2021

A Bayesian hierarchical mixture model with continuous-time Markov chains to capture bumblebee foraging behavior

Max Thrush Hukill
Bowdoin College

A Bayesian hierarchical mixture model with continuous-time Markov chains to capture bumblebee foraging behavior

An Honors Paper for the Department of Mathematics
By Max Thrush Hukill

Bowdoin College, 2021

Abstract

> Some say that unto bees a share is
> given of divine intelligence...
>
> –Virgil, *Georgics* IV: 221–222

The standard statistical methodology for analyzing complex case-control studies in ethology is often limited by approaches that force researchers to model distinct aspects of biological processes in a piecemeal, disjointed fashion. By developing a hierarchical Bayesian model, this work demonstrates that statistical inference in this context can be done using a single coherent framework. To do this, we construct a continuous-time Markov chain (CTMC) to model bumblebee foraging behavior. To connect the experimental design with the CTMC, we employ a mixture model controlled by a logistic regression on the two-factor design matrix. We then show how to infer these model parameters from experimental data using Markov chain Monte Carlo and interpret the results from a motivating experiment.

Acknowledgements

Despite the tragedy of the past fifteen months, the following people have risen to the occasion and made this experience a profoundly fulfilling capstone project for my undergraduate education. I am honored to have had the priviledge to work with all of them.

This project would not have been possible were it not for the efforts of Professor Patty Jones and her lab. They supplied the data and initial statistical analyses, in addition to the biological motivation for my model. For all of this, I am deeply grateful.

I want to thank the Bowdoin Mathematics Department for offering me a wonderful place to grow intellectually these past four years. Not only have they supported me throughout this project in hosting and encouraging my two honors talks, but they've instilled in me a passion for mathematical learning that will persist for years to come. I particularly want to thank Professor William Barker, Professor Naomi Tanabe, and Sam Harder for bringing me into the major in the first place, and for solidifying my deep love of mathematical problem solving. I am so fortunate to have met them when I did.

I want to thank my classmates over the years who have made the major the experience that it has been. The collaboration, care, and sense of humor that have defined my time here will remain in my fondest memories. Along those lines, I am particularly indebted to Jaya Blanchard, Dan Ralston, John Hood, Junyoung Hwang, Josh George, Charlotte Hall, Bryan Vargas, Huma Dadachanji, Juliana Taube, Sam Harder, and Connor Fitch.

I want to thank my advisor, Professor Jack O'Brien, for his integral role in my intellectual and statistical development. Ever since I entered his office as a curious calculus student, he has encouraged my study of probability and statistics with remarkable conviction. From his genuinely inspirational lectures, to his reliable assistance in office hours, to his bravery in debugging code, to his empathetic and deeply interested support as a research advisor, Jack has demonstrated the best of what the small liberal arts college experience has to offer. I could not be more grateful to him.

I want to thank my parents, Jane Thrush and Warren Hukill, and my partner, Mirai Hutheesing, for their steadfast support and willingness to endure ceaseless presentation practice. I simply could not have done this without you three.

Finally, I want to thank the brave bumblebees who gave their humble lives to make this science possible. Humanity owes you all.

Preface

> In their little bodies beats a mighty spirit.
>
> –Virgil, *Georgics* IV: 83

The following thesis presents a novel implementation of continuous-time Markov chains for statistical analysis. I begin with the mathematical foundations for the work, taking an abridged tour through probability theory, Markov processes, and Markov chain Monte Carlo inference methodology. I then present the motivating biological experiment, going through the intricacies of the set-up and standard approach to statistical analysis. After reflecting on the limitations of those current methods, I present the novel Bayesian hierarchical mixture model that defines the core of my work. After fully specifying the model, I offer an outline of my inference strategy, relying principally on Markov chain Monte Carlo, coded entirely in base R. With the tools defined, I then present simulation studies to test the theoretical limits of the algorithm. I proceed then to applying the inference algorithm to real data collected by the lab of Dr. Patty Jones at Bowdoin College. I apply various model checking methods to interrogate the veracity of the parametric inference accomplished by my model. I conclude with next steps, both immediate and more sophisticated, for augmenting this solution to an important statistical problem.

Please note that I fully built, tested, and applied two distinct algorithms: one using continuous-time Markov chains, and one using discrete-time Markov chains as an approximation. The latter has been moved to the appendix for clarity. The second appendix contains chapter-specific hints, such as proofs, supplementary figures, and interesting details. Feel free to jump between the main body of the work and these appendices. I, for one, found it immensely helpful to keep the discrete-time approximation in mind when developing the continuous-time model. Thank you for your readership!

The code for both models and all figures will be supplied in an open-source manner on GitHub. In the coming summer of 2021, we will be translating this work into a manuscript for journal submission, at which point the code will become available. For those members of posterity interested in this information, please reach out to me at maxhukill@gmail.com.

Contents

Chapter 1

Mathematical Background

1.1 Probability and statistical theory

1.1.1 Fundamentals

What follows should be comfortable to a reader with the basic set-theoretic foundation of probability and statistical theory [1]. The following table contains the notation conventions I will be using throughout this work.

Notation	Description
$X = x$	The event where a random variable X takes the value x
$\mathbb{P}(X = x)$	the probability of that event
$\pi_X(x)$	the probability density of X at value x
$\pi(x\|y)$	conditional probability distribution of x given y
$\theta \in \Theta$	a general parameter of interest θ in the parameter space Θ
D	general data, which we necessarily observe
$\boldsymbol{M}$	a matrix
$[M_{ij}]$	an entry in that matrix located in the i^{th} column and j^{th} row
$\vec{v}$	a vector
MLE	maximum likelihood estimate
PDF/PMF	probability density/mass function
CDF	cumulative density function
$\hat{p}$	MLE estimate for a parameter p

Table 1.1: Notation

To see this notation in context, I briefly review the standard axioms of probability. Following from above, a probability is a function that maps certain events within a sample space, $A \subseteq S$, onto the real line such that the following three axioms hold:

1. $\mathbb{P}(S) = 1$
2. $\mathbb{P}(A) \geq 0$
3. $\mathbb{P}(\bigcup_{i=1}^{\infty}) = \sum_{i=1}^{\infty} \mathbb{P}(A_i)$, where $\{A_i\}$ represent a collection of disjoint events.

At the heart of probability theory lies the random variable: a measurement of an experiment, a function from a sample space into $\mathbb{R}$, and a probability distribution. Throughout this work,

I will refer to individual event probabilities with $\mathbb{P}$ and probability distributions with π. For example, if I flip a fair coin with outcomes $X \in \{H, T\}$, the probability of landing on heads is $\mathbb{P}(H) = 0.5$. However, the probability mass of my Bernoulli random variable X at the point representing heads would be written as $\pi(X = H) = 0.5$. These reflect the same process, but with different emphases.

1.1.2 Likelihoods

At the heart of classical (or frequentist) statistics lies the concept of the likelihood, a function that encodes the probability of the data given the model parameters. For example, suppose we had a series of data independently and identically distributed (iid) from the same Poisson random variable. We would write this

$$x_1, x_2, \ldots, x_N \overset{iid}{\sim} \textsc{Poisson}(\lambda).$$

We would then write the likelihood as

$$\mathcal{L}(D|\Theta) = \pi(X_1 = x_1, X_2 = x_2, \ldots X_n = x_n) = \prod_{i=1}^{N} \pi(X_i = x_i|\lambda) = \prod_{i=1}^{N} \frac{e^{-\lambda}\lambda^{x_i}}{x_i!}.$$

Notice, it is composed of a product of probability density/mass functions, following from the assumption of independence. As we'll soon see, these likelihood structures can become arbitrarily complex to cope with real data.

Much of classical statistics focuses on deriving maximum likelihood estimates (MLEs) for Θ. In an MLE, the log of the likelihood is maximized, yielding the parameter values that (unsurprisingly) maximize the value of the likelihood. The MLE provides the fit of Θ with D, without including any other information that the researcher may have about Θ.

Seeking to include such information, Bayesian statistics applies Bayes' Law to the likelihood function,

$$\underbrace{\pi(\Theta|D)}_{\text{posterior}} = \frac{\overbrace{\pi(D|\Theta)}^{\text{likelihood}}\overbrace{\pi(\Theta)}^{\text{prior}}}{\underbrace{\int_{\Theta} \pi(D|\Theta)\pi(\Theta)\ d\Theta}_{\text{marginal}}}. \qquad \text{(Eq. 1)}$$

The distribution of our interest is the posterior distribution, which arises from the prior, likelihood, and marginal distributions via the update function in Eq. 1. The prior allows us to specify our belief about a set of processes quantitatively *before* seeing the data itself. This may indicate the bounds of possible values, or incorporate the findings of previous studies. The marginal, on the other hand, proves to be an often intractable quantity to find directly. Fortunately, as I outline in Section 1.3, we can harness the power of computation to sidestep this issue.

1.2 Markov chains

Markov chains are a foundational stochastic process ubiquitous in scientific modeling. The concept of Markov chains forms a consistent through line for this thesis in two key ways: how I model the data, and how I infer the model parameters. Specifically, the data are modelled by a novel continuous-time Markov chain, while the guiding inference strategy relies on Markov chain Monte Carlo—a method for simulating a Markov chain to approximate a distribution of interest. The stochastic process introduction has been adapted from [2], [3], and [4].

1.2.1 Discrete-time Markov chains

A stochastic process in discrete time $s \in \mathbb{N}$ is a sequence of random variables X_s indexed by time, represented $\boldsymbol{X} = \{X_s : s \in \mathbb{N}\}$. The range of X_s is called the state space $\mathcal{S}$, with each X_s taking a state within $\mathcal{S}$. In discrete time, the Markov property states that

$$\begin{aligned}\mathbb{P}(X_t = j | X_0 = i_0, X_1 = i_1, \ldots, X_{t-2} = i_{t-2}, X_{t-1} = i) &= \mathbb{P}(X_t = j | X_{t-1} = i) \\ &= P_{ij},\end{aligned}$$

where indices $\{i_0, \ldots, i_{t-2}, i, j\} \in \mathbb{N}$, and states $\{X_i\} \in \mathcal{S}$ for all indices. In other words, the system's future behavior does not depend on its past behavior, given its present state. (A stochastic process that has the Markov property is said to be a Markov chain.) Note that the notation P_{ij} reflects the probability of transitioning from state i to state j in one time step. We can then define the transition matrix specific to a given Markov chain:

$$\mathbf{P} = [P_{ij}],$$

with the necessary condition that $\sum_{j \in \mathcal{S}} P_{ij} = 1$. This condition ensures that we've formed a probability mass function for the process of going from state i to the next state j; that is, all options of j are not only enumerated, but they're assigned probabilities that will sum to 1. Consequently, transition matrices are square of size $|\mathcal{S}| \times |\mathcal{S}|$.

1.2.2 Continuous-time Markov chains

To extend the ideas of a DTMC to continuous time, we'll consider cases of discretely valued spaces, but now with time on the positive real line, as we experience it in the natural world. The Markov Property can be defined in these contexts as well: in a continuous stochastic process, denoted $\{X(t) | t \geq 0\}$ with some arbitrary state space $\mathcal{S}$, the Markov property is defined as

$$\mathbb{P}(X(t) = j | X(s) = i, X(t_{n-1}) = i_{n-1}, \ldots, X(t_1) = i_1) = \mathbb{P}(X(t) = j | X(s) = i)$$

where $0 \leq t_1 \leq t_2 \leq \cdots \leq t_{n-1} \leq s \leq t$ is any non-decreasing sequence of $n+1$ times (ignoring zero), and $i_1, i_2, \ldots, i_{n-1}, i, j \in \mathcal{S}$ are any $n+1$ states [3].

There are two critical features of this definition. First, these processes forget their past, as the conditional probability that depends on the entire chain's past is indistinguishable from that which is only conditioned on the previous state. I refer to this as *memorylessness*. Second, at least here, these processes are *time-homogeneous*: for any $s \leq t$ and any states $i, j \in \mathcal{S}$, we find that

$$\mathbb{P}(X(t) = j | X(s) = i) = \mathbb{P}(X(t-s) = k | X(0) = i).$$

The time spent in any given state i, denoted t_i, is a random variable called the ***dwell-time***. In the appendix corresponding to this chapter, I show that these dwell-times are necessarily exponentially distributed. Recall the definition of the exponential random variable with mean λ^{-1},

$$\pi(x|\lambda) = \lambda e^{-\lambda x}.$$

Using the exponential nature of dwell-times, we can consider an equivalent construction of a CTMC that will make certain things easier to work with. Consider the following process using $|\mathcal{S}|$ alarm clocks.

1. We begin in some state i of the chain at time $t = 0$.
2. At $t = 0$, we set $|\mathcal{S}|$ alarm clocks, one for each possible state $j \in \mathcal{S}$ (with $j \neq i$) that the chain could transition to. These alarm clocks are exponential random variables with rates, $\lambda_{i,j}$.
3. When the first alarm clock detonates, we transition to that state. This is now state j.
4. We reset the alarm clocks and continue.

This process is clearly a Markov chain in continuous time, but with dwell-times arising from minima of various exponentials. We can cluster these exponential rate parameters into a rate matrix $\boldsymbol{Q}$. Note, this is of the same square form as the DTMC transition matrices, although instead of representing probabilities, it represents infinitesimal rates. However, the diagonal entries are still undefined: in the CTMC construction it no longer makes sense to transition back to the current state; the dwell-time construction already accounts for that. In the DTMC, we'd "transition" each time unit regardless of whether or not we'd actually moved; in the CTMC, we dwell for a certain amount of time until we transition to a new state. As such, the diagonal entries of $\boldsymbol{Q}$ must be equal to the negative of the sum of the corresponding row:

$$-Q_{i,i} \equiv \sum_{i,j \neq i} Q_{i,j}.$$

The rate matrix (also called the infinitesimal or generator matrix), in conjunction with a state space and starting distribution, are sufficient to uniquely define a CTMC [3]. This then is the critical object of study in much the same way that a transition matrix functions for DTMCs. We will use this rate matrix to develop a CTMC likelihood (see Chapter 3).

To better understand these infinitesimal rate matrices, we will consider the instantaneous transition matrix at a given time, $\boldsymbol{P}(t)$. We'll need to employ the Chapman-Kolmogorov equations in continuous time [3]. The change in the transition probabilities can be given as a pair of matrix differential equations, stated

$$P'_{i,j} = \sum_{k \in \mathcal{S}} P_{i,k}(t) Q_{k,j} \quad \text{or in matrix form} \quad \boldsymbol{P'}(t) = \boldsymbol{P}(t)\boldsymbol{Q} \qquad \text{(Forward)}$$

$$P'_{i,j} = \sum_{k \in \mathcal{S}} Q_{i,k} P_{k,j}(t) \quad \text{or in matrix form} \quad \boldsymbol{P'}(t) = \boldsymbol{Q}\boldsymbol{P}(t) \qquad \text{(Backward)}$$

These are simple first order differential equations in matrix form, and so often tractable. Just as the solution to the initial value problem $f'(t) = cf(t)$ with $f(0) = f_0$ is $f(t) = f(0)e^{ct}$, the solution to these equations will be the matrix exponential.

$$\boldsymbol{P}(t) = e^{\boldsymbol{Q}t} \equiv \sum_{n=0}^{\infty} \frac{\boldsymbol{Q}^n t^n}{n!},$$

where $\boldsymbol{P}(0) = I$. This is the unique solution to the Chapman-Kolmogorov equations, and demonstrates the theoretical (if not practical) power the rate matrix has, in that it directly determines each instantaneous transition matrix for the CTMC [3]. (Proofs in cited sources.)

1.3 Markov chain Monte Carlo

Markov chain Monte Carlo (MCMC) is a powerful tool used across disciplines of modern science to recover complex probability distributions that cannot be analyzed exactly. The sheer quantity and diversity of applications of MCMC have earned it a place among the most important advances in empiricism to date, with one of its key examples, the Metropolis-Metropolis-Hastings algorithm (MHA), often cited as among the top-ten most important algorithms of the 20$^{\text{th}}$ Century [5], [6], [7]. It has been a paradigm-shifting development for not just statistics, but the use of probabilistic models across the sciences.

