     ylab = "Temperature (°C)")

# Plot M vs t
plot(sol_tmp$t, sol_tmp$M, type = "l", col = "black", lwd = 2,
     main = "", xlab = "Time (days)",
     ylab = "Female mosquito adults")

# Add an overall title for the entire plot
title("Time Series Plots", outer = TRUE)
}
```

**Exercise 2b.3:** Discuss with one of the tutors the qualitative behavior of the trajectory. How does it differ from the constant parameters case in Exercise 2a.4?

**Note:** If you want to learn more about the "thermal biology" of mosquitoes and mosquito-borne disease check out this overview article: https://onlinelibrary.wiley.com/doi/10.1111/ele.13335

## Part 3: *(Markov chain) Monte Carlo Methods*

While it is often possible to determine plausible values of parameters (and initial conditions) using previous literature, these "prior" guesses are in most cases still subject to uncertainty. For example, the parameter $\kappa$ reflects the relative increase in mosquito adult mortality rate under field conditions compared to laboratory conditions. We may want to simulate the dynamics of a mosquito population in a very specific geographical location and wonder what a realistic value for $\kappa$ would be. Other researchers may have determined this parameter for a different mosquito population in another location. The reported value might be close to the parameter realization in our population but may very well differ due to differences in, for example, the presence of mosquito predators in our location.

### Part 3a: *Forward problem: Sensitivity of simulations to parameter uncertainty*

Here we want to investigate how uncertainty about parameter values propagates to uncertainty in our model simulations. As an example we will generate solutions to our model for 5 different values of the parameter $\kappa$. Going from a set of given parameters and a model (the "causality") to the model solutions/trajectories (the "effects/observations") is called solving the forward problem.

```
#create a vector "kappas" with 5 different values of the parameter kappa
M = 5
kappas <- seq(2,8,length.out=M)

#prepare a dataframe for storing simulation results
solutions <- as.data.frame(matrix(NA, nrow = (tend - t0)/h+1, ncol = M))

#simulate the model for each value in kappas and store the resulting female adult mosquito
#time series in "solutions"
for(i in 1:M){
  parms <- c(alpha = 0.003, kappa = kappas[i], avg = 15)
  solutions[,i] <- Euler(RHS_tmp, t0, tend, y0, parms, h)$M
}

#color gradient for plotting the M different solutions
colors <- colorRampPalette(c("red", "blue"))(M)

#plot the different solutions
{par(mar=c(5, 4, 4, 11)) # increase right margin to fit a legend
```

```
plot(sol_tmp$t, solutions[,1], type='l', lwd=2, col=colors[1],
      ylim=c(0,max(solutions)),  main = "Number of female adults over time",
      xlab = "Time (days)", ylab = "Female mosquito adults")
for(i in 2:M){
  lines(sol_tmp$t, solutions[,i], lwd=2, col=colors[i])
}
# Add a legend
legend("topright", inset=c(-0.45, 0),
       legend=sapply(kappas,
                     function(x) as.expression(substitute(kappa == B,
                                                          list(B = as.name(x))))),
       col=colors, lty=1, lwd=2, xpd=TRUE)
}
```

If we instead of pre-selecting a few different values of the parameter $\kappa$, assume that the parameter comes from probability distribution (which could reflect our "prior belief") we can propagate this distribution to distributions about the charactersitic of the corresponding model solutions by sampling from the parameter distribution and then solving the forward problem.

**Exercise 3a.1:** Generate 100 samples of the parameter $\kappa$ from a uniform distribution with minimum values 1 and maximum value 10. For each sample solve the forward problem and store the value of the simulated adult mosquito population time t=210 (M(210)) in a vector. Generate a histogram for the sample distribution of $\kappa$ and of the distribution of the the simulated adult mosquito populations at time t=210.

