A User's Guide to MBITES

Mosquito Bout-based and Individual-based
Transmission Ecology Simulator

MBITES Development Team:
Sean Wu, Hector Sanchez, Qian Zhang, John Henry, Daniel Citron,
Amit Verma, Arnaud Le Menach, David L Smith

Introduction

What is MBITES?

This document is a users guide to MBITES (Mosquito Bout-based and Individual-based Transmission Ecology Simulator), a software package for simulating mosquito-borne pathogen transmission and vector control.

MBITES is comprised of several parts: a **landscape** describing resources used by adult mosquitoes; the algorithms describing adult behavior as a sequence of flight **bouts**; a set of configurable **options** for simulating other aspects of mosquito life-history and behavior; **vector control**; and the aquatic ecology. MBITES also includes functionality for simulating pathogen transmission as part of a larger software package called MASH (Modular Analysis through Simulation of human Health).

The following five sections describe these parts of MBITES. Each section begins with a brief summary or heuristic followed by implementation notes, relevant details, and a guide to the R code.

The MASH Framework

NOTE: *The following section gives some background and vocabulary for understanding some of the basic concepts and design principles of MASH and MBITES. We recommend reading it at some point, but feel free to skip it if you want to get on with learning MBITES.*

MBITES is a flexible module that runs the adult mosquito component of MASH. The terms "component" and "module" and "flexible" all have a specific meaning in MASH. The following section explains the terminology used in MBITES and MASH and the rationale for developing it.

MASH is a framework for simulating human health events and related phenomena (*e.g.,* pathogen transmission by mosquito populations). MASH is a simulation modeling framework, not a simulation model. To put it another way, the problem MASH was trying to solve was how to build a plug-and-play simulator for human health, not how to simulate human health. The core design problem for MASH was how to build an API (application program interface) defining how various components of a model interact. The philosophy of MASH was that every component should have at least two modules: one would be highly realistic and simulate the relevant phenomena in exquisite detail, and the other would be dead simple. The first disease to be implemented within MASH was malaria.

The collection of all mechanistic models of human malaria have the same basic structural parts: a model of human malaria infection and immunity; a model of mosquito populations, including infection and development of malaria parasites; and some model of immature mosquito populations in their aquatic habitats. In MASH, each one of these parts is called a `COMPONENT`. Four distinct biological processes determine how these three components interact: malaria parasites can be passed from mosquitoes to infect humans when mosquitoes probe in anticipation of taking a blood meal; parasites infecting humans can be passed in the blood meal, infecting mosquitoes; adult mosquitoes lay eggs in aquatic habitats; and the eggs hatch, develop through four larval instars, then pupate and emerge as imago (the mature form) from the aquatic habitats to begin their lives as volant adults. A structure allowing MASH components to interact is called a `QUEUE`. In its most general form, transmission describes the events when pathogens are transmitted, either from mosquitoes to humans or *vice versa*, but it also the dispersion of pathogens, the distances traveled by the pathogens between infection events. The `LANDSCAPE` is the part of the model that determines how dispersion is defined.

To be truly plug-and-play, it must be possible to run a component in standalone mode, by passing the inputs from other components trivially as parameters. This is what we have done here.

MBITES is the exquisitely detailed module for simulating the blood feeding behavior of adult mosquito populations. This document also describes how MBITES is coupled to the component describing the aquatic ecology. We describe how MBITES can be used to simulate pathogen transmission among populations of humans and mosquitoes; here we describe how that would happen without actually simulating transmission. This document describes how to run MBITES in standalone mode.

The structure of MASH is summarized in the following way:

- MASH Structure
  - A population of `HUMANS` and their health states and health events is the core of MASH.
  - A `COMPONENT` is a basic functional part of any human health model.
  - A `LANDSCAPE` defines the spatial scaffold for interactions.
  - A `QUEUE` is a component level structure allowing two distinct components to interact.
- A `Module` is a specific set of algorithms that instantiates a component. A module could be flexible in the sense that the same process could be represented by one of many different functional forms.
- A `Synthetic Population` is the fully configured set of initial conditions.