The details supporting MCMC have been left as citations. I will sketch the contours of the theory leading up to it (particularly the proof of the MHA, located in the appendix) in order to buttress intuition for this revolutionary idea.[1]

To begin, recall the definition of the posterior from Bayes' Law:

$$\underbrace{\pi(\Theta|D)}_{\text{posterior}} = \frac{\pi(D|\Theta)\pi(\Theta)}{\int_\Theta \pi(D|\Theta)\pi(\Theta)\ d\Theta}.$$

Our interest lies in the posterior distribution, and how to draw samples from it. Suppose we have no way to solve the integral analytically. Fortunately, MCMC allows sampling from the

[1]Please note, the following abridged tour of the theory behind MCMC contains elements that have been adapted from the very elegant synopsis composed by my friend and classmate, Huma Dadachanji (Bowdoin 2020), for our Bayesian statistics course in December 2019. See [2] for more complete details.

posterior distribution. Then, with these samples, we can construct an approximation of the posterior distribution.

A Markov chain is said to be *ergodic* when it meets the following three criteria:

- *Irreducible:* each state of the Markov chain can communicate with all other states (meaning for all states i there exists some path to state j in finite steps).
- *Aperiodic:* the chain contains no closed loops that induce periodic cycles.
- *Recurrent:* the chain has a positive probability of returning to each state.

The power behind ergodicity lies in the Fundamental Limit Theorem for Ergodic Markov Chains, which states that the limiting distribution of an ergodic Markov chain is the unique stationary distribution (proof in Debrow, Chapter 3.10). The following paragraph further develops these definitions.

Consider a stochastic transition matrix $\boldsymbol{P}$ with state space $\mathcal{S}$, where the states represent values for our parameters of interest, $\theta \in \Theta$. Indeed, this means that our state space has become infinite, and $\boldsymbol{P}$ is thus referred to as a kernel. We can now define the *stationary distribution*, which is a row vector $\vec{\lambda}_\theta = \{\lambda_\theta : \theta \in \Theta\}$ over the state space such that $\lambda \boldsymbol{P} = \lambda$ for a given Markov chain. Notice, the action of the Markov chain does not alter this distribution. Next, let us define a *limiting distribution* of a Markov chain, which is a row vector such that

$$\lambda_i = \lim_{N \to \infty} \frac{1}{N} \sum_{m=1}^{N} P_{ij}^m,$$

for all initial states i. This means that the distribution of λ is independent of the starting state. As it turns out, limiting distributions are necessarily stationary distributions. But then the question arises, is my given stationary distribution *the* one and only limiting distribution? Hence the power of ergodicity (see above).

The last piece we need is *reversibility*, a condition met by Markov chains that satisfy

$$\pi_j P_{ji} = \pi_i P_{ij},$$

for all $i, j \in \mathcal{S}$, where π is a stationary distribution. It can be shown that if a reversible chain is also ergodic, then π must be *the* one and only stationary distribution of $\boldsymbol{P}$ (proof in [2], Chapter 5). This condition is known as *detailed balance*.

The goal of MCMC then rests in constructing a $\boldsymbol{P}$ whose limiting (and thus stationary) distribution is a specific posterior distribution, $\pi(\theta|D)$. We now know this can be accomplished by ensuring that the chain is reversible and ergodic. This can be achieved using the Metropolis-Hastings Algorithm (MHA).

The Metropolis-Metropolis-Hastings Algorithm. We seek to sample from a target distribution, π^*, but cannot do so directly. However, we are able to sample from a stochastic matrix $\boldsymbol{Q}$ that can propose a new state j given a current state i (using $\mathcal{S}$ as the state space for both of these matrices). We can then construct the following Markov chain:

- Initialize the chain X_0 to some state $i \sim \alpha$, where α is a distribution over $\mathcal{S}$. Set $t = 1$.
- For $t > 0$:
  - Propose (sample) j from Q_{ij}.
  - Then, calculate
$$\alpha_{ij} = \frac{\pi^*(j) \cdot Q_{ji}}{\pi^*(i) \cdot Q_{ij}}.$$
  - Sample $U \sim \text{UNIFORM}(0, 1)$.
  - If $U < \alpha_{ij}$,
    - $X_{t+1} \leftarrow j$
  - Else
    - $X_{t+1} \leftarrow X_t$
  - Set $t = t + 1$.

If $\boldsymbol{Q}$ is ergodic, the MHA constructs a Markov chain X with a stationary distribution of π^* (proven in appendix). Despite its apparent brevity, this algorithm possesses a staggering quantity of potential.

Consider the specific case where π^* is our posterior of interest $\pi(\Theta|D)$. Notice the beauty of the Metropolis-Hastings ratio when we incorporate that fact:

$$\begin{aligned}
\alpha_{ij} &= \frac{Q_{ji}}{Q_{ij}} \cdot \frac{\pi^*(j)}{\pi^*(i)} \\
&= \frac{Q_{ji}}{Q_{ij}} \cdot \frac{\pi(D|\Theta_j)\pi(\Theta_j)}{\int_\Theta \pi(D|\Theta)\pi(\Theta)\ d\Theta} \div \frac{\pi(D|\Theta_i)\pi(\Theta_i)}{\int_\Theta \pi(D|\Theta)\pi(\Theta)\ d\Theta} \\
&= \frac{Q_{ji}}{Q_{ij}} \cdot \frac{\pi(D|\Theta_j)\pi(\Theta_j)}{\pi(D|\Theta_i)\pi(\Theta_i)}.
\end{aligned}$$

Sometimes the universe is kind to us—the intractable integrals of the marginal cancel, meaning we only need to specify the posterior up to proportionality, which we can do using only the likelihood and prior.

While we can rest assured that our Markov chains will *eventually* converge and enable our sampling of an arbitrarily complex posterior, in practice this may be slow. Indeed, it often ends up being more art than science in getting these chains to converge in feasible windows of time. As such, when we can, we'd like to simplify the above procedure as much as possible. We can often guarantee that our proposed values are always good enough to be accepted, and so omit the middle accept/reject step.

An algorithm that accomplishes exactly that is the Gibbs Sampler, which proposes new states of the chain from the full posterior distribution.

The Gibbs Sampler. The following is adapted from [8]. Suppose we have a K-dimensional posterior parameterized $\Theta = \{\theta_1, \theta_2, \ldots, \theta_k\}$. We seek to iteratively sample from the conditional probability distribution $\pi(\theta_k|D, \Theta_{-k})$, where k is one parameter in Θ and Θ_{-k} represents the set of all parameters in Θ other than k. The algorithm proceeds as follows:

- Initialize the chain to X_0 by assigning values from each of the k prior distributions. Set $t = 1$.
- For $t > 0$:
  - For each $k \in K$:
    - Sample
    $$\theta_k^* \sim \pi(\theta_k^{(t)} | D, \Theta_{-k})$$
    - Set $\theta_k^{(t+1)} \leftarrow \theta_k^*$
  - Set t = t+1.

This is a special case of the MHA where we've bypassed the checking step. To see this exactly, we will substitute the conditional posterior for the proposal distribution, and evaluate the MH-ratio. First, observe that by conditioning

$$\pi(\Theta|D) = \pi(\Theta_k, \Theta_{-k}|D) = \pi(\Theta_k|D, \Theta_{-k})\pi(\Theta_{-k}|D).$$

(For notation purposes, I will replace the i, j notation with the presence of an asterisk to indicate the subsequent state.) Then, we can evaluate the MH-ratio as

$$\begin{aligned}
\alpha_{ij} &= \frac{Q_{ji}}{Q_{ij}} \cdot \frac{\pi(D|\Theta^*)\pi(\Theta^*)}{\pi(D|\Theta)\pi(\Theta)} \\
&= \frac{\pi(\Theta|D, \Theta_{-k})}{\pi(\Theta^*|D, \Theta^*_{-k})} \cdot \frac{\pi(D|\Theta^*)\pi(\Theta^*)}{\pi(D|\Theta)\pi(\Theta)} \\
&= \frac{\pi(\Theta|D, \Theta_{-k})}{\pi(\Theta^*|D, \Theta^*_{-k})} \cdot \frac{\pi(\Theta^*|D, \Theta^*_{-k})\pi(\Theta^*_{-k})}{\pi(\Theta|D, \Theta_{-k})\pi(\Theta_{-k})} \\
&= \frac{\pi(\Theta^*_{-k})}{\pi(\Theta_{-k})} = 1.
\end{aligned}$$

The last equality follows from the fact that $\Theta_{-k} = \Theta^*_{-k}$ as k is the only difference between Θ and Θ^*. Since $\alpha_{ij} = 1$, we have $\mathbb{P}(U \leq \alpha_{ij}) = 1$, so no checking step is needed.

The natural question, of course, is how we can sample the conditional posterior in the first place. In our discrete-time inference (and in the typical Gibbs sampler), we rely on a phenomenon known as *conjugacy*, a powerful algebraic trick for certain probability distributions. Conjugacy occurs when a particular pair of likelihood and prior distributions yields a posterior distribution of the same form as the prior. The idea is to multiply the prior and likelihood PDFs, which are necessarily proportional to the posterior, and algebraically manipulate the expression until it is proportional to a PDF of the prior distribution (although this time with new parameters, of course).

Chapter 2

Experimental Design

2.1 Experimental motivation

The motivation for this work is ecological in nature. At the Bowdoin College Department of Biology, the lab of Patricia Jones studies bumblebee (*Bombus impatiens*) ethology—with particular interest in the interplay between color and secondary metabolites, in how they affect behavior. Our task in this project has been to develop a statistical procedure for analyzing one particular class of experiment conducted in her lab. The following chapter will outline the experimental design and standard statistical methodology for this class of experiment, setting the stage for our novel statistical work.

2.1.1 Physical description and core questions

The fundamental question of the experiment asks, **do the bees show a preference for their training color, given their nectar treatment?** The experiment was comprised of three phases: the initialization phase, the training phase, and the experimental phase. During the initialization phase, bumblebees from 10 colonies were permitted to forage freely in the experimental enclosure, with pollen supplied *ad libitum* (as needed). The enclosure, a 114cm x 69cm x 30.5cm plywood box with a clear plexiglass top, was positioned in a natural light greenhouse, and was attached to a clear tube with a sliding door to control entry. Inside the box, a grid of clear, artificial "flower" tiles (Perspex squares 22mm x 23mm x 3mm) sat atop glass vials containing 30% sucrose solution (the base nectar). This occurred for at least 2 days to permit bees within each colony time to learn to forage from artificial flowers in the enclosure.

With the basic initialization phase complete, the training phase then commenced. Instead of clear tiles with basic sucrose, each colony was assigned a specific treatment regimen to probe the effects of training color (blue or white) and secondary metabolites (caffeine or ethanol) on the training process. The bees had either 12 blue or 12 white tiles in their enclosure (Perspex Blue 727 or Perspex White); each flower contained either control (30% sucrose by volume), 10^{-5}M caffeine (in 30% sucrose), or 1% ethanol (in 30% sucrose). Shown below is the design matrix for all experiments, forming a two factor case-control. Each colony received exactly one color treatment and one nectar treatment, along with a letter classification (the number of individuals in the colony in parentheses).

Figure 2.1: Experimental apparatus. See the wood and plexiglass enclosure used for bumblebee foraging experiments. Note the blue and white tiles serving as flowers for the bees to forage from. These tiles contain different nectars depending on the experiment.

Nectar	Trained to Blue	Trained to White
Control	B (19), I (12)	A (8), H (14)
Ethanol	D (18), G (20)	F (19)
Caffeine	C (20), J (9)	E (15)

Table 2.1: Design matrix of colony treatments

After marking bees that successfully foraged in the training phase with paint, the bees later progressed to the experimental phase. The bees were tested individually for 5 minutes or until they attempted to return to the colony via the connector tube, whichever occurred first. Here, the bees were given a choice of flower color, but not nectar type. That is, instead of 12 tiles of the same color and nectar, the enclosure was outfitted with 6 tiles of each color forming a checkerboard pattern, all equipped with the same nectar solution as that to which the bees were trained. The Jones Lab hypothesized that bees would train stronger to blue than to white, that caffeine would augment training, and that ethanol would weaken training.

2.1.2 Translation of video data into coordinates

Before we proceed to the statistical attempts at modeling the experimental results, it is worth mentioning the logistical task of formatting the data produced in the experimental phase in such a way that it can be analyzed computationally. The raw data are video files (.mp4) taken of the experimental phase, shot from a birds-eye-view of the experimental apparatus. The machine intelligence software, DeepLabCut, which has recently revolutionized the animal ethology community [9], tracks the location of each bee, as well as that of each flower, as coordinates in 2D space, for each frame of the video.

These data, a series of xy-coordinates outlining all objects in the experiment, are then transformed into comma-separated values files (.csv) reflecting each new location (either a

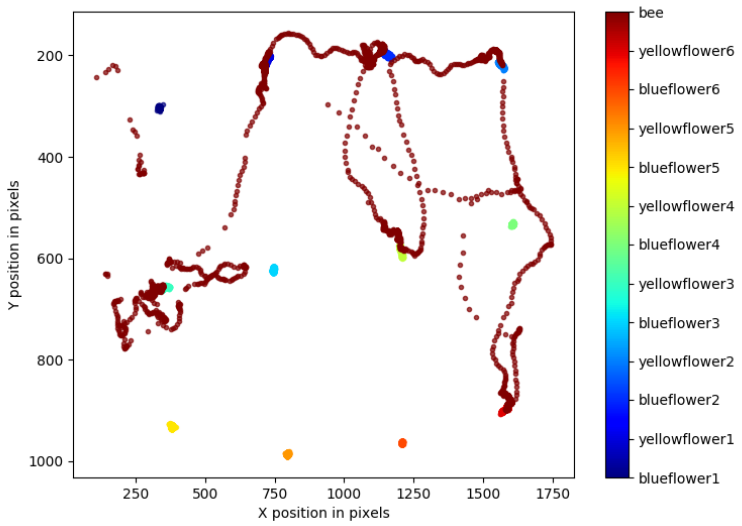


Figure 2.2: Raw data output. The machine intelligence software DeepLabCut tracks the location of the bees as they traverse their enclosure. The various colored dots represent tile locations, and the maroon dots represent a bee's location over time. These coordinate pairs are received in .csv file format.

specific flower or "flying") that the bee occupies, for all frames of the video. Lastly, these are simplified to reflect only the series of states the bees occupy and the time spent in each state: flying, blue tile, or white tile.

2.2 Standard approach to statistical analysis

The standard methodology for analyzing an experiment of this class relies on generalized linear-mixed models (GLMMs), a regression framework that can analyze either dwell-times or training proportions, though not both simultaneously. It is worthwhile to reflect on how GLMMs model the data within the context of these types of experiments, and what exactly I will attempt to improve upon in the subsequent, holistic model.

2.2.1 Generalized linear-mixed models

The idea behind a generalized linear-mixed model (GLMM) extends the capabilities of a generalized linear model to include both fixed and random effects [10]. In other words, we can consider random effects in the linear prediction process that generalized linear models cannot accommodate [11]. To see this, let us define the linear predictor η as

$$\eta = \mathbf{X}\vec{\beta} + \mathbf{Y}\vec{\gamma},$$

where $\mathbf{X}$ is an $N \times p$ matrix of p predictor variables; $\mathbf{Y}$ is an $N \times q$ matrix of q random-effect variables; $\vec{\beta}$ is a $p \times 1$ column vector of fixed-effect regression coefficients; and $\vec{\gamma}$ is a $q \times 1$ column vector of random-effects. But in order to relate this linear predictor to an outcome in our data D, we need a link function, $g(\cdot)$. In order to ensure the regression model, g has the property

$$g(\mathbb{E}(D|\vec{\beta}, \vec{\gamma})) = \eta,$$

in reference to the conditional expectation of D given our predictors.

In this context, the relevant random variable is a Poisson GLMM, used in instances of count data. Recall the probability mass function of the Poisson distribution,

$$\mathbb{P}(X = k) = \frac{\lambda^k e^{-\lambda}}{k!},$$

with a corresponding link function defined

$$g(\cdot) = \ln(\cdot).$$

With this in place, we can turn to the specific structure of the bumblebee training model. The output Poisson data are dwell-times, or the number of seconds (counts) that bees spend at artificial flowers. The fixed effects are categorical random variables corresponding to various conditions. The random effect accounts for bee-specific randomness (unaccounted for by the other parameters). These fixed and random effects combine via Eq. 2 (below) to generate a rate parameter for a certain bee, which controls the Poisson random variable from which the dwell-time arises. The parameters of this model can be found below in Table 2.2.

Variable	Representation	Effect	Range
nectar chemistry treatment	α	fixed	caffeine, ethanol, control
training color	δ	fixed	white, blue
colony	ω	fixed	1,2,. . . ,10
flower type	ψ	fixed	conditioned, novel
interaction: nectar & flower type	ξ	fixed	NA
individual bee	y	random	bee name tag

Table 2.2: Parameters of interest in the Poisson GLMM.