```
M <- 100 # number of draws

kappas <- #'*Complete line: generate samples of kappa.*
solutions <- numeric(M) # prepare vector for storing maximum value of the simulated
#adult mosquito population

for(i in 1:M){
  parms <- c(alpha = 0.003, kappa = kappas[i], avg = 15)
  solutions[i] <- #'*Complete line: store maximum value of the simulated adult mosquito population.*
}

# plot histograms for the sample distribution of $\kappa$ and for the distribution of the
#maximum values of the simulated adult mosquito populations
{
par(mfrow = c(1, 2), mar = c(4, 4, 2, 1), oma = c(1, 1, 2, 1))
# Plot M vs t
#'*Complete line: histogram for the sample distribution of $\kappa$*
# Plot M vs t
#'*Complete line: histogram for the distribution*
#'*of the mosquito populations at time t=210.*
}
```

### Part 3b: *Inverse Problem: Parameter estimation*

In the previous exercise, we plotted the distribution of the peak number of adult mosquitoes during the season when drawing parameter values, *kappas* values, from a uniform distribution. In this exercise, we will look at the inverse problem, i.e., we are given data (the "effects/observations"), and we want to find a distribution for $\kappa$ (the "causality") that reflects the likeliness of different values of $\kappa$, conditional on the data (the so called posterior distribution). More theoretical background will be provided in tomorrow's lecture.

For this exercise we will work with synthetic data which we will generate using our model (in this way we know the parameter value that generated the data and can easily check whether the resulting posterior

distribution is sensible). To this end, we will simulate our model using $\kappa = 5$. We will assume that the output of our model represents the median realization of the stochastic process that generates our data. More specifically, we assume that our data are realizations of independent random variables and we assume that the random variables are lognormally distributed with a median equal to the adult mosquito abundance predicted by our model, and a standard deviation on logscale equal to 0.25. We can think of the added stochasticity by the lognormal distribution as reflecting natural fluctuations in the mosquito population that our model does not capture. The data is given as the female adult mosquito abundance at the end of each week between day number 119 and 280.

We will then pretend that $\kappa$ is an unknown parameter and our task is to find the posterior distribution of $\kappa$, i.e., the probability density function of $\kappa$, conditional on our data.

We find an approximation of the posterior distribution by generating samples from this distribution. To this end, we apply the Metropolis-Hastings algorithm. The algorithm creates the posterior samples in a sequential manner. To start the algorithm, we provide an initial guess for the parameter and set this to be the current parameter value. The algorithm then iteratively proposes new parameter values, which are drawn from the so called proposal density distribution. The new proposed parameter is accepted with a probability equal to the ratio of posterior probabilities between the new and current parameter values (acceptance ratio). If the proposed parameter is accepted, we set this as the new current parameter value; otherwise, the new current value is the same as the old value. This new current value becomes our next sample. The algorithm continues until we have generated the desired number of samples.

First we generate our synthetic data.

```
parms <- c(alpha = 0.003, kappa = 5, avg = 15) # true parameter values
sol <- Euler(RHS_tmp, t0, tend, y0, parms, h)
end_of_weeks_idx <- seq(11901,28001,700) #Indices corresponding to the end of
#each week between day number 119 and 280
data_p <- sol$M[end_of_weeks_idx] #Data points (female adult mosquito abundance
#at the end of each week)
sigma = 0.25
for(i in 1:length(data_p)){
  data_p[i] <- rlnorm(1, meanlog = log(data_p[i]), sdlog =0.25)
}

#the line represents the median of the data generating process, the dots represents the
#simulated data points
{
plot(sol$t,sol$M,type = "l", col = "black", lwd = 2,
     main = "Number of female adults over time (alpha = 0.003, kappa = 5)", xlab = "Time (days)",
     ylab = "Female mosquito adults", ylim = c(0,max(c(sol$M,data_p))))
points(sol$t[end_of_weeks_idx],data_p)
}
```

To calculate the acceptance ratio in the Metropolis-Hastings algorithm we need to evaluate the likelihood of the data given a value of $\kappa$. In our case we know that the data was generated from a lognormal distribution. The function below evaluates the log likelihood based on the lognormal distribution, a model output "sol_temp" (that corresponds to a certain value of $\kappa$), and the logscale standard deviation sigma.