MASH is a framework for building models that is modular and flexible. This gives rise to a complementary vocaublary to describe the models that arise. We consider MBITES to be a CLASS of models:

- Two models belong to the same ORDER if they belong to the same CLASS and share the same state space.
- Two models belong to the same FAMILY if they belong to the same ORDER and use the same functional forms.
- Two models belong to the same GENUS if they belong to the same FAMILY and share the same parameters.
- Two models belong to the same SPECIES if they belong to the same GENUS and share the same synthetic population.
- An individual model is an individual member of a SPECIES defined by its random number seed.

Species-level similarity is the default meaning of **model** in MASH. Without some explicit way of mapping the state space, it is difficult to compare two models belonging to a different order. This hierarchical nomenclature system (borrowed from Linnaeus) makes it possible to talk about models defined by some other level of similarity (*e.g.,* model families).

1 Landscape

MBITES uses a landscape structure called `MICRO`. We think of a `MICRO` landscape as being comprised of different point sets on which mosquito activities occur:

f :: haunts or blood feeding sites where feeding occurs

l :: habitats, short for aquatic habitats where egg laying occurs

s :: sugar, or sites where sugar feeding occurs

m :: mating sites

In MBITES, blood feeding is an event that occurs only at a blood feeding site, which typically include all the houses and other dwellings where people live. Another set of points describes where humans go to seek care:

c :: clinics or community health workers

While this is the way we think about landscapes, there is in fact one master point set where *any* events happen, and each unique point is called a *site*. A set of boolean switch describes the kinds of events (*e.g.,* blood feeding, egg laying, sugar feeding, mating, or care seeking) occurring there.

Each site is determined by a location in space (x, y), and it has a *site type*, which determines how mosquitoes rest (*e.g.,* around a homestead; `restingSpot()` is described in Options : Resting Hazards).

There are also other properties of each site. Each site has a pair of numbers that determine its location (x, y). Each type of function $(*)$ has a search weight associated with it for each species: $\omega_{i,*}$.

The landscape provides the spatial template upon which objects from different components of the model can interact. Humans and mosquitoes interact mainly through blood feeding, but for this to occur, there must be a way for humans and mosquitoes to meet up on the landscape. The structures that allow for components to interact are called queues.

atRiskQ The `atRiskQ` holds information about all the blood hosts who are present at each blood feeding site. A paired function, called `chooseHost()` returns the identity of the host a mosquito will approach during the blood feeding attempt.

eggQ Each aquatic habitat has an `eggQ` that olds all the eggs laid at each aquatic habitat. This queue facilitates modularity by making it possible for any adult female module to interact with any ecology module describing immature aquatic forms.

imagoQ The imago is the mature adult insect. The `imagoQ` holds females emerging from aquatic habitats after pupation. This queue facilitates modularity by making it possible for any adult female module to interact with any ecology module describing immature aquatic forms.

2 Mosquito Bouts

A mosquito bout tracks a mosquito from its launch to initiate one bout up to the point when it launches itself to initiate the next bout. Conceptually, the structure of a bout is: **Launch - Try - Land - Rest**. The **Try** part of the bout is found in code `boutX` where `X` is one of the following:

F :: A search to find a place where blood feeding can occur

B :: An attempt to blood feed on a host that is present

R :: The post-prandial resting bout

L :: A search to find a place where egg laying can occur

O :: An attempt to lay eggs

M :: Mating, which could include a search flight

S :: Sugar feeding, which could include a search flight

After the bout, a mosquito lands at a particular kind of place, determined by the function `restingSpot()`. The mosquito's energetic state is then updated, with `sugarEnergetics()`.

Mortality can occur as a result of flight stress or from one of the many hazards during landing. At the end, survival is computed by their associated functions: `surviveResting()` and `surviveFlight()`.Finally, every mosquito event is logged through `historyTrack()`.

Implementation Notes All bouts are implemented within a generic bout framework, in the function `mbites_oneBout()`.

In computing the outcome of a flight bout, variables track the event times (`tNow`, and `tNext`) – the moment when it started the bout; the mosquito's location among the point sets (`locNow`, and `locNext`); and the mosquito's present and future state (`state`, and `stateNew`). Before starting a new flight bout, `tNext` has been checked to be sure other dependent elements of the simulation have been correctly set. Upon entering the bout, the mosquitoes time, state, and location are updated:

- `tNow = tNext`
- `tNext = timing()`
- `locNow = locNext`
- `locNext = moveMe()`
- `state = stateNew`

The `state` (=`X`) determines which kind of bout occurs; `boutX` is run, setting `stateNew`.