Thus, we have

$$\begin{gathered} t|\lambda \sim \textsc{Poisson}(\lambda), \quad \lambda = e^{\eta}, \quad \text{and} \\ \eta = \mathbf{X}\vec{\beta} + \mathbf{Y}\gamma, \quad \text{or equivalently} \\ \lambda = \exp(\alpha\beta_1 + \delta\beta_2 + \omega\beta_3 + \psi\beta_4 + \xi\beta_5 + \gamma y). \end{gathered} \tag{Eq. 2}$$

To see this visually, consider the following directed acyclic graph (DAG):

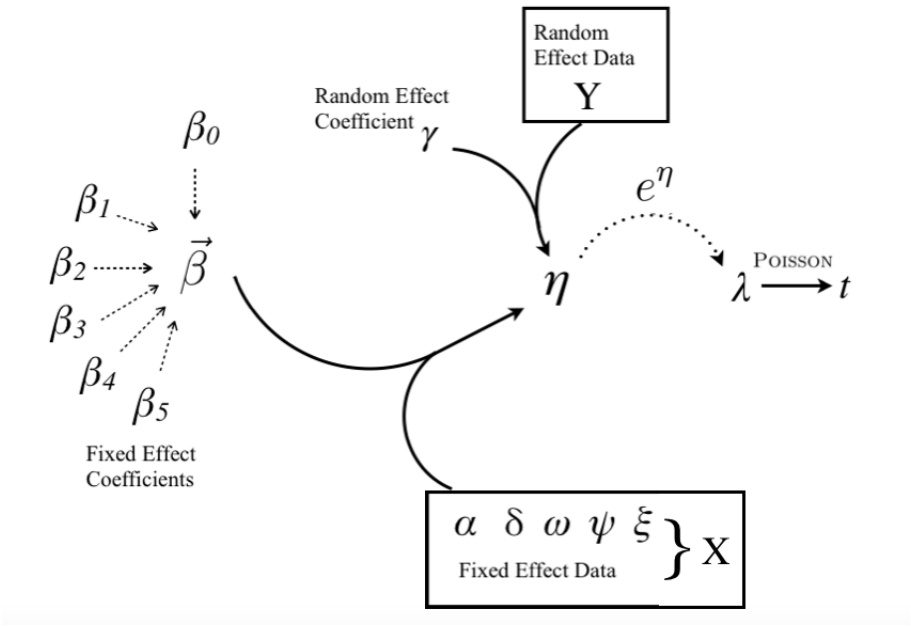


Figure 2.3: Directed acyclic graph of Poisson GLMM. The Poisson rate parameter λ is controlled by a linear combination of the fixed and random effects. The coefficients that govern this combination are thus the objects of parametric inference. The Poisson rate parameters in turn control bumblebee floral dwell-times.

Note the boxed fixed effect data defines the matrix $\mathbf{X}$, while matrix $\mathbf{Y}$ contains the single random effect of the model (bee individual).

2.2.2 Limitations of the standard method

The fundamental research question asks if various training regimens affect the foraging behavior of bumblebees. More specifically, the experimental data are the journeys of each bee—defined by the flowers a foraging bee visits, and the amount of time spent at each site. The explanatory variables of interest are nectar treatment and floral training. However, the relationship is complicated by colony-specific effects, and possible interaction effects. In other words, a mix of random and fixed effects are likely at play. Hence the classical statistics approach would employ a generalized linear-mixed model (GLMM) for inference. More specifically, a Poisson GLMM could be employed, with time being treated as counts.

There are several limitations to this approach. Fundamentally, it reduces the complexity of the data—the bee's journey—to a series of counts. Additionally by using the Poisson specifically, the model treats time as quanta rather than as a continuous variable, and sets the variance equal to the mean. The regression structure specifically demands that each interaction effect and random effect be enumerated *a priori*. Perhaps the most fundamental shortcoming of this approach is its inability to reflect the sophisticated dimensionality of the bees' behavior. More specifically, within the GLMM framework we must choose between

modeling the dwell-times *or* the training states, when instead we ought to model them together in the same model. To this end, I have developed a hierarchical model to answer the questions posed by the GLMM regression, without sacrificing the nuance of the foraging data.

Chapter 3

Statistical Model

To overcome the limitations of the GLMM, our model seeks to directly tie the complexity of foraging behavior to the experimental design. How do we answer the questions posed by the regression: how do the various treatment regimes affect foraging behavior? To this end, I present a hierarchical Bayesian mixture model featuring continuous-time Markov chains. Overall, the complete model consists of three components arranged in a hierarchy, with fully specified probability distributions. At the level of the foraging data, I use a continuous-time Markov chain model to describe the path that the bees take. At the level of the design matrix, I employ a logistic regression model to describe the treatment effects. In between these two components, I apply a mixture model structure to stochastically link the results of the treatment to the Markov chain. This last piece represents what it means for a bee to be successfully trained, or remain untrained. We will now explore each of these pieces of the hierarchy in turn.

3.1 The Journey Model

We begin with two principle assumptions: ***spatial independence***, and ***memorylessness***[1]. Recalling Figure 2.2, the raw data output consists of a series of coordinates describing a bee's location over time. By assuming ***spatial independence***, we assume that the state space can be simplified to $\mathcal{S} = \{B, F, W\}$. That is, a bee must occupy only one of three states: occupying a white flower, occupying a blue tile, or flying between them.

Several factors justify spatial independence. First, the "flowers" are colored tiles that are identical in every way except for color. Furthermore, each tile of the same color is effectively indistinguishable from the other members of its class. However, by the nature of the experiment, one might suggest that the different locations of each tile within the apparatus render every individual tile unique. While this may seem relevant at first glance, the size of the experiment is small enough that we can expect the bees to not factor this spatial information into account. Bumblebees routinely traverse vast distances when foraging, relying on acute eyesight and olfaction to target and visit different nectar sources [12]. As such, any supposed travel-related, visual, or olfactory disparities between tiles can be reasonably ignored, as they likely do not factor into the bees' decision-making processes.

[1] I make a third modeling assumption called ***training***, which is detailed in Chapter 4, section 4.1 Transforming the state space.

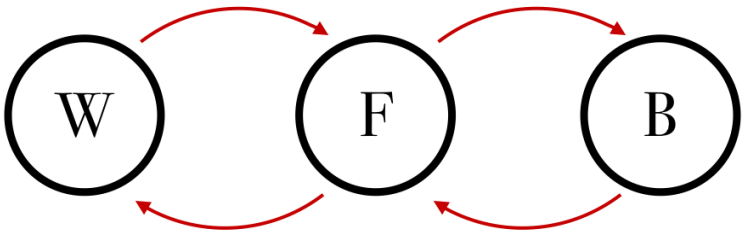


Figure 3.1: State space of foraging journey. By spatial independence, we can condense the spatial information present in the raw data into a series of states $s \in \mathcal{S} = \{W, F, B\}$. Note that, by construction, bees in flight can transition to either flower type, but alighted bees must transition to flying before switching to another flower type.

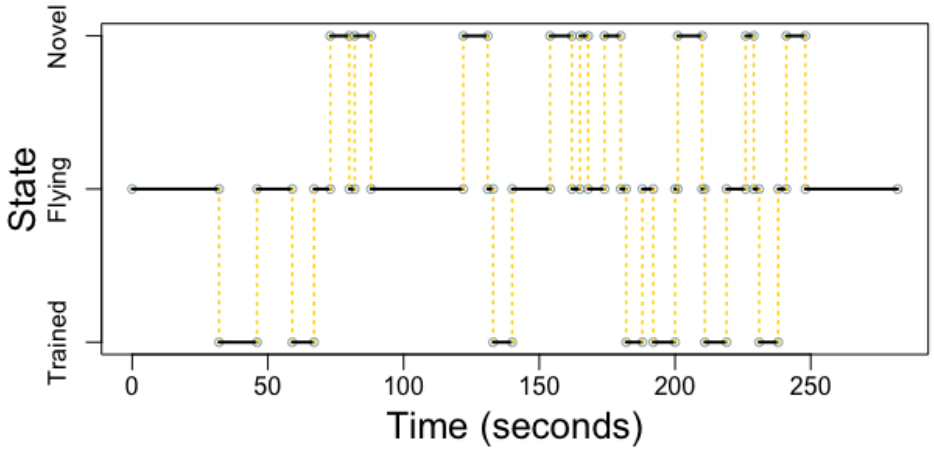


Figure 3.2: Example foraging journey. The spatial independence assumption grants us the ability to condense the raw data information from Figure 2.2 into this elegant flight diagram. Note that the state space has been transformed from blue, white, and flying to trained, untrained, and flying; this will be explored in the next chapter.

By assuming ***memorylessness***, we assume that the system's future behavior does not depend on its past behavior, given the present state (within the same experimental phase). The bees spend certain quantities of time at each location in space: perhaps they spend only a second on a white flower, and then fly around for a minute before settling down on a blue flower for an even longer stay. We are thus supposing that the bees do not change their decision-making processes based on the information gathered during the experimental phase. More precisely, the probability of a bee taking any arbitrary action does not depend on *when*

the bee takes that action. While this assumption has some ecological backing[2], the principle motivation behind this assumption is mathematical. We are assuming that the bees obey the Markov property, which then unlocks the framework of a Markov chain to describe a bee's path data.

3.1.1 Hierarchical model

We begin by visualizing the model as a directed acyclic graph (DAG), and annotating it with the generative model. The subsequent sections of the chapter will focus on explaining this structure and its derivation.

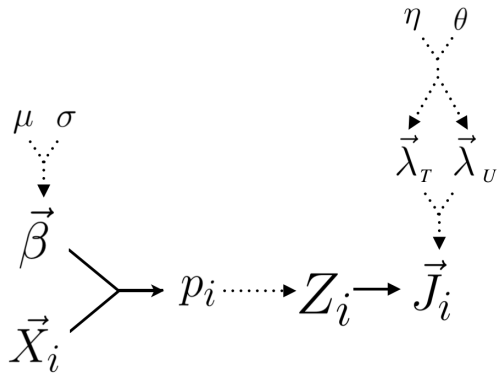


Figure 3.3: DAG of the continuous-time model. Dotted lines indicate stochastic connections, solid lines indicate deterministic connections. See the generative model for the details of the connections.

The prior distributions (located at the top of the DAG) are defined,

$$\vec{\beta} \sim \textsc{Normal}(\mu, \sigma),$$
$$\boldsymbol{Q} \leftrightarrow \vec{\lambda} \sim \textsc{Gamma}(\eta, \theta).$$

The design matrix determines the training state via the logistic,

$$p_i = \text{logit}[\vec{X}_i \cdot \vec{\beta}],$$
$$Z_i \sim \textsc{Bernoulli}(p_i).$$

The training state determines the foraging journey data via a continuous-time Markov chain,

$$\vec{J}_i|\boldsymbol{Q}_{Z_i} \sim \text{CTMC}(\boldsymbol{Q}_{z_i}),$$
$$\pi(\vec{J}_i|\boldsymbol{Q}_{Z_i}) = [-q_{x_n,x_n} \exp(q_{x_n,x_n} \cdot t_n)] \left[\prod_{x_s \in \vec{J}, x_s \neq x_n} \exp(q_{x_s,x_s} \cdot t_s)\right] \left[\prod_{(s,r)\in\mathcal{T}} q_{x_s,x_r}^{c_{x_s,x_r}}\right].$$

Note that $\vec{\lambda}$ and $\boldsymbol{Q}$ are different representations of the same parameters.

[2]Again, the foraging journey in question is rather short relative to a bee's typical expedition [13], and the training phase is 2 days, while the experimental phase is only 5 minutes.

3.1.2 Foraging journey as a continuous-time Markov chain

As shown in the foraging graph (Figure 3.2), each bee i exhibits a *journey* $\vec{J}_i$. Mathematically, the memorylessness assumption enables us to safely assume that a Markov chain representation of the journey is adequate. To construct the statistical model, suppose we have a sequence of categorical random variables X_t that can take a value $s \in \mathcal{S} = \{W, B, F\}$, with t indexing the order of each realized random variable. By memorylessness, we have that

$$\mathbb{P}(X_{t+1} = s_{t+1} \big| X_t = s_t) = \mathbb{P}(X_{t+1} = s_{t+1} \big| X_t = s_t, X_{t-1} = s_{t-1}, \ldots, X_1 = s_1),$$

where $s \in \mathcal{S}$ throughout. In other words, our bees' journeys possess the rather poetic quality wherein their future steps are dependent solely upon their current state, rather than their entire history. Talk about living in the present!

Here we find ourselves at a crossroads: do we treat each step in time the same way, homogenizing the process and making things more accessible? Or do we incorporate the temporal dimension more carefully, complicating the situation? Effectively, we must choose whether to use a discrete- or continuous-time Markov chain to model $\vec{J}_i$. This thesis focuses on the more powerful, descriptive, and complex continuous-time model. I have fully built and explained the discrete-time model, as well; it is attached as its own separate appendix.

We will require another representation of $\vec{J}_i$, namely, the counts matrix, $\boldsymbol{C}_i$. In continuous-time, recall that $\vec{J}_i$ is composed of two vectors, $\vec{t}_i$ and $\vec{x}_i$, containing dwell-time and state information respectively. The counts matrix considers only the transition information from $\vec{x}_i$. For all entries $a \in \vec{x}_i$ except the last, there exists the subsequent entry b. Thus, $a \to b$ describes a single class of transition. The number of times this transition class occurs in $\vec{x}_i$ defines $c_{a,b}$. Repeating this process for all $a, b \in \mathcal{S}$ yields our object $\boldsymbol{C}_i$. As expected, $\boldsymbol{C}_i$ must then be of size $|\mathcal{S}| \times |\mathcal{S}|$:

$$\begin{array}{c} \\ \text{Current state} \end{array} \begin{array}{c} \text{Next state} \\ \begin{bmatrix} c_{T,T} & c_{T,F} & c_{T,N} \\ c_{F,T} & c_{F,F} & c_{F,N} \\ c_{N,T} & c_{N,F} & c_{N,N} \end{bmatrix} \end{array} = \boldsymbol{C}_i.$$

While we've lost the ordering information of these transitions by condensing into the counts matrix, this isn't a true loss given that we assume that the system is memoryless.

3.1.3 Regression effects

As described before, the experiment is a two-factor case-control study investigating the role of training color and training nectar. As such, our experimental inputs consist of the combination between a binary random variable (to what color was the bee trained?), and a ternary random variable (with what nectar was the bee trained?). We can summarize this information in a single object that will later permit us to do regression using these inputs as experimental fixed effects, in much the same way sought by the GLMM approach. For any

given bee i, we can define the vector $\vec{X}_i$ with components representing the bee's treatment regimen: if the bee was trained to blue, the first entry shall be 1; if the bee was trained to ethanol, the second entry shall be 1; if the bee was trained to caffeine, the third entry shall be 1. Notice, all six experimental regimens outlined above now correspond to a unique $\vec{X}_i$ of three dimensions.

Nectar	Trained to Blue	Trained to White
Control	$\vec{X} = \langle 1, 0, 0 \rangle$	$\vec{X} = \langle 0, 0, 0 \rangle$
Ethanol	$\vec{X} = \langle 1, 1, 0 \rangle$	$\vec{X} = \langle 0, 1, 0 \rangle$
Caffeine	$\vec{X} = \langle 1, 0, 1 \rangle$	$\vec{X} = \langle 0, 0, 1 \rangle$

Table 3.1: Key to design matrix of colony treatments

Lastly, we can summarize this information for an entire data set by combining $\vec{X}_i$ for all N bees into a single $N \times 3$ matrix $\mathbf{X}$, whose i^{th} row corresponds to the i^{th} bee.

3.1.4 Training data: the missing link

So far, we've described our experimental influences on a bee as a vector that can be used in a later regression, and a bee's particular foraging journey as a Markov chain. Our model is then tasked with linking these two pieces of information together. We'll do this by invoking a piece of hidden, underlying data that's remarkably present within our entire data set. Consider the following two path diagrams for two distinct bees within the same colony.

While one bee overwhelmingly prefers the color flower to which it has been trained, the other bee shows no obvious preference. One might say that one bee has been successfully trained, while the other remains untrained. In fact, the large majority of the 156 bees present in the data set can be classified as trained or untrained by simple inspection! Some are ambiguous (at least to our eyes), and two are seemingly "anti-trained," meaning they strictly prefer the novel flower color, as opposed to the trained color. (While we initially built our model to include the anti-trained possibility, we have insufficient data to truly explore it.)