```
log_likelihood_lognormal <- function(data, sol_temp, sigma) {
  n <- length(data)
  log_data <- log(data)
  mu <- log(sol_temp)  # sol_temp is the mean of the lognormal distribution
  log_likelihood <- -n/2 * log(2 * pi * sigma^2) - (1/(2 * sigma^2)) * sum((log_data - mu)^2)
  return(log_likelihood)
}
```

And now we create samples using the Metropolis-Hastings algorithm (this may take a while... grab a coffee :) )

```
n_samples = 300 #Number of samples

kappas <- rep(0, n_samples) #Container of kappa samples

kappa_current <- 3 #Initial guess

parms_temp <- c(alpha = 0.003, kappa = kappa_current, avg = 15)
sol_temp <- Euler(RHS_tmp, t0, tend, y0, parms_temp, h)


logL_current <- log_likelihood_lognormal(data_p, sol_temp$M[end_of_weeks_idx], sigma)

#Create kappa samples (Metropolis-Hastings algorithm)
for(i in 1:n_samples){
  kappa_guess <- rnorm(1, mean = kappa_current, sd = 0.5) #Propose new kappa value
  kappa_guess <- max(0.0001,kappa_guess) #Ensure that kappa>0

  parms_temp <- c(alpha = alpha, kappa = kappa_guess, avg = 15)
  sol_temp <- Euler(RHS_tmp, t0, tend, y0, parms_temp, h)

  logL_guess <- log_likelihood_lognormal(data_p, sol_temp$M[end_of_weeks_idx], sigma)

  if (runif(1, min = 0, max = 1) < exp((logL_guess-logL_current))) { #Check
#acceptance of proposed kappa
    kappa_current <- kappa_guess
    logL_current <- logL_guess
  }

  kappas[i] <- kappa_current
}

hist(kappas, freq=F) #Distribution of kappa samples
```

**Exercise 3b.1:** Calculate the median, mean, standard deviation, and 95% credible interval for $\kappa$. Does this information provide a good estimate of the true value of $\kappa$?

```
kappa_median <- #'*Complete line: Median. Complete line*
kappa_mean <- #'*Complete line: Mean.*
kappa_sd <- #'*Complete line: Standard deviation.*
kappa_CI <- #'*Complete line: 95% credible interval.*

print(kappa_median)
print(kappa_mean)
print(kappa_sd)
print(kappa_mean)
print(kappa_CI)
```

**Note**: Usually, the acceptance step in the Metropolis Hastings algorithm requires calculating the ratio of posterior probabilities between the proposed and current parameter values. However, in the implementation above we calculate the acceptance ratio solely based on the likelihoods of the data given the current and the proposed value of $\kappa$. This is equivalent to assuming a uniform prior distribution for $\kappa$. In other words, a priori ("before seeing the data") we do not consider any values of $\kappa$ more likely than others. In this case calculating

the acceptance ratio by evaluating the posterior or by evaluating the likelihood becomes equivalent.

**Exercise 3b.2:** Now, we change the code so that we estimate $\alpha$ instead of $\kappa$. We keep the same data and make the following changes to the code:

- Rename *kappas* to *alphas*.
- Rename *kappa_current* to *alpha_current*
- Set the initial *alpha_current* to 0.0024.
- Rename *kappa_guess* to *alpha_guess*
- Change the standard deviation (*sd*) of the proposal density distribution from 0.5 to 0.0001.
- Make the appropriate changes to *parms_temp*.

Run your code to verify that it works, and show your results and code to one of the tutors.

## Part 4: *Climate Change*

Let us now assume that our model is well calibrated. Using our model, we want to explore how increases in average daily temperature impact our mosquito population simulations.

**Exercise 4.1:**: Simulate the model with our baseline assumption of an average temperature "avg" of 15 ("baseline scenario"), with an increased average temperature "avg" of 17 ("increased scenario"), and an even higher average temperature "avg" of 20 ("extreme scenario"). Describe what you see. Try to find an explanation for the observed outcome and discuss your thoughts with one of the tutors.