To avoid errors caused by updating `state` at the wrong time, all calculations within a bout use the *present* time, location, and state: `tNow`, `state`, and `locNow`.

To avoid resurrecting a dead mosquito or resuscitating an estivating mosquito, any code that could change a mosquito's behavioral state is run only if the function `isActive()` returns `TRUE`.

R-Code

```
> mbites_oneBout <- function(){
+
+   # update time and state
+   private$tNow = private$tNext # update time
+   private$state = private$stateNew # update current state
+   self$timing() # update tNext
+
+   # movement
+   self$moveMe()
+
+   # bout
+   switch(private$state,
+     F = {self$boutF()},
+     B = {self$boutB()},
+     R = {self$boutR()},
+     L = {self$boutL()},
+     O = {self$boutO()},
+     M = {self$boutM()},
+     S = {self$boutS()},
+     {stop(cat("illegal behavioral state: ",private$state,"\n",sep=""))}
+   )
+
+   # landing spot
+   self$restingSpot()
+
+   # energetics
+   self$sugarEnergetics()  # MBITES-Generic-Energetics.R
+
+   # survival
+   self$surviveResting()     # MBITES-Generic-Survival.R
+   self$surviveFlight()      # MBITES-Generic-Survival.R
+
+   # log history
+   private$history$historyTrack(privateEnv = private, alive = self$isAlive())
+ }
```

2.1 Blood Feeding Search Bout [F]

If a female mosquito is not in the neighborhood of a blood meal host, she must search for one. She might also initiate a new search bout, even if the site she occupies is suitable for blood feeding. In some cases, such as peri-domestic breeding, where it is not necessary to move to find a bloodmeal host, she might initiate a blood feeding attempt bout without a blood meal search bout.

A flight search bout for blood feeding could begin anywhere, but it always ends at a *feeding site* or *haunt*:

$$\{l\} \cup \{f\} \cup \ldots \overset{\mathcal{K}_F}{\to} \{f\} \tag{1}$$

The outcome of a search bout is either mosquito death, a new (possibly the same) location and behavioral state, or a frustrated search attempt without a change of behavioral state.

The function $\mathcal{K}_F$ is implemented by `rMove()`, described in the options below.

Configuring the Blood Feeding Search Bout

Implementation

- The function `rMove()` is called to determine the mosquito's next location.
- A parameter, `bfsb.s`, determines whether a mosquito has a "success" on its approach to the haunt. If the flight was a "success," then its behavioral state is set to **B**. Otherwise, it remains in state **F**.
- The function `restingSpot()` is then called. A mosquito can fail to find a suitable resting place and thus decided to leave (denoted `bfrs.f`) in which case its behavioral state is set to **F**.
- The function `sugarEnergetics()` is then called. A mosquito can fail to find a suitable resting place and thus decided to leave (denoted `bfrs.f`) in which case its behavioral state is set to **F**.
- The functions `surviveFlight()` is then called, which computes the probability of surviving the flight, `bfsb.fs`. If a mosquito dies, its state is set to **D**.
- The functions `surviveResting()` is then called, which computes the probability of surviving from landing through launch, `bfsb.rs`. If a mosquito dies, its state is set to **D**.

Transition Probabilities The mosquito's ending state is:

D with probabilty $P_{FD} = 1 - \texttt{bfsb.fs} \times \texttt{bfsb.rs}$

F with probability $P_{FF} = (1 - P_{FD})(\texttt{bfrs.f} + (1 - \texttt{bfrs.f})(1 - \texttt{bfsb.s}))$.

B with probability $P_{FB} = 1 - P_{FF}$.

R-Code

```
> mbites_boutF <- function(){
+   if(private$lspot != "l" & runif(1) <
+      private$FemalePopPointer$get_MBITES_PAR("F_succeed")){
+     private$stateNew = "B"
+   } else {
```

```
+     private$stateNew = "F"
+   }
+ }
```

2.2 Blood Feeding Attempt Bout [B]

A blood feeding bout begins when a mosquito chooses a host to approach. Information about the human hosts present in each haunt are stored in a queue, called *atRisk*. Other potential blood meal hosts (*e.g.,* cattle) can also be present. Each host has a weight, which is used in a multinomial probability distribution function to select one of the hosts to approach.