Our goal is to understand how the design of the experiments controls the frequency of whether or not a given bee is successfully trained (excluding both anti-trained individuals from the analysis, as they are distinct from untrained individuals). Here we employ a mixture model structure—a mixture of individuals in trained and untrained states. To model an individual's state, define a random variable Z_i pertaining to the ith bee whose value equals one if the bee has been successfully trained, and zero if not. Notice, we are assuming that training is a dichotomy rather than a spectrum; inspecting the data set gives credence to this assumption (most bees are not ambiguous). Furthermore, the bulk of the ecological analysis remains interested in the regression effects, rather than modelling partial training. Each bee now has its own Markov chain, hidden training state, and regression vector. In the next section, we'll outline the mathematical connections between these three objects, which constitutes our statistical model.

Figure 3.4: Trained versus untrained bees from Colony B. Represented above are two distinct journey graphs corresponding to distinct individuals from the same colony—meaning they underwent the same treatment. The left graph depicts a bee choosing to forage on the type of flower to which it was trained. The right graph depicts a bee showing no such preference. Clearly these are fundamentally different behaviors arising from identical conditions. (See Chapter 4.1 for how this relates to the training assumption.)

3.2 Continuous-time model

The continuous-time model (CTM) is the same as the discrete-time model (DTM, see appendix), except in how we view the path data $\vec{J}_i$. While the DTM effectively forces a transition every second (or some arbitrary, constant unit), the CTM considers $\vec{J}_i$ as the composition of both dwell-times $t_s \in (0, \infty)$ and states $x(t_s) \in \mathcal{S}$,

$$\vec{J}_i = (x_1, t_1, x_2, t_2, \ldots, x_n, t_n),$$

for n sequential states. Note, $x_1 = F$ for all i because the bees are released into the chamber while flying; thus, $\mathbb{P}(\vec{J}_i(0) = F) = 1$ is our starting condition.

Because we treat $\vec{J}_i$ as arising from a continuous-time Markov process, we now specify the parameters of that process. As discussed in the first chapter, CTMCs are sufficiently specified by a state space, a starting distribution, and a rate (or generator) matrix $\boldsymbol{Q}$. We can construct the rate matrix for this process as

$$\begin{array}{cc} & \\ \textit{from} & \end{array} \begin{array}{c} \\ \begin{matrix} \text{W} & \text{F} & \text{B} \end{matrix} \end{array} \quad \begin{array}{c} \textit{to} \\ \begin{matrix} \text{W} \\ \text{F} \\ \text{B} \end{matrix} \end{array} \begin{bmatrix} -\lambda_1 & \lambda_1 & 0 \\ \lambda_2 & -(\lambda_2 + \lambda_3) & \lambda_3 \\ 0 & \lambda_4 & -\lambda_4 \end{bmatrix} = \boldsymbol{Q}.$$

These matrices define what it means for a bee to be trained or untrained. We have thus identified our parameters of interest, $\vec{\lambda}_U$ and $\vec{\lambda}_T$, two vectors of four components each.

3.2.1 CTMC likelihood

We can now begin constructing our likelihood using the CTMC model. To this end, we'll use a common understanding of CTMCs, which is to consider the process as a contest between various alarm clocks. Suppose we enter a new state in the chain, x_s. Instantly, we then set alarm clocks at all possible novel states $x_{s+1} \neq x_s$, which are the candidates for our next state. These clocks detonate at a time that is exponentially distributed according to rate $q_{x_s,x_{s+1}}$, the corresponding entry in the rate matrix. Once they do, we transition to x_{s+1} immediately. Our likelihood must reflect this process.

To build it up, we'll first break it down into two tasks. First, we must dwell in state x_s for time t_s. Then, we must transition to state x_{s+1}.

Consider the first task. In order to dwell in state x_s for t_s, the alarm clock corresponding to x_{s+1} must detonate at t_s, *and* no other alarm clocks must detonate before it. In other words, the minimum of all candidate dwell-times must be t_s. Fortunately, Theorem 3.1 (see appendix) grants us the ability to represent this with a single probability density function, an exponential whose rate is equal to the sum of the rates of the candidate alarm clocks. Recall that for $\boldsymbol{Q}$ we have the diagonal entries equal to the opposite of the sum of the off-diagonal entries:

$$-q_{s,s} = \sum_{s \neq r} q_{s,r}.$$

Thus, the rate parameter for the minimum of the competing exponentials can be found by the opposite of the corresponding diagonal entry. Equivalently,

$$\pi(t_s \text{ is the minimum dwell-time}) = -q_{x_s,x_s} \exp[q_{x_s,x_s} t_s],$$

which, again, is just the density at time $= t_s$ of an exponential random variable with rate equal to $-q_{x_s,x_s}$. Finally, because each dwell-time is independent of the rest, we can link these by the basic principle of counting. Indexing according to sequential states, we find

$$\pi(\vec{t}|\boldsymbol{Q}) = \prod_{x_s \in \vec{J}_i} -q_{x_s,x_s} \exp[q_{x_s,x_s} t_s].$$

Now for the second task, we're to consider transitions. Define $\mathcal{T}$ as the set of all viable transitions from one state to the next. By the Markov property, we can ignore the memory of the system, and simply cluster all transitions of the same class together. That is, we can condense the sequential state information of $\vec{J}_i$ into a matrix of counts $\boldsymbol{C}_i$ accounting for each time a class of transition occurs. (Note that this counts matrix appears in the DTM; however, in the DTM it counted far more empty transitions than true transitions, which is not the case for the CTM.) We then must calculate the probability of each class of transition occurring, which amounts to the probability of the subsequent state's alarm clock detonating

first. From our corollary to Theorem 3.1, we have this quantity: the probability that the alarm clock at state r detonates first is simply the rate of the r^{th} alarm clock normalized by the sum of the rates of the contending alarm clocks. In terms of $\boldsymbol{Q}$, this is

$$\mathbb{P}(s \to r) = \frac{q_{x_s,x_r}}{-q_{x_s,x_s}}.$$

We can now construct the entire likelihood:

$$\begin{aligned}
\pi(\vec{J}|\boldsymbol{Q}) &= \left[\prod_{x_s \in \vec{J}} -q_{x_s,x_s} \exp(q_{x_s,x_s} \cdot t_s)\right] \left[\prod_{(s,r)\in\mathcal{T}} \left(\frac{q_{x_s,x_r}}{-q_{x_s,x_s}}\right)^{c_{x_s,x_r}}\right] \\
&= \left[(-q_{x_n,x_n} \exp(q_{x_n,x_n} \cdot t_n)) \left(\prod_{x_s \in \vec{J}, s\neq n} \exp(q_{x_s,x_s} \cdot t_s)\right) \left(\prod_{(s,r)\in\mathcal{T}} -q_{x_s,x_s}^{c_{x_s,x_r}}\right)\right] \\
&\quad \cdot \left[\prod_{(s,r)\in\mathcal{T}} \left(\frac{q_{x_s,x_r}}{-q_{x_s,x_s}}\right)^{c_{x_s,x_r}}\right] \\
&= \underbrace{[-q_{x_n,x_n} \exp(q_{x_n,x_n} \cdot t_n)]}_{\text{final dwell-time}} \underbrace{\left[\prod_{x_s \in \vec{J}, s\neq n} \exp(q_{x_s,x_s} \cdot t_s)\right]}_{\text{bulk dwell-times}} \underbrace{\left[\prod_{(s,r)\in\mathcal{T}} q_{x_s,x_r}^{c_{x_s,x_r}}\right]}_{\text{transitions}}.
\end{aligned}$$

There are a couple nuances here that merit further remarks. First, we set the probability of the initial distribution to one, since the bees always begin in the flying state. Second, the re-indexing step in the second line allows for a simpler representation for the bulk of the data, but requires that we treat the final dwell-time differently. This edge case lacks a transition (because the experiment ends), and is better considered on its own (see the leftmost factor in the final line).

Chapter 4

Inference

The following algorithm contains various flavors of the Metropolis-Hastings Algorithm, as per my discussion in Chapter 1. According to the model, the regression and journey process parameters are always conditionally independent of each other, given the training state. This allows us to completely ignore the regression parameters when updating the journey process parameters, provided we have the training information (and vice-versa). Furthermore, the training posteriors are automatically defined when we condition upon both the journey process and regression parameters. So, while the training state might be a hidden variable, it makes sampling much more accessible.

Note that all inference was accomplished using the open source statistical programming language R [14]. Many plots are made possible by the ggplot2 environment from Hadley Wickham's tidyverse [15].

4.1 Transforming the state space

When coding the algorithm, we have to be slightly more specific in what we mean by "trained." Recall that all of our state spaces thus far have been explicitly defined as $\{W, F, B\}$, representing the two flower types and flight. Recall further that each bee was exposed to *either* blue *or* white in the training phase. As such, we want our Markov chains to actually model the *training* states, rather than what flower the bee is on, because that's what the rest of the model seeks to explain. To that end, we perform a mapping from $\mathcal{S} \in \{B, F, W\}$ to $\mathcal{S}' \in \{T, F, N\}$, before running the algorithm. These states are *trained* (the flower color of the training phase), *flying*, or *novel* (the opposite flower color of the training phase). (We could also refer to the novel state as the "antitrained" state, in honor of the two renegade bees that actively disobeyed their training.) This allows us to format all Markov chains (and thus all transition, Dirichlet, and rate matrices, in both discrete- and continuous-time) such that each state is assigned a sequential index ($T = 1, F = 2, N = 3$). I refer to this mapping as the third assumption of the model, the ***training*** assumption, and puts figure 3.4 in better context.

4.2 Continuous-Time Markov chain inference

The algorithm is broken down into update routines that attempt to sample from the conditional posterior of each variable class. Careful tempering and ordering aid convergence.

4.2.1 Move 1: Metropolis-Hastings sampling of regression coefficients

When it comes to updating the regression coefficients $\vec{\beta}$, no conjugacy comes readily to mind. As such, we must recourse to the computationally slower (but still revolutionary!) Metropolis-Metropolis-Hastings algorithm of 1970. Again, our goal is to sample from the marginal posterior distribution of $(\vec{\beta}|\mathbf{X}, \vec{\lambda}, \vec{Z})$, which can be simplified to $(\vec{\beta}|\mathbf{X}, \vec{Z})$ by the aforementioned conditional independence structure. Recalling the key to the design matrix from Chapter 2, there are three weight parameters and one intercept parameter that need to be sampled (two binary, one ternary variable in this logistic regression).

Recall the explanation of the MHA in the first chapter. At the heart of the MHA we have the MH ratio, used in the expression

$$A = \min\left(1, \frac{\pi^*(x'_{t+1})Q(x_t|x'_{t+1})}{\pi^*(x_t)Q(x'_{t+1}|x_t)}\right),$$

where π^* refers to a target distribution, and Q refers to the proposal distribution (sometimes referred to as a transition kernel). Additionally, note that x'_{t+1} has merely been proposed from Q, rather than being installed in the chain (hence the prime demarcation). In this case, our target distribution is the posterior, and our proposal distribution is a normal distribution, seeing as the parameters of interest are real numbers. For example, $Q(x_t|x'_{t+1})$ takes the density of a normal distribution with mean x_{t+1} and standard deviation equal to our predetermined walk parameter. To represent the target posterior distribution, we apply Bayes's Law,

$$\begin{aligned}
\frac{\pi^*(x'_{t+1})}{\pi^*(x_t)} &= \frac{\pi(\vec{\beta}'_{t+1}|\mathbf{X}, \vec{Z})}{\pi(\vec{\beta}_t|\mathbf{X}, \vec{Z})} \\
&= \frac{\pi(\vec{Z}|\vec{\beta}'_{t+1}, \mathbf{X})\pi(\vec{\beta}'_{t+1})}{\pi(\mathbf{X})} \div \frac{\pi(\vec{Z}|\vec{\beta}_t, \mathbf{X})\pi(\vec{\beta}_t)}{\pi(\mathbf{X})} \\
&= \frac{\pi(\vec{Z}|\vec{\beta}'_{t+1}, \mathbf{X})\pi(\vec{\beta}'_{t+1})}{\pi(\vec{Z}|\vec{\beta}_t, \mathbf{X})\pi(\vec{\beta}_t)} \\
&= \frac{\left[\prod_i \text{logit}(\vec{\beta}'_{t+1} \cdot \mathbf{X})^{Z_i}(1 - \text{logit}(\vec{\beta}'_{t+1} \cdot \mathbf{X}))^{1-Z_1}\right]\left[\prod_k \frac{\exp[-(\beta'_{k,t+1}-\mu_k)^2/(2\sigma_k^2)]}{\sigma_k\sqrt{2\pi}}\right]}{\left[\prod_i \text{logit}(\vec{\beta}'_t \cdot \mathbf{X})^{Z_i}(1 - \text{logit}(\vec{\beta}'_t \cdot \mathbf{X}))^{1-Z_1}\right]\left[\prod_k \frac{\exp[-(\beta'_{k,t}-\mu_k)^2/(2\sigma_k^2)]}{\sigma_k\sqrt{2\pi}}\right]},
\end{aligned}$$

where i indexes each bee, and k each $\vec{\beta}$ component. Note, the two prior distributions ($\pi(\vec{\beta})$) simply obey the density of a normal distribution, with predetermined mean μ_k and standard deviation σ_k defined for each component of $\vec{\beta}$. The likelihood refers to i independent Bernoulli outcomes weighted according to $p_i = \text{logit}(\vec{X}_i \cdot \vec{\beta})$. Clearly, this ratio is readily computable when conditioned upon $\vec{Z}$.

With this ratio calculated, we accept the proposal state $\vec{\beta}'_{t+1}$ as $\vec{\beta}_{t+1}$ with probability A. (Note, the minimum statement simply prevents the probability from exceeding unity.)

4.2.2 Move 2: Metropolis-Hastings sampling of rate matrix parameters

Recall from the hierarchical model that each of the two rate matrices is defined by four rate parameters, denoted $\vec{\lambda}_{Z_i} \in \{\vec{\lambda}_T, \vec{\lambda}_U\}$ with $\vec{\lambda} \in \mathbb{R}^4_{>0}$. We then have eight parameters to find, with no conjugacy relationship to aid our search. As such, we recourse to Metropolis-Hastings, using our likelihood from the previous chapter, and $\textsc{Gamma}$(shape = 1.5, rate = 1.5) distributions for our priors on each. (Note, the prior for the real data runs is more nuanced, as I discuss in the next section.)

For the proposal distribution, we have to be careful that the proposed parameters do not become negative (the domain of an exponential random variable is strictly positive). Instead of using a normally distributed random walk with the old parameter set as the mean (as is done in the discrete-time case), we'll use a proportionality-based variant that avoids proposing non-positive parameters. We draw a scale factor R from

$$R \sim \textsc{Uniform}\left(\frac{x}{1+x}, \frac{1+x}{x}\right),$$

and multiply it by the current λ to yield the proposed update. The key here is that the Hastings ratio is symmetric; in other words, the probability of proposing state Θ_i from Θ_j doesn't change if we swap i and j. We found that $x = 3$ aids convergence.

4.2.3 Move 3: Training state posterior calculation

In the case where we hold the rate matrix and regression parameters constant, we can sample from the training state posterior directly. First, recall that because $Z_i \sim \textsc{Bernoulli}(p_i)$, there are only two possible outcomes of Z_i, denoted $m \in \{\text{Trained}, \text{Untrained}\}$. As such, the Law of Total Probability enables direct posterior sampling: we know the likelihood and prior of both possible states, which defines an unnormalized posterior (it lacks the marginal, which is the normalizing constant from Bayes's Law). Then, by dividing through with their aggregate, we can find the posterior up to a constant.