```
parms_base <- c(alpha = 0.003, kappa = 5, avg = 15) # parameter vector for baseline scenario
sol_tmp_base <- Euler(RHS_tmp, t0, tend, y0, parms, h) # model solution for baseline scenario
tmp_base <- tmp(sol_tmp_base$t, avg = 15) # temperature curve in baseline scenario

parms_plus2 <- #'*Complete line: parameter vector for increased scenario.*
sol_tmp_plus2 <- #'*Complete line: model solution for increased scenario.*
tmp_plus2 <- #'*Complete line: temperature curve in increased scenario.*

parms_plus5 <- #'*Complete line: parameter vector for extreme scenario.*
sol_tmp_plus5 <- #'*Complete line: model solution for extreme scenario.*
tmp_plus5 <- #'*Complete line: temperature curve in extreme scenario.*



# plot the temperatures and model solutions (only female adult mosquitoes) for each
#temperature setting
{
par(mfrow = c(1, 2), mar = c(4, 4, 2, 1), oma = c(1, 1, 2, 1))
# Plot tmp vs t
plot(sol_tmp_base$t, tmp_base, type = "l", col = "black", lwd = 2, #plot temperature curves
#vs t in baseline scenario.
     main = "", xlab = "Time (days)",
     ylab = "Temperature (°C)",
     ylim=c(min(tmp_base, tmp_plus2, tmp_plus5), # adjust limits of y-axis to fit all
#line plots
            max(tmp_base, tmp_plus2, tmp_plus5)))
lines(sol_tmp_base$t, tmp_plus2, col="orange", lwd = 2) #plot temperature curves vs t in
#increased scenario.
lines(sol_tmp_base$t, tmp_plus5, col="red", lwd = 2) #plot temperature curves vs t in
#extreme scenario.

# Plot M vs t
```

```
plot(sol_tmp_base$t, sol_tmp_base$M, type = "l", col = "black", lwd = 2, #plot female
#adult mosquito curves vs t in baseline scenario
     main = "", xlab = "Time (days)",
     ylab = "Female mosquito adults",
     ylim=c(min(sol_tmp_base$M, sol_tmp_plus2$M, sol_tmp_plus5$M), # adjust limits
#of y-axis to fit all line plots
            max(sol_tmp_base$M, sol_tmp_plus2$M, sol_tmp_plus5$M)))
lines(sol_tmp_plus2$t, sol_tmp_plus2$M, col="orange", lwd = 2) #plot female adult
#mosquito curves vs t in increased scenario.
lines(sol_tmp_plus5$t, sol_tmp_plus5$M, col="red", lwd = 2) #plot female adult
#mosquito curves vs t in extreme scenario.
title("Climate Change plots", outer = TRUE)
}
```

## Some references for further reading


- Gelman, A., Carlin, J. B., Stern, H. S., & Rubin, D. B. (2013). Bayesian data analysis. Chapman and Hall/CRC.
- Mordecai, E. A., Caldwell, J. M., Grossman, M. K., Lippi, C. A., Johnson, L. R., Neira, M., ... & Villena, O. (2019). Thermal biology of mosquito-borne disease. Ecology letters, 22(10), 1690-1708.
- Smith, D. L., Battle, K. E., Hay, S. I., Barker, C. M., Scott, T. W., & McKenzie, F. E. (2012). Ross, Macdonald, and a theory for the dynamics and control of mosquito-transmitted pathogens. PLoS pathogens, 8(4), e1002588.
- Erguler, K., Chandra, N. L., Proestos, Y., Lelieveld, J., Christophides, G. K., & Parham, P. E. (2017). A large-scale stochastic spatiotemporal model for Aedes albopictus-borne chikungunya epidemiology. PloS one, 12(3), e0174293.
- Liu-Helmersson, J., Stenlund, H., Wilder-Smith, A., & Rocklöv, J. (2014). Vectorial capacity of Aedes aegypti: effects of temperature and implications for global dengue epidemic potential. PloS one, 9(3), e89783.
- Liu-Helmersson, J., Rocklöv, J., Sewe, M., & Brännström, Å. (2019). Climate change may enable Aedes aegypti infestation in major European cities by 2100. Environmental research, 172, 693-699.