After selecting a host, different functions determine the outcome of an attempt, depending on whether the host is human or another vertebrate animal. As a mosquito approaches a human attempting to blood feed, there are several steps: an approach, probing, and the blood meal. The mosquito could be frustrated or killed at each one of these steps (*e.g.,* from swatting). A mosquito blood feeding on a human becomes infected with a pathogen with some probability. Pathogens mature by a (possibly temperature dependent) rule determining EIP at some point in the future, if the mosquito is still alive.

The process for feeding on a non-human vertebrate animals is simpler; parameters determine the probability of death or success (and their complement, failure followed by a new attempt).

After successfully blood feeding, a mosquito's behavioral state changes to the post-prandial resting bout. If a mosquito survives the attempt but does not succeed in taking a blood meal, then it can either repeat a blood feeding attempt or leave the area to search for a new site.

If a mosquito has successfully blood fed, a random variate is drawn to determine the size of the blood meal, and there is another function that determines whether the mosquito survives the stress associated with blood feeding, depending on the size of the blood meal.

Post Prandial Resting Bout [R] The outcome of a post prandial resting bout is either death, another attempt to blood feed, or an egg laying search bout.

Transition to egg laying behaviors depends on a rule determining how eggs mature, depending on the time required for egg maturation *vs.* the time required for a feeding bout. If egg maturation occurs every feeding bout, then blood is provisioned into eggs; the larger the blood meal, the larger the egg batch. A mosquito could choose to feed again, depending on the size of the blood meal /

egg batch. Another option is that the eggs require time or blood resources to mature. If eggs are maturing, a mosquito repeats a blood feeding attempt until the eggs are mature.

If a mosquito survives, but the eggs are not mature or if she decides to top-up, she will attempt another blood meal. Even if eggs are mature, the mosquito may attempt to blood feed again, depending on the size of the last bloodmeal. Otherwise, a surviving mosquito makes an attempt to lays eggs.

Egg Laying Search Bout [L] A female mosquito must search for an egg laying habitat, though (as in the case of peri-domestic breeding) one might be readily available. In such cases, the "search" results only in a change of state. A flight search bout for blood feeding could begin anywhere, but it always ends at a haunt:

$$\{l\} \cup \{f\} \xrightarrow{L} \{l\} \tag{2}$$

The outcome of a search bout is either mosquito death, success and a change of location and behavioral state, or a frustrated search attempt with a change of location but without changing behavioral state. In some cases, such as peri-domestic breeding, where a mosquito finds herself already in an egg laying site, the mosquito might bypass an egg laying search flight.

Egg Laying Attempt Bout [O] An egg laying bout involves a series of egg laying attempts where a mosquito dies, successfully lays some fraction of the eggs and initiates a new egg laying search bout, or is frustrated. If frustrated, she will either either make another attempt to lay eggs or initiate a new attempt somewhere else.

Maturation

Male Mosquito Populations

Mating Bouts and Maturation Mating requires simulating male mosquito populations. MBITES can simulate both female and male individuals. After an adult female has emerged from her pupal case and hardened, she is considered to be *pre-gonotrophic*. The conditions for maturation include a mating requirement and (possibly) an energetic requirement, meaning that she could not lay eggs until she had mated and satisfied an early life energy intake requirement. The model has not pre-determined what these pre-gonotrophic energetic requirements are, but provides a flexible set of functions and parameters to stipulate, depending on the species and either evidence or expert opinion. Mating and maturation can also be turned off, such that a female is mature and mated upon emergence.

Male mosquitoes emerge from aquatic habitats and by default attempt to sugar feed (see below). At a specific time of each day, the male behavioral state switches to **M**, and it moves to a mating point, $\{m\}$, to be present for any female arriving at that point to mate. Male search thus always begins at a mating or sugar feeding site, $\{s\}$ to arrive at a mating site $\{m\}$:

$$\{s\} \cup \{m\} \overset{\mathcal{K}_M}{\to} \{m\} \tag{3}$$

If the male mosquito survives the flight, it enters a swarm. The *swarming queue* tracks all the male mosquitoes present at each point in $\{m\}$ on each day. After the swarm disperses, surviving males behavioral states switch back to **S** to sugar feed (see below).