To that end, we must consider the prior and likelihood. The prior is simply the underlying probability of training given the treatment regimen. The likelihood is the probability of the journey data occurring if it is distributed as a continuous-time Markov chain with rate matrix $\boldsymbol{Q}$. (For ease of reading, we group $\vec{J}_i$, $\boldsymbol{C}_i$, and $\vec{X}_i$ as data D; and we use $\boldsymbol{Q}$ to represent both the trained and untrained matrices). We can see all this mathematically as

$$\begin{aligned}\underbrace{\pi(Z_i = m | D, \vec{\beta}, \boldsymbol{Q})}_{\text{posterior}} &\propto \sum_{\forall m} \left(\underbrace{\pi(\vec{J}_i | \boldsymbol{Q}_m)}_{\text{likelihood}} \cdot \underbrace{\pi(Z_i = m | \vec{X}_i, \vec{\beta})}_{\text{prior}} \right) \\ &\propto \mathcal{L}(\vec{J}_i | \boldsymbol{Q}_T) \cdot \left(\underbrace{\text{logit}(\vec{X}_i \cdot \vec{\beta})}_{p_i} \right) + \mathcal{L}(\vec{J}_i | \boldsymbol{Q}_U) \cdot \left(\underbrace{1 - \text{logit}(\vec{X}_i \cdot \vec{\beta})}_{1-p_i} \right),\end{aligned}$$

where the likelihood, as derived in Chapter 3, is

$$\mathcal{L}(\vec{J}_i|\boldsymbol{Q}) = \left[\prod_{s\in\vec{J}} -q_{s,s}\exp(q_{s,s}\cdot t_s)\right]\left[\prod_{(s,r)\in\mathcal{T}}\left(\frac{q_{s,r}}{-q_{s,s}}\right)^{c_{s,r}}\right],$$

where we simplify the notation by replacing x_s with just s.

Because we've defined all states that the unnormalized posterior can take, we can normalize these states by dividing through with their sum. Thus, for an example of one of the posterior probabilities, consider the probability of being trained:

$$\begin{aligned}
\pi(Z_i = T|D,\vec{\beta},\boldsymbol{Q}) &= \frac{\mathcal{L}(\vec{J}_i|\boldsymbol{Q}_T)\cdot\left(\text{logit}(\vec{X}_i\cdot\vec{\beta})\right)}{\left[\mathcal{L}(\vec{J}_i|\boldsymbol{Q}_T)\cdot\left(\text{logit}(\vec{X}_i\cdot\vec{\beta})\right)\right] + \left[\mathcal{L}(\vec{J}_i|\boldsymbol{Q}_U)\cdot\left(1-\text{logit}(\vec{X}_i\cdot\vec{\beta})\right)\right]} \\
&= \left[\prod_{s\in\vec{J}} -q_{T,s,s}\exp(q_{T,s,s}\cdot t_s)\right]\left[\prod_{(s,r)\in\mathcal{T}}\left(\frac{q_{T,s,r}}{-q_{T,s,s}}\right)^{c_{s,r}}\right]\cdot\left(\text{logit}(\vec{X}_i\cdot\vec{\beta})\right) \\
&\div \left(\left[\prod_{s\in\vec{J}} -q_{T,s,s}\exp(q_{T,s,s}\cdot t_s)\right]\left[\prod_{(s,r)\in\mathcal{T}}\left(\frac{q_{T,s,r}}{-q_{T,s,s}}\right)^{c_{s,r}}\right]\cdot\left(\text{logit}(\vec{X}_i\cdot\vec{\beta})\right)\right. \\
&\quad \left. + \left[\prod_{s\in\vec{J}} -q_{U,s,s}\exp(q_{U,s,s}\cdot t_s)\right]\left[\prod_{(s,r)\in\mathcal{T}}\left(\frac{q_{U,s,r}}{-q_{U,s,s}}\right)^{c_{s,r}}\right]\cdot\left(1-\text{logit}(\vec{X}_i\cdot\vec{\beta})\right)\right).
\end{aligned}$$

We've thus fully specified the posterior distribution for the training state. By evaluating this expression, we find the posterior probabilities of training versus untraining, which then makes updating $\vec{Z}$ equivalent to a series of weighted coin tosses.

4.2.4 Compound CTM Algorithm

Each of the three classes of moves requires that we condition upon one of the three major variables that we seek to update. As such, the algorithm proceeds in a step-wise manner, doing exactly one of the moves for each next step. The order and frequency of these moves is up to us. This—along with tuning parameters like priors and proposal dispersion—is where science becomes art.

To set up the algorithm, we must define all parameters for each prior distribution. For the regression coefficients, $\vec{\beta}\in\mathbb{R}^4$. These can, in theory, be any real number, but the logistic likelihood becomes zero for many combinations; after some experimentation, we used normal distributions with mean = 0 and standard deviation = 8 for all four components. The rate matrix parameters $\vec{\lambda}$ must be greater than zero for the exponential to be defined, that is $\lambda\in\mathbb{R}_{>0}$. We employ the disperse $\textsc{Gamma}(1.5, 1.5)$ generally, but take a more nuanced approach for the real data.

For the real data, I have crafted a more involved, data-related prior structure. The CTM has trouble converging to a sensibly separated mixture on its own. In other words, it frequently collapses into a state where all bees are identically trained, or eventually sticks to a local mode wherein the majority of bees are not sensibly classified. To combat these problems, I reflected on what prior information we have that wasn't being included in the model.

By using a process known as *tempering the chain*, we can hone the parameters to a somewhat reasonable place very rapidly, before letting the chain freely update. We do this by ignoring one of the three parameter classes. By ignoring the regression coefficients, we can ask the $\boldsymbol{Q}$ and $\vec{Z}$ parameters to come to some reasonable agreement independently of the regression; similarly, by ignoring the rate matrices, we can ask the $\vec{\beta}$ and $\vec{Z}$ parameters to come to some sort of agreement. Then, when we run the chain with these parameters together, each parameter will have a far more reasonable starting position, which cuts down on computation time significantly.

According to the exploratory data analysis, most bees clearly occupy either a trained or untrained state, which we can identify graphically. If we could represent those states in terms of CTMC rate matrices, we could inform our prior distribution on $\vec{\lambda}$ accordingly. To that end, I selected two sets of bees (that I'll call training set candidates) that appeared, by eye, to be unequivocally either trained or untrained. I further limited the selection to bees that performed at least 20 transitions (relatively high data quality). Of the 154 bees in total, 22 trained and 40 untrained bees met this criteria. I then ran 10 parallel chains in the following fashion: 10 bees from each training set were randomly sampled and treated as fixed. Then, 2,000 rate matrix MH moves were run, and the last quarter were stored. Looking over the ten runs, the convergence was consistent enough (even across different training sets!) to use these matrices as the priors. Effectively, these values informed the shape and rate parameters of our prior gamma distribution (details in the appendix for Chapter 6). This obviated the need for a $\boldsymbol{Q}$-$\vec{Z}$ tempering process and label switching check[1] in the full, real data runs. Note, the $\vec{\beta} - \vec{Z}$ pair was still tempered.

Because of the prior search procedure, the regression coefficients are updated three times as frequently as the other two parameter classes, with $\vec{\lambda}$ being updated 2.5 times more frequently than $\vec{Z}$ (due to the variability in a random-walk MH).

4.3 Convergence

Each run consisted of two thousand tempering iterations, followed by 200,000 unconstrained iterations. The algorithm recorded one out of every 500 states to increase storage efficiency and reduce autocorrelation (see plots below). I ran ten chains in parallel for an effective sample size of roughly 8,000 per $\vec{\beta}$ component (with similarly large values for the other parameters). Across all chains, I found specific convergence to a single mode, with the median posterior samples for each independent runs being similar.

[1] See details of the discrete-time inference algorithm for a discussion of the label-switching problem [16].

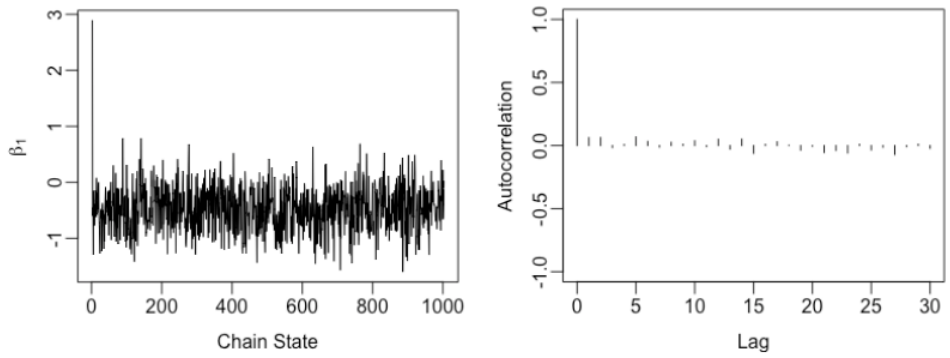


Figure 4.1: Here we see the traceplot (left) and autocorrelation plot (right) for an example $\vec{\beta}$'s components in one run. The traceplot demonstrates the typical wire-brush pattern characteristic of convergence (around approximately -0.5) [17]. The autocorrelation hovers around zero, suggesting that we eventually sample the posterior distribution without bias from early samples. These figures are representative of the real data runs.

Chapter 5

Simulation Studies

Before we apply the model to the experimental data, we want to have some understanding of and confidence in its ability to infer a variety of hypothetical parameters given the size of our datasets. To accomplish this, we employ simulation studies, where we simulate data according to the hierarchical model with known parameters of our own choosing, and then attempt to retrieve those parameters using the model. Obviously, we will not tell the model the true parameters; instead, we will assume knowledge of only priors, and use the inference algorithm to find the values.

Since we aim to generate hypothetical data according to the model, we need to define the parameters governing the underlying process. Recall that the regression parameters control the training probabilities as a function of the experimental design, and the rate matrix parameters control the journeys. With this in mind, consider the following simulation routine:

1. Define an underlying regression vector that, through experimentation, yields stable inference:

$\beta_{\text{intercept}}$	β_{blue}	β_{EtOH}	β_{caff}
-0.74	1.5	-1.5	0.02

Table 5.1: Hypothetical regression vector for simulation studies

2. Assign randomly generated regression effects $\vec{X}$ to each of the N bees. We now have a fully defined p_i for each bee i, as $p_i = \text{logit}(\vec{\beta} \cdot \vec{X}_i)$.

3. Simulate one Bernoulli random variable weighted p_i for each of the N bees: this amounts to assigning the bees their underlying training status.

4. Define what those training statuses imply by specifying two rate matrices, $\boldsymbol{Q}_U$ and $\boldsymbol{Q}_T$.

5. For each bee, simulate a continuous-time Markov chain journey according to $\boldsymbol{Q}_{Z_i}$ (the relevant rate matrix, as determined by the training status). These journeys are (at least) t seconds long[1].

The algorithm receives only the simulated $\vec{J}_i$ and $\vec{X}_i$ information (reflecting realistic conditions), and is tasked to infer $\boldsymbol{Q}$, $\vec{\beta}$, and $\vec{Z}$.

[1]The CTMC simulation routine requires a minimum total flight duration. When adding a dwell-time would exceed this threshold, the time and subsequent transition are still recorded, but the journey ceases.

In order to assess the inference to come, we need to define some performance metrics. Recall the three classes of parameters: $\vec{Z}$, $\vec{\lambda}$, and $\vec{\beta}$. To assess the performance of $\vec{Z}$, we define the somewhat endearing statistic "fraction bad bees" (FBB), which is simply the proportion of bees whose training state the algorithm incorrectly classifies. Thus, a value approaching zero suggests ideal performance. For the $\vec{\beta}$ and $\vec{\lambda}$ values, we simply calculate the Euclidean distance (EUD) between the simulated and inferred vectors. As this is a distance metric [18], the lesser values correspond to better performance.

Next, we aim to select an array of parameters that adequately probes our model. Ideally, we'd not only investigate the region that our experimental data occupy, but the boundaries as well. Where does the algorithm perform well, and where does it collapse? For full details of the parameters chosen, see the appendix of Chapter 5. Each chain (with its specific values for $\vec{\beta}$, $\boldsymbol{Q}$, N, and t) was run 10 times for 200,000 iterations each.

We begin by considering the performance of the algorithm as a function of sample size. Consider the following representative results, where we track inference performance as a function of number of bees and number of seconds.

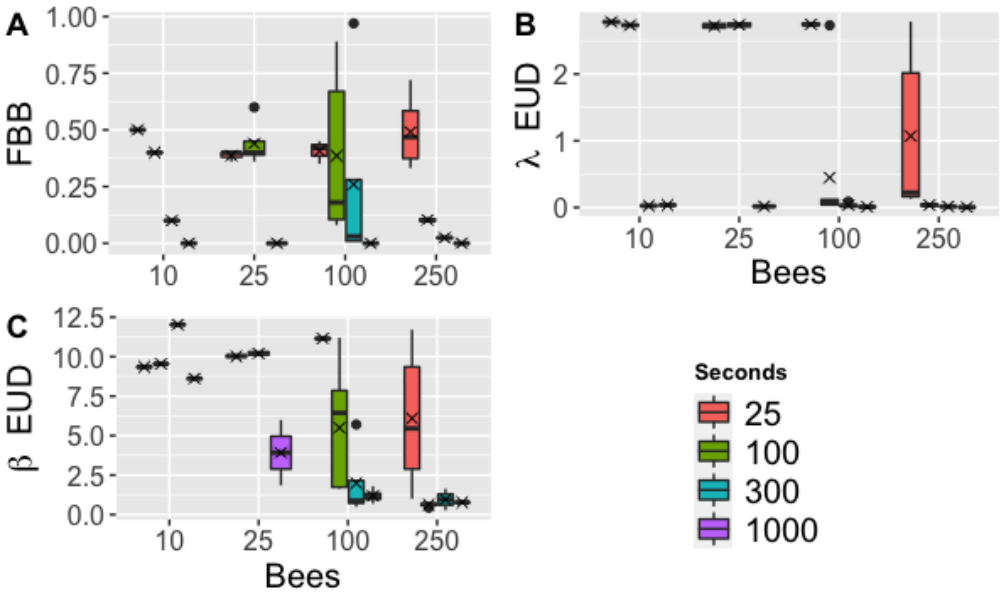


Figure 5.1: Inference performance and convergence increase with two different types of sample size (flight durations, and number of flights). The box plots show 160 runs of 200,000 iterations, with identical $\vec{\beta}$ and $\boldsymbol{Q}$ parameters. These runs differ only in sample size; each run has a specific number of bees, all of which have a specific minimum journey time associated with them. Convergence can be observed as the width of the box plot whiskers decreases, indicating that the various runs agree on the parameter values. Inference performance can be observed as the values approach zero, per the above metric discussion. Within the same number of bees, adding flight duration increases performance; within the same flight duration, adding bees increases performance. Indeed, sample size is comprised of both number of bees and flight duration.

As desired, the convergence and performance improve as sample size increases. Convergence indicates that our runs behave reasonably, and can be observed as the range of distances diminishes. Note that the regression inference appears to require the most information, and has the poorest overall performance of the three. We might expect this for two reasons: the logistic likelihood can be fairly narrow and difficult to sample [19]; and any uncertainty from the journey and training information ripples up toward the logistic. If the job of the logistic is to map the design matrix to the $\vec{Z}$ information, then any ambiguity in the $\vec{Z}$ information will directly translate to the $\vec{\beta}$. Similarly, any ambiguity in the $\vec{\lambda}$ information directly translates to $\vec{Z}$, which in turn translates to $\vec{\beta}$ once more. This helps explain why wherever the $\vec{Z}$ and $\vec{\lambda}$ struggle, the $\vec{\beta}$ necessarily struggles, too.

Let us investigate some of the features of these $\vec{\lambda}$ on our algorithm's inference. The critical distinction between the trained and untrained rate matrices lives in the second row: the exponential parameters that govern both (a) the *probabilities* of transitioning to either the trained or novel state; and (b) the *frequency* of those transitions. Consider the role of transition frequency. One might surmise that high transition frequency implies higher data, and consequently more efficient inference—meaning greater precision and accuracy in convergence. The inference of two rather similar, high transition frequency rate matrices is demonstrated below.

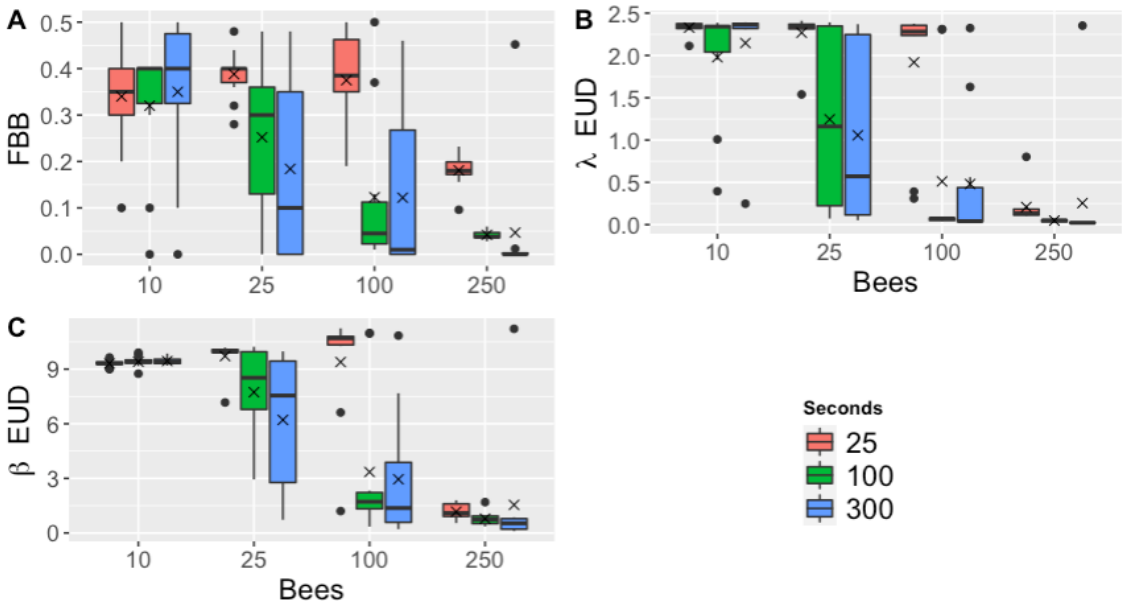


Figure 5.2: High transition frequency aids performance. As sample size increases, the convergence of the runs increases the IQRs and whiskers of the box plots diminish. Similarly, increased sample sizes lead to lower distance values, suggesting more accurate performance.

We find the same result as before, now with incremental increases in data greatly aiding inference and convergence. But what if we have lower transition frequency rate matrices governing the process?

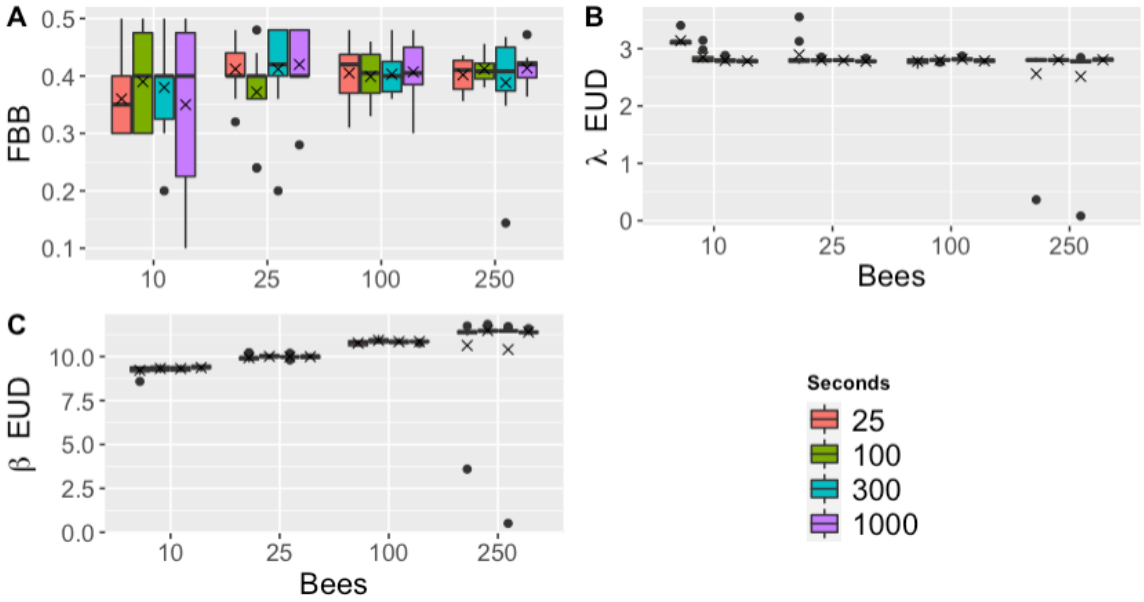


Figure 5.3: Low transition frequency results in tight convergence, but terrible inference accuracy. The patterns of the previous figures do not persist here: augmenting sample size does not aid inference performance, and only marginally increases convergence. This indicates that the algorithm requires some baseline of transition frequency to perform successful inference.

Interestingly, the inference appears to not benefit noticeably from increased sample size. Note that transition probability has not changed: only transition frequency. That is, when transitions do occur, they favor one state over the other in the same way that the high transition frequency study does above. This emphasizes the marked importance of transition frequency to inference accuracy.

On that note, we ought to consider the transition probabilities separately from transition frequency. In other words, what if two rate matrices differ appreciably (or negligibly) in terms of transition probabilities? That is, what if trained bees behave drastically different from untrained bees?

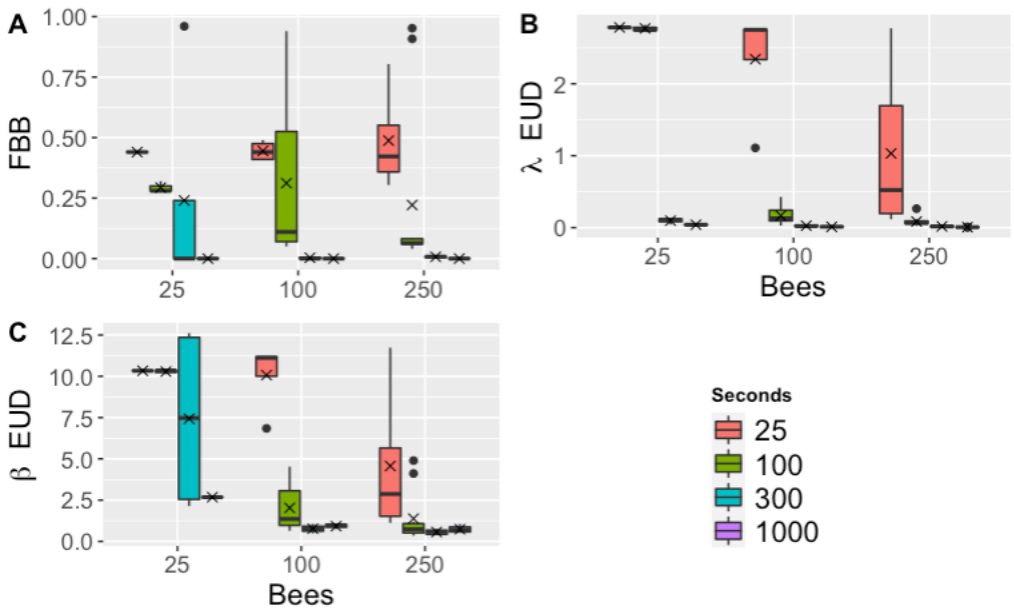


Figure 5.4: Distinct transition probabilities result in easier convergence. The two rate matrices of interest differ severely in transition probability, but nothing else. The expected convergence pattern emerges once again: as the sample size increases, the algorithm performs very accurately and precisely. Clearly, rate matrix distinction aids convergence.

We see our expected pattern emerge once again. Intuitively, if the mixtures are drastically different, the algorithm can parse the mixture very readily. And if we lower the distinction between transition probabilities, while preserving the same transition frequency, we find the expected collapse in inference.

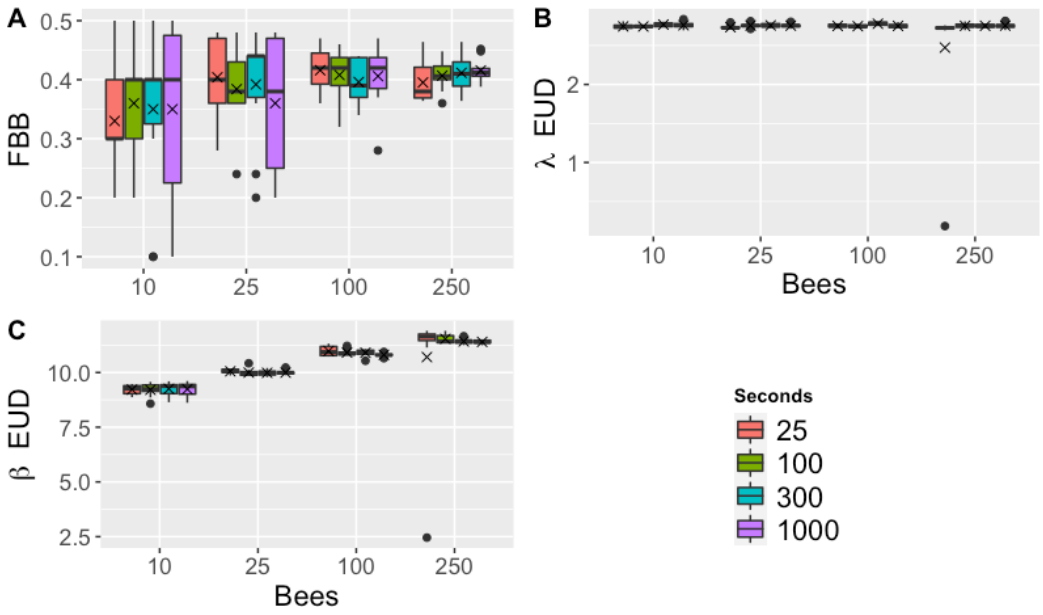


Figure 5.5: Homogeneous transition probabilities result in weaker convergence. The two rate matrices differ very slightly in transition probability, and are identical in other respects. The algorithm fails to converge to any reasonable parametric estimation. Clearly, rate matrix similarity harms convergence.

Fortunately, the real data demonstrate a clearly marked difference in transition probabilities between the two states. The transition frequency is more variable across the data sets, but anything above 20 transitions (roughly 40% of the data set) we've found to be quite high quality. We will keep the importance of that critical second row of the rate matrices in mind as we move into the real data.

Chapter 6

Data Analysis

With the encouraging results from the simulation studies in-hand, we proceed to applying the algorithm to real data. That is, can we infer the underlying parameters of interest governing the foraging trials conducted in the lab?

6.1 Regression and training results

Posterior Results		
Parameters	90% credible interval	median
β Intercept	-3.1 to -1.2	-2.0
β Blue?	1.6 to 3.5	2.4
β EtOH?	-1.7 to -0.079	-0.88
β Caff?	-0.80 to 0.080	-0.011
Untrained λ_1	0.107 to 0.121	0.114
Untrained λ_2	0.0636 to 0.0740	0.0691
Untrained λ_3	0.0422 to 0.0504	0.0465
Untrained λ_4	0.0998 to 0.1145	0.107
Trained λ_1	0.104 to 0.120	0.112
Trained λ_2	0.0917 to 0.120	0.100
Trained λ_3	0.000242 to 0.00148	0.000718
Trained λ_4	1.18 to 1.59	1.37

Table 6.1: Parametric inference of each posterior distribution.

Consider the inference of the regression vector $\vec{\beta}$. Notice that there are four components, when we have only two variables to check. The first component corresponds to the intercept. The second corresponds to the color; because the color can be either blue or white, a single binary regressor (set to equal 1 when blue) is sufficient to encode the information. The third and fourth correspond to nectar; because nectar is ternary, two binary regressors are required to encode the information. (For example, the third component reads 1 when treated with ethanol, the fourth reads 1 when treated with caffeine.) Note that the confidence intervals preserve the sign for the first three coefficients, but not the last. As such, we can conclude that the coefficient of caffeine is the only one of the three that is not appreciably different from zero. To see the implications of these coefficients, consider the following table.

Odds Ratios		
Effect	90% credible int.	median
EtOH over control	0.19 to 0.92	0.42
caffeine over control	0.45 to 2.2	0.99
blue over white	4.8 to 32	11

Table 6.2: Odds ratios for the $\vec{\beta}$ regression results.

To summarize the regression effects, caffeine appears to do nothing, ethanol roughly halves training probability, and bees train over an order of magnitude better to blue flowers than to white flowers. Next, to summarize the mixture model component $\vec{Z}$, consider the following figure.

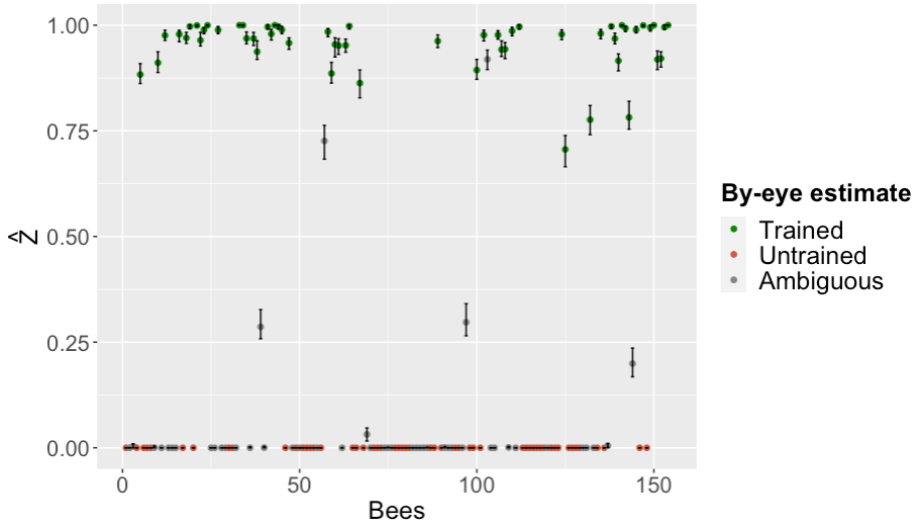


Figure 6.1: Mixture model inference. The x-axis reflects the index of each bee (1 through 154). The y-axis shows the frequency with which the algorithm classifies the bees as trained. The color scheme indicates by-eye characterization of the bees' training states upon viewing the journey graphs. To human eyes, red appear surely untrained, green appear surely trained, and the grey are ambiguous. Note that all surely trained bees are characterized as trained upwards of 75% of the time; even more striking, all surely untrained bees are characterized as untrained nearly 100% of the time. Furthermore, most ambiguous bees are characterized as untrained.

All bees that can be confidently classified by eye fall where we'd expect, with some of the core trained bees being less than perfectly trained. Indeed, only two ambiguous bees are assigned to the trained category. In contrast, the algorithm assigns several ambiguous bees to the untrained category, and with marked confidence, at that. (Note the confidence intervals were computed by repeatedly sampling from the posterior and calculating the proportion of training; the minimum and maximum values define the error bars.) Interestingly, the

untrained core bees are assigned to their expected category noticeably more fervently than their trained counterparts. Indeed, the algorithm appears to be more hesitant to classify bees as trained, suggesting that the inferred definition of trained may be too conservative. Lastly, there are still some bees occupying the middle ground between these two poles. The fact that the majority of the bees congregate to the poles offers evidence supporting the binary training assumption. If a spectrum were more appropriate, we would see more bees falling in between the poles.

To place these regression results in more direct conversation with the training results, we turn to the following modified version of the former figure.

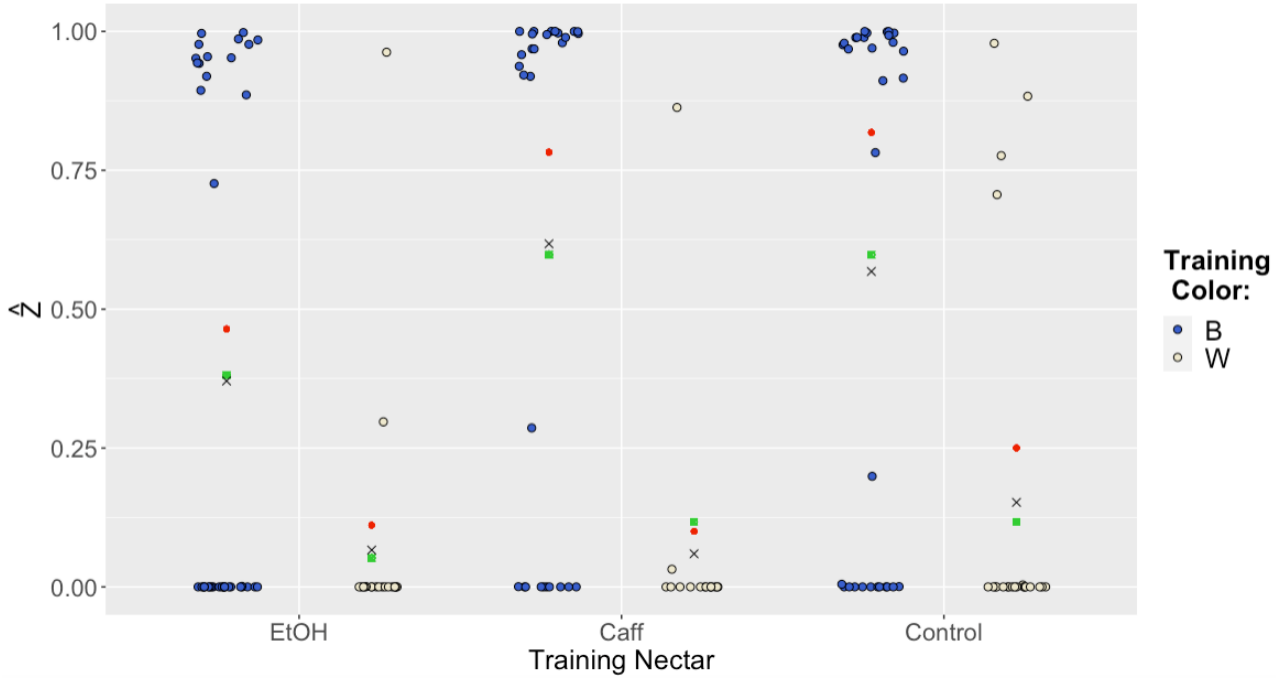


Figure 6.2: Regression and training summary. On the y-axis we have the frequency with which the algorithm classifies the bees as trained. The x-axis clusters the bees according to the experimental design: by nectar treatment and by color training. The black '$\times$' represents the mean value for each group. The green value represents the expected $\hat{p}$ for each group, calculated by the inferred $\vec{\beta}$. The red value represents this same value but based on by-eye estimation.