A pre-gonotrophic female does not lay eggs, though she can blood feed. Female mating bouts in this model occur only when she is in a pre-gonotrophic state, which (like males) can be triggered during a specific time window each day, when her state switches to **M**. The female search kernel could thus begins at any mating, blood feeding, or sugar feeding site but always ends at a mating site $\{m\}$:

$$\{f\} \cup \{s\} \cup \{m\} \overset{\mathcal{K}_M}{\to} \{m\} \tag{4}$$

A female chooses from among the males in the queue at that location. The probability of success is a function of the number of males present in the swarm. When a female mosquito mates, the identity of the male mosquitoes she mated with are chosen from those present at the swarming site and stored. The outcome of a mating bout is either death, success, or failure. After a mating bout, a female's behavioral state switches to search for a blood meal, **F**.

An alternative to swarming behavior is opportunistic mating during some other activity (*e.g.,* blood feeding). If males are present where females blood feed, a random variate is drawn to see if the blood feeding attempt bout is interrupted and mating occurs. In such cases, ordinary hazards apply, and if the female survives, her behavioral state does not change. (NOTE: we need to add this possibility, just as we do for sugar feeding.)

Sugar Feeding Bouts and Energetics A variable track's the level of each mosquito's energy reserves, and each bout depletes a mosquito's energy reserves by some amount. These energy reserves could be restored by blood feeding, depending on the species (a parameter is set to determine how much energy a female mosquito is able to derive from blood). Otherwise, a mosquito must restore its energy reserves by sugar feeding. Sugar sources are distributed across the landscape, at the points in $\{s\}$, including possibly at swarming sites, haunts, and habitats. Sugar feeding could occur in two ways, through a sugar feeding bout or opportunistically.

A female mosquito's energy reserves are checked at the end of every other kind of bout, and a function determines whether a mosquito switches from some other state into an active sugar feeding behavioral state, thereby initiating a sugar feeding bout. A mosquito sugar feeding bout could begin at any haunt or at another habitat, but it always ends in a sugar feeding site:

$$\{f\} \cup \{l\} \cup \{m\} \cup \{s\} \xrightarrow{L} \{s\} \tag{5}$$

The outcome of a sugar bout is either death, a sugar meal, or another attempt. If a sugar feeding bout succeeds, the sugar reserves are topped up.

During any other kind of bout, opportunistic sugar feeding can be triggered in a female if sugar is present. The functions triggering obligate and opportunistic feeding are, by default, shifted so that a female mosquito is more likely to feed opportunistically than to actively sugar feed. The sugar reserves of a mosquito are checked. The sugar at the source has a "search weight," and functions determine whether a mosquito chooses to feed based on energy reserve levels and the weight. Another function determines how much the reserves are topped up. If a mosquito does sugar feed opportunistically, the mosquito could die during the attempt. If it survives opportunistic feeding, its behavioral state remains the same.

Male mosquitoes are obligate sugar feeders when they are not mating. Sugar feeding will occur repeatedly until the time window when mating occurs, when its state shifts back to mating.

Estivation [E] Estivation is a state of inactivity. In this model, estivation is induced seasonally, whenever certain conditions are met (*e.g.,* no rain). If a mosquito survives estivation, it initiates a blood feeding search bout at some point in the future.

3 Options

Other properties of a site include a set of queues that coordinate information sharing from one `COMPONENT` to another.

3.1 Time at Risk

3.2 Egg Laying

3.3 Emergence

3.4 Mosquito Dispersal

Mosquito dispersal occurs as the result of a search bout, called by the function `rMove()`.

3.4.1 Kernels

3.5 Surviving the Flight

When `surviveFlight()` is called, a random number is drawn to determine whether the mosquito survived the stress associated with flight. The function is called from **MBITES-GenericSurvival.R** and at the moment, it looks like this:

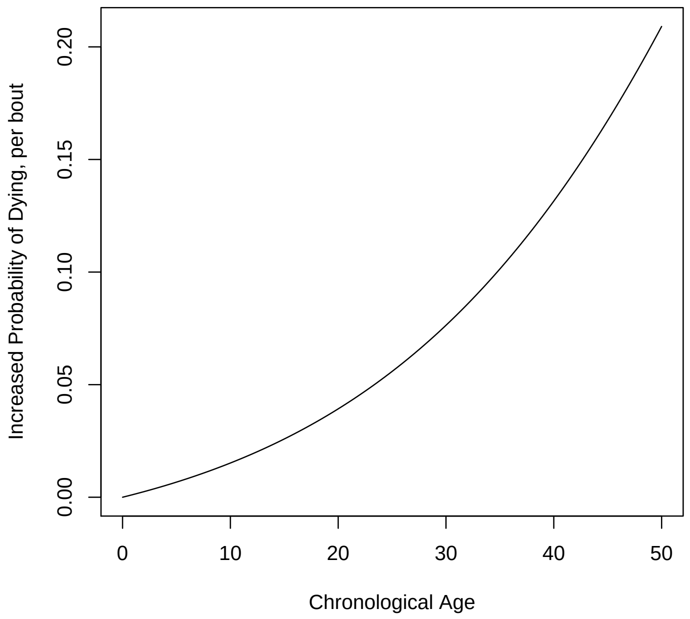
```
> mbitesGeneric_surviveFlight <- function(){
+   if(self$isActive()){
+     p = self$get_surviveFlightProb()
+     if(private$FemalePopPointer$get_MBITES_PAR("TATTER")){
+       private$damage = private$damage + self$WingTattering()
+       p = p * self$pTatter()
+     }
+     if(private$FemalePopPointer$get_MBITES_PAR("SENESCE")){
+       p = p * self$pSenesce()
+     }
+     if(runif(1) < 1-p){
+       private$stateNew = "D"
+     }
+   }
+ }
```

Baseline Survival The function `get_surviveFlightProb()` returns the probability of surviving. **I can't find this function**

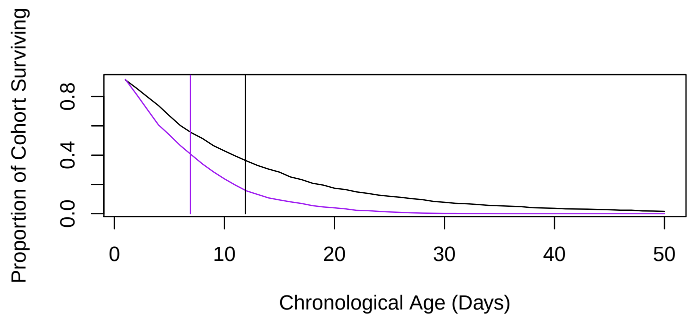
Senescence Senescence is defined in this model as a reduction in the probability of surviving that is associated with a mosquito's chronological age, a.

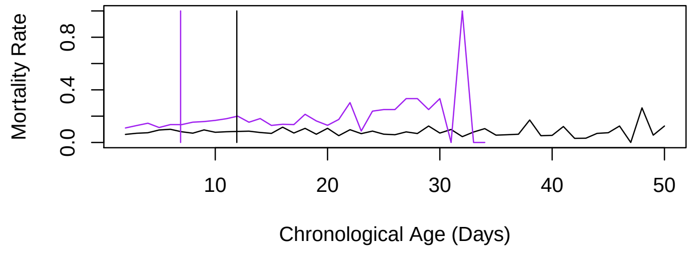
In MBITES, senescence has been implemented as an increased probability of mortality, per bite, that increases with the chronological age of the mosquito when it takes a flight bout. The function describing the probability of surviving is

$$\frac{2 + sns.b}{1 + sns.b} - \frac{e^{a*sns.a}}{sns.b + e^{a*sns.a}}.$$



NOTE: overall survival vs. age is difficult to compute without simulation. It would be good for the GUI to show surival of 1,000 mosquitoes with and without senescence, and to compute (and show) the average age. Something like this:





Wing Tattering

3.6 Resting Hazards

This discusses implementation and options for `surviveResting()`. One reason for determining where a host rests is for purposes of simulating various modes of vector control or entomological surveillance, including IRS, housing improvements, area repellants, eave tubes, or window traps.

3.6.1 A Simple Site

A simple type is given an index

R Code

```
> mbites_restingSpot <- function(){
+   if(self$isActive()){
+     if(self$searchFail()){
+       private$lspot = "l"
+     } else {
+       oldSpot = private$lspot
+       private$lspot = self$newSpot() # choose new lspot
+       if(oldSpot != "i" & private$lspot == "i"){
+         self$enterHouse() # enterHouse
+       }
+     }
+   }
+ }
```

3.7 The Blood Meal

3.8 Time to Event

4 Aquatic Ecology