Note that the red dots, the green dots, and the black cross all approximate same quantity: the probability of a bee being trained given its treatment effects. The red dot reflects our by-eye estimation of this quantity, whereby we simply inspect the data set. The green dot calculates the probability directly using the inferred regression vector; recall, $\hat{p} = \text{logit}(\vec{\beta} \cdot \vec{X})$. The black cross calculates the quantity indirectly by averaging over its $\vec{Z}$ inference. Because the black cross and green dot are both based on parameters inferred by the algorithm, their close overlap indicates high internal consistency within the algorithm. Because these line up quite well with the red dots, we know that the algorithm approaches our by-eye expectations. Taken together, these figures suggest that the algorithm performs

consistent and sensible inference, albeit more conservative—in terms of categorizing bees as trained—than I expected.

6.2 Model checking

Now that we've built some understanding of the inference results, we can consider the fit of the model itself. In the last section, we investigated the fit of the training $\vec{Z}$ and regression $\vec{\beta}$ parameters. We now wish to do the same with the Markov model of the data. How well does our model capture the journeys of our bees?

For this, we use posterior predictive p-values (PPP) [20]. Recall that the Bayesian posterior distribution represents the probability of certain parameters, given the observed data. Our model has sampled from this distribution, yielding high-probability estimates for our parameters of interest. Interrogating the veracity of these estimates, we can ask ourselves what type of data we'd expect to see, if our newly sampled posterior estimates were correct. In other words, we can use the inferred parameters to simulate new data from the posterior predictive distribution,

$$\tilde{D} \sim \pi(\theta|D).$$

We can then compare the real data to the distribution of simulated data. Indeed, the probability of observing the real data $\pi(D|\tilde{D})$ or something more extreme in our simulated distribution is known as a posterior predictive p-value. This is the metric I will use to assess the validity of the model, from the standpoint of the foraging data.

With this handy metric, we need only an entity to measure. Recall the idea of dwell-times: each bee spends a given amount of time in various states. Through the lens of our model, we can break down these dwell-times into specific classes: times spent flying are distinct from those spent in the trained state, which in turn differ from times spent in the novel (or "antitrained") state. Furthermore, whether a bee is trained or untrained will affect these distributions of dwell-times. The fact that we have two distinct rate matrices grants the model the ability to represent these dwell-times differently.

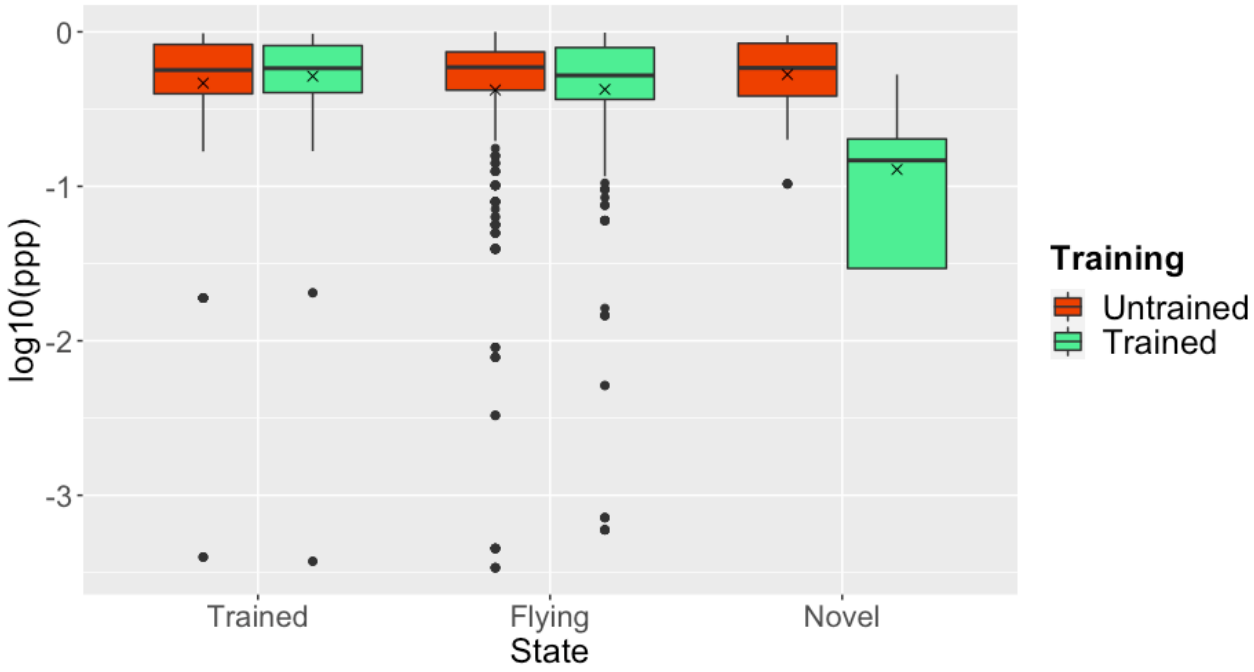


Figure 6.3: Box plots of posterior predictive p-values (PPP) for dwell-times, by state. The y-axis depicts the log probability of observing the data or something more extreme if we assume the inferred parameters were accurate. The x-axis is broken down by the experimental design: each dwell-time (and consequently each PPP) corresponds to a certain state and training status.

The trained state and untrained, novel state dwell-times fit remarkably well: the entire IQRs fall well within 90% credibility, the medians and means are consistently around 50%, and there are only three visible outliers. The flying dwell-times share the same benefits, except that there are noticeably many severe outliers. That is, the model captures the bulk behaviour of the flying dwell-times rather well, but struggles to account for the tail-end behaviour. Lastly, the novel dwell-times for the trained state fit noticeably worse. On the one hand, much of the distribution, including the mean and median, falls within 90% credibility. However, there appears to be formidable weight of the distribution entirely off the graph—that is, far more unlikely than we see here.

To investigate this difficult region, let's overlay the observed data directly on top of the posterior predictive distribution.

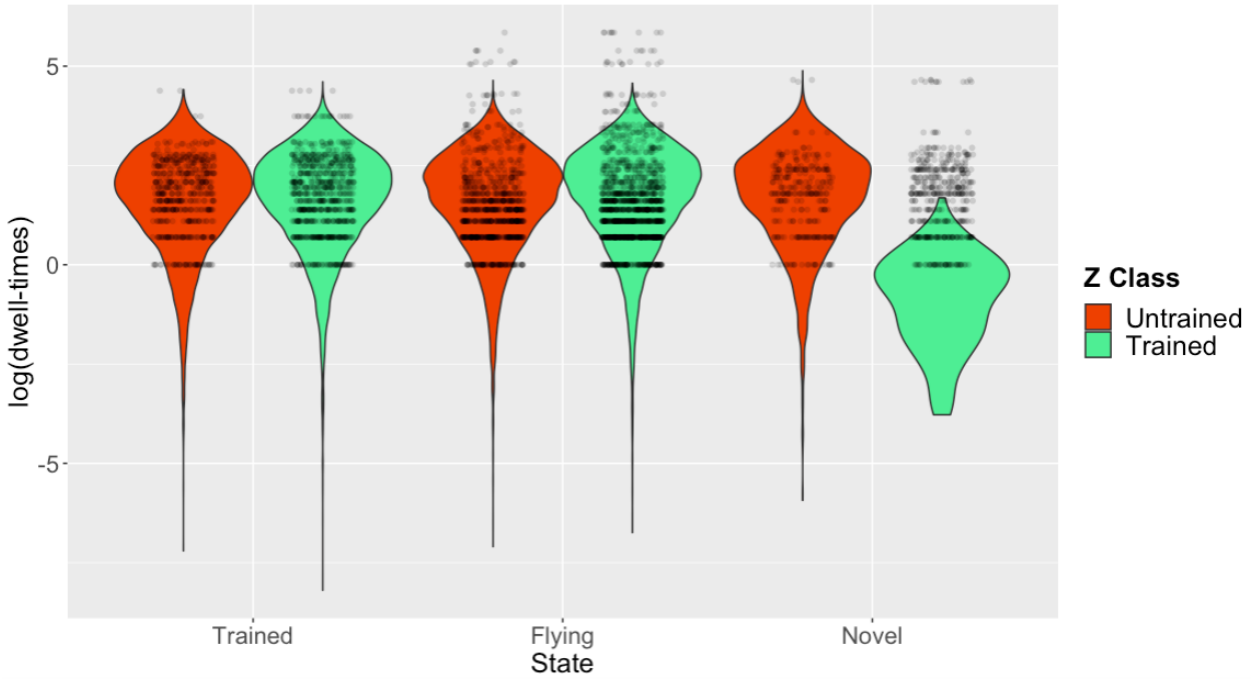


Figure 6.4: Real data superimposed on the inferred posterior predictive distributions. The posterior predictive distributions for each class of dwell-time are plotted in log scale. Laid atop these distributions in black are the dwell-times from the real data set. (Note that the banding results from the dataset being specified to the nearest second.)

These results shed light on PPP box plots: the black dots indicate real data overlaid atop the empirical distributions in question. Indeed, the bulk of the observed, novel dwell-times from the trained state falls outside of the posterior predictive distribution entirely. There are a handful of examples of this occurring for the flying dwell-times, and even fewer for the other dwell-times. As such, it appears that the fit depends fairly drastically on the class of dwell-time in question.

6.3 Dwell-times revisited

Our posterior predictive checks have given us mixed information. On the one hand, the novel dwell-times of the untrained state remain elusive. Yet on the other, the other non-flying dwell-times fit the model excellently. The flying dwell-times are somewhere in between, but seem satisfactory. This prompts us to revisit our understanding of dwell-times. In my exploratory data analysis, I plotted the empirical cumulative density function (ECDF) of all dwell-times with the curve of an exponential CDF (with an MLE fit of the data). These, among many other factors, drove this project to the continuous-time Markov chain model (which in turn prompted the discrete-time approximation in the appendix).

Now that we've broken down the data into trained versus untrained statuses, and

trained, flying, and antitrained states, perhaps we ought to revisit that procedure. In other words, how well does the exponential random variable fit our dwell-times? By answering that question, we can asses how well the CTMC modelling approach fits these experiments in general.

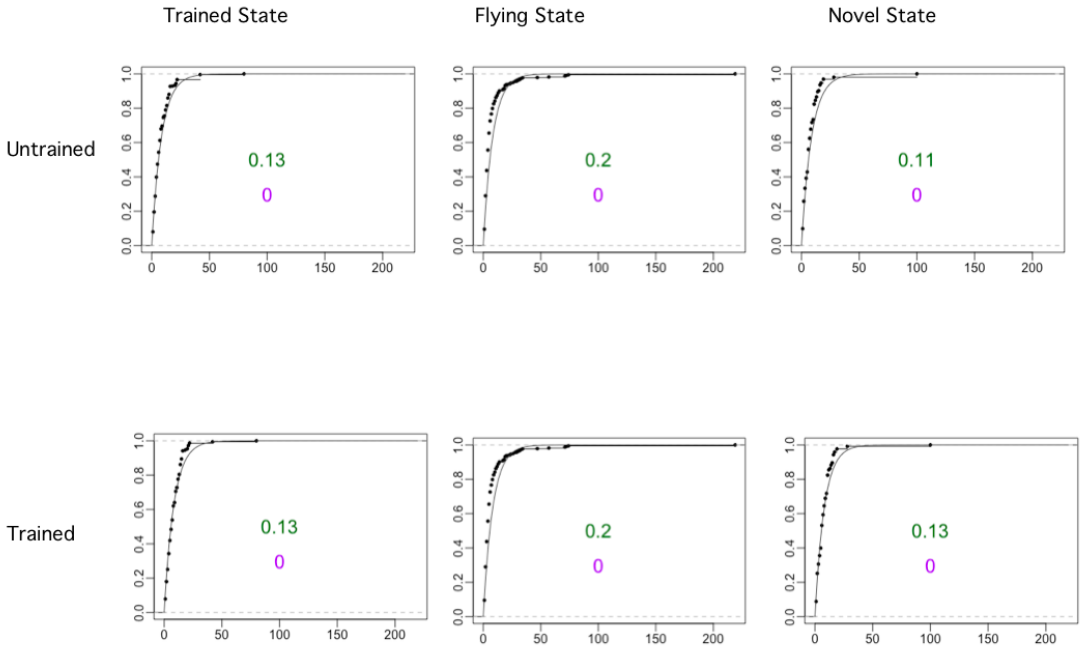


Figure 6.5: ECDFs of observed dwell-times with MLE fit, by state. The two numbers in the center are the results of a one-sided Kolmogorov-Smirnov test for the goodness of fit for the overlaid CDF [21]. The green, top number indicates the D statistic, and the purple, bottom number indicates the corresponding p-value.

Shown above are the ECDFs of all observed dwell-times, separated according to state and status. Overlaid atop these data are exponential CDFs, parameterized by the MLE estimates of the data. Notice, the D statistic is noticeably higher for the flying states, suggesting that the exponential fit is worse for those distributions. Overall, the exponential fit seems appropriate, providing evidence for the CTMC model of $\vec{J}_i$.

Chapter 7

Future Directions

We've seen the continuous-time Markov chain modeling approach produce results that match and expand upon our understanding of the biology. As outlined in the appendix, the discrete-time model excels at rapid inference with very weak prior information, resulting in reasonable training classification and regression inference. The discrete-time Markov chain model of the data thus proves sufficient to a first approximation at making our desired inference, including an approximation of the dwell-time distributions. The continuous-time model, while computationally more intensive, models time notably better than the DTM. We see excellent fit for the floral dwell-times, with the exception of those pertaining to the novel state and untrained status.

The continuous-time algorithm requires more prior information than its discrete-time counterpart. The CTM is more fragile: without help in parsing which rate matrices corresponded to which training states, it sticks to local modes wherein all bees are classified in the same state. To avoid both this issue and the label-switching problem, we instituted a more sophisticated, empirical prior search. We translated our ideas on how trained versus untrained bees behave into rate matrices, which helped the algorithm converge consistently and faster. These rate matrices fit the floral dwell-times even better, and modelled time more honestly. However, the issues of the DTM persisted, as the flying and novel-trained dwell-times resulted in suboptimal posterior predictive p-values.

I originally assumed that I would be following a progression from discrete-time, to continuous-time, to then loosening the exponential dwell-time requirement. This last feature would require a semi-Markov process, which in addition to being more difficult to model (not to mention the memorylessness property would be lost), actually seems unnecessary. Recall that the exponential dwell-time fit is far from the greatest issue: in addition to being mostly consistent across the floral dwell-times from a goodness-of-fit perspective, the posterior predictive p-values are quite promising. Indeed, the weakest part of the exponential fit—the flying dwell-times—is also the least biologically pertinent part of the model.

In general, the CTM appeared to be a more specific, consistent, and confident version of the DTM. However, this resulted in more conservative estimates, where some bees that are likely more trained than untrained were classified as entirely untrained. I am confident this stems from the prior being too rigid. The prior I've outlined represents a first attempt; I have yet to fine-tune how much weight to give the initial rate matrices, and how much leeway to give the algorithm to find a compromise between the bees that are unequivocally trained and those that aren't quite there. We can likely accomplish this with less strict rate matrices, rather than replacing the $\vec{Z}$ dichotomy with a spectrum. This process of prior refinement is the most immediate and attainable next step.

While in my exploratory data analysis I've repeatedly found evidence suggesting that colony doesn't require an explicit random effect, we have begun to glimpse that training might actually look different depending on training color. The DTM argued that blue increases training over 5 to 1, and the CTM doubled that figure. A more honest phenomenon is likely that training simply looks different depending on color. Perhaps white training is simply less severe: we ought to model this at the $\vec{\lambda}$ level, rather than the $\vec{Z}$ level. As a more exciting next step, I will begin to build out a regression scheme on the rate matrix parameters themselves, allowing the training treatments to explicitly affect the definition of training (in addition to the latent probability of training that we currently model). I will likely use Hamiltonian Monte Carlo to accomplish this [22].

The work thus far has not only served to offer reasonable first results, but validated the usefulness of modeling these types of experiments from a Bayesian context. In the near future I hope to offer the insect ethology community the toolkit to holistically, thoroughly, and efficiently analyze secondary metabolite training experiments.

Appendices

Appendix A

Discrete-Time Model

A.1 Discrete-Time Model

The goal in this section is to describe a discrete-time Markov chain model for the journey data. This will allow us to write down the likelihood of the entire multi-step process, and proceed to computational inference of the model's parameters. We have three major components, as described above, the journey data $\vec{J}_i$, the experimental design $\vec{X}_i$, and the training state Z_i. We will first link the journey and training state, and then link the training state to the experimental design.

A.1.1 Link between journey data and training state

To begin, we'll treat time as a series of discrete, ordered quanta, say seconds—this we'll refer to as the discrete-time case. Then, for any given foraging journey, we record the bee's state every second, thereby forming a discrete-time Markov chain (DTMC) (still with the necessary assumption of memorylessness).

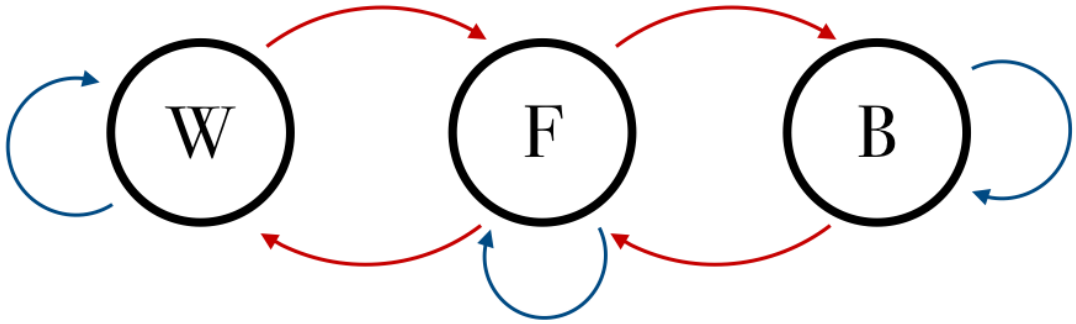


Figure A.1: Discrete-time state space. While this appears nearly identical to its continuous-time counterpart, notice that the empty transitions are shown in blue. In discrete-time, bees are permitted to instantly "transition" back to the state from which they depart.

Necessarily, this means that the chain "transitions" every second; of course, most "transitions" will result in the bee returning immediately back to the state it currently occupies, seeing as the bee typically spends more than a single second in any given state.

We'll call these *empty transitions*, seeing as they're not transitions in a physical sense, but necessities from the mathematical perspective of the DTMC. They are artifacts of discretizing a continuous process, and serve to describe the dwell-times.

The reason for such care concerning transitions in a DTMC has to do with the fact that these stochastic processes are governed by a *transition matrix*, which we'll denote $\boldsymbol{\kappa}$. These square, row-stochastic matrices define the probabilities of transitioning to the next state, given the current state.

$$\begin{array}{c} \\ \text{Current state} \end{array} \begin{array}{c} \text{Next state} \\ \begin{bmatrix} P_{T,T} & P_{T,F} & P_{T,N} \\ P_{F,T} & P_{F,F} & P_{F,N} \\ P_{N,T} & P_{N,F} & P_{N,N} \end{bmatrix} \end{array} = \boldsymbol{\kappa}$$

As such, a transition that a bee undertakes from state s results from the realization of a categorical random variable with range equal to the the state space, whose probability simplex resides in the s^{th} row of $\boldsymbol{\kappa}$. Thus, each second of the experiment, the bee transitions to its next state according to $\boldsymbol{\kappa}$; recall that empty transitions (on the diagonals of $\boldsymbol{\kappa}$) are vastly more likely than physical transitions.

Because a DTMC is defined by its state space and transition matrix, and because we are modelling the experiment such that these DTMCs describe the bees' observed behavior, we can model the bees' output as coming from some $\boldsymbol{\kappa}$. However, we must be slightly more precise in *how* $\boldsymbol{\kappa}$ gives rise to $\vec{J}_i$ to define the DTMC transition probabilities. In order to form a likelihood description of this process, recall the description of a counts matrix $\boldsymbol{C}_i$ from continuous-time.

Now that we can work with counts data, the categorical distribution seems appropriate. Consider the r^{th} row from $\boldsymbol{C}_i$ and $\boldsymbol{\kappa}$: we have a vector of counts and a probability simplex, respectively. Both of these objects describe the same phenomena: $\vec{\kappa}_r$ describes the probability distribution of where the bee transitions to from state r; $\vec{c}_r$ describes the number of counts for each of those transitions. Thus, we can model each row as a categorical distribution, and each matrix as a product of the $|\mathcal{S}| = 3$ rows. This forms the likelihood of the DTMC, and defines what it means to be distributed as a DTMC parameterized by $\boldsymbol{\kappa}$:

$$\pi\left(\vec{J}_i|\boldsymbol{\kappa}\right) = \pi\left(\boldsymbol{C}_i|\boldsymbol{\kappa}\right) = \prod_{r=1}^{3} \underbrace{\left[\prod_{k=1}^{3} \kappa_{r,k}^{c_{r,k}}\right]}_{\textsc{Categorical}(\vec{c_k},\vec{\kappa_k})} .$$

Because we're building a Bayesian model, we require a prior on $\boldsymbol{\kappa}$. To that end, we'll use the Dirichlet distribution as a prior on $\boldsymbol{\kappa}$. Let $\boldsymbol{\alpha}$ be a matrix of the same dimensions as $\boldsymbol{\kappa}$, comprise of $|\mathcal{S}|$ Dirichlet vectors joined row-wise. In other words, the r^{th} row of $\boldsymbol{\kappa}$ is distributed according to a Dirichlet distribution parameterized by the r^{th} row of $\boldsymbol{\alpha}$; that is, $\vec{\kappa}_r \sim \textsc{Dirichlet}(\vec{\alpha}_r)$. Using a slightly abusive notation, we can see the larger picture as follows,

$$\begin{bmatrix} \kappa_{T,T} & \kappa_{T,F} & \kappa_{T,N} \\ \kappa_{F,T} & \kappa_{F,F} & \kappa_{F,N} \\ \kappa_{N,T} & \kappa_{N,F} & \kappa_{N,N} \end{bmatrix} \sim \textsc{Dirichlet}\left(\begin{bmatrix} \alpha_{T,T} & \alpha_{T,F} & \alpha_{T,N} \\ \alpha_{F,T} & \alpha_{F,F} & \alpha_{F,N} \\ \alpha_{N,T} & \alpha_{N,F} & \alpha_{N,N} \end{bmatrix} \right).$$

Our model assumes there exist two distinct training classes: trained and untrained. By definition, for a bee i to belong to a training class Z, it is necessarily the case that $\vec{J}_i \sim \text{DTMC}(\boldsymbol{\kappa}_Z)$. As such, there are two $\boldsymbol{\kappa}$ parameters to infer, each with a corresponding $\boldsymbol{\alpha}$ prior.

A.1.2 Link between training state and experimental design

With the training state Z_i and journey data $\vec{J}_i$ linked via $\boldsymbol{\kappa}$, the next task is to link the experimental design $\vec{X}_i$ to the training state Z_i. This follows the same structure as the CTM.

A.1.3 Hierarchical model

Having connected the components of the mode, we can finally visualize the it as a directed acyclic graph (DAG), and annotate it with the generative model.

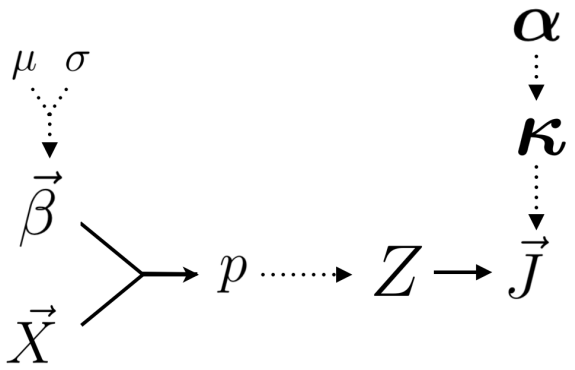


Figure A.2: Discrete-time model DAG. The solid lines indicate deterministic connections, and the dotted lines represent stochastic connections. While this appears very similar to its continuous-time counterpart, please see the generative model for details.

Our prior distributions are defined,

$$\vec{\beta} \sim \textsc{Normal}(\mu, \sigma),$$
$$\boldsymbol{\kappa} \sim \textsc{Dirichlet}(\boldsymbol{\alpha}).$$

We thus have the experimental design determining the training state via the logistic,

$$p_i = \text{logit}[\vec{X}_i \cdot \vec{\beta}],$$
$$Z_i \sim \textsc{Bernoulli}(p_i),$$

and the training state determining the journey data,

$$\vec{J}_i|\boldsymbol{\kappa}_{Z_i} \sim \text{DTMC}(\boldsymbol{\kappa}_{Z_i}),$$
$$\pi(\vec{J}_i|\boldsymbol{\kappa}_{Z_i}) = \prod_{r=1}^{3}\left[\prod_{k=1}^{3}\kappa_{r,k}^{c_{r,k}}\right].$$

A.2 Discrete-Time Algorithm

As outlined before, the DTM algorithm alternates between updating the transition matrices $\boldsymbol{\kappa}$, the regression coefficients $\vec{\beta}$, and the training states $\vec{Z}$. Once again, time is the differentiating feature; it will affect not just the Markov chain likelihood, but the training state posteriors (and ultimately the regression coefficients). The following explanation of each type of move will outline how we sample from each conditional posterior.

A.2.1 Move 1: Gibbs sampler for updating the transition matrices

For this process, we know that the transition matrices and their Dirichlet parameters are conditionally independent of the regression parameters, given the training states. Thus, we can consider only the Dirichlet and transition matrices with the training states. Recall the discussion from the previous section. The i^{th} row of $\boldsymbol{\kappa}$ (the transition matrix) is distributed as a Dirichlet of the form $\textsc{Diric}(\kappa_{i,1}, \kappa_{i,2}, \kappa_{i,3}|\alpha_{i,1}, \alpha_{i,2}, \alpha_{i,3})$, or more succinctly $\textsc{Diric}(\vec{\kappa_i}|\vec{\alpha_i})$, for $i \in \mathcal{S}$.

To develop this idea, consider the probability density of the Dirichlet distribution,

$$\pi(\kappa_i, \ldots, \kappa_k|\alpha_1, \ldots, \alpha_k) = \frac{1}{B(\vec{\alpha})}\prod_{i=1}^{K}\kappa_i^{\alpha_i - 1},$$

where $\sum_{i=1}^{k}\kappa_i = 1$, and $\kappa_i \geq 0$ for all $i \in \{1, \ldots, K\}$; and the normalizing constant in terms of the multivariate beta function is

$$B(\vec{\alpha}) = \frac{\prod_{i=1}^{K}\Gamma(\alpha_i)}{\Gamma(\sum_{i=1}^{K}\alpha_i)}.$$

We can use this function to demonstrate that the Dirichlet distribution is a conjugate prior of the categorical distribution (shown in the appendix)[1]. Recall that conjugacy refers to a particular pair of prior and likelihood distributions interacting such that the posterior distribution is of the same form as the prior.

As a typical Gibbs sampler, this type of move is effectively sampling from the conditional distribution of the three $\vec{\kappa}$'s, conditioned upon a given set of training outcomes $\vec{Z}$. Suppose

[1]Note that the appendix demonstrates multinomial-Dirichlet conjugacy, which is a slightly more general form of what our model requires.

we have sampled outcomes for $\vec{Z}$. Recall that we can calculate $\boldsymbol{C}_i$ for each bee i, by counting the pairwise entries (transitions) of $\vec{J}_i$. Furthermore, because we assume that there exist only two classes of $\boldsymbol{\alpha}$, we can then aggregate all $\boldsymbol{C}_i$ of the same class to generate two matrices in total:

$$\boldsymbol{C}_T = \sum_{\forall Z_i = T} \boldsymbol{C}_i \quad \text{and} \quad \boldsymbol{C}_U = \sum_{\forall Z_i = U} \boldsymbol{C}_i,$$

because for all i, $Z_i \in \{T, U\}$, by our conditioning. Notice, we have all we need to update our $\boldsymbol{\alpha}$ and $\boldsymbol{\kappa}$ for both training classes:

$$\begin{aligned} \boldsymbol{\alpha}'_T &= \boldsymbol{\alpha}_T + \boldsymbol{C}_T, & \boldsymbol{\alpha}'_U &= \boldsymbol{\alpha}_U + \boldsymbol{C}_U, \\ \boldsymbol{\kappa'}_T &\sim \textsc{Diric}(\boldsymbol{\alpha'}_T), & \boldsymbol{\kappa'}_U &\sim \textsc{Diric}(\boldsymbol{\alpha'}_U), \end{aligned}$$

(noting again that the matrices correspond to three rows of vectors in $\mathbb{R}^3$, for which the distribution relation is defined). Hence, we've sampled two new $\boldsymbol{\kappa}$ from the six corresponding posterior distributions $\pi(\vec{\kappa}|\vec{c}, \vec{\alpha})$, which obey Dirichlet distributions parameterized by $\vec{\alpha}' = \vec{\alpha} + \vec{c}$.

A.2.2 Move 2: Metropolis-Hastings sampling of the regression coefficients

Same as continuous-time. See Chapter 3.

A.2.3 Move 3: Training state posterior calculation in the DTM

In the case where we hold the transition matrix and regression parameters constant, we are actually able to sample from the training state posterior directly. First, recall that because $Z_i \sim \textsc{Bernoulli}(p_i)$, there are only two possible outcomes of Z_i, denoted $m \in \{\text{Trained}, \text{Untrained}\}$. As such, we can actually calculate the posterior directly by the Law of Total Probability. We know the likelihood and prior of both possible states, which defines an unnormalized posterior (it lacks the marginal, which is the normalizing constant from Bayes's Law). Then, by dividing through with their aggregate, we can normalize the posterior.

To that end, we must consider the prior and likelihood. The prior is simply the underlying probability of training given the treatment regimen; this comes from the regression. The likelihood is the probability of the journey data occurring if it is distributed as a discrete-time Markov chain with transition matrix $\boldsymbol{\kappa}$. (For ease of reading, we group $\vec{J}_i$, $\boldsymbol{C}_i$, and $\vec{X}_i$ as data D; and we use $\boldsymbol{\kappa}$ to represent both the trained and untrained matrices). We can see all

this mathematically as

$$\begin{aligned}\underbrace{\pi(Z_i = m|D, \vec{\beta}, \boldsymbol{\kappa})}_{\text{posterior}} &\propto \sum_{\forall m} \left(\underbrace{\pi(\vec{J}_i|\boldsymbol{\kappa}_m)}_{\text{likelihood}} \cdot \underbrace{\pi(Z_i = m|\vec{X}_i, \vec{\beta})}_{\text{prior}} \right) \\ &\propto \prod_{r=1}^{3} \left[\left(\prod_{k=1}^{3} \kappa_{T,r,k}^{c_{r,k}} \right) \right] \cdot \left(\underbrace{\text{logit}(\vec{X}_i \cdot \vec{\beta})}_{p_i} \right) \\ &+ \prod_{r=1}^{3} \left[\left(\prod_{k=1}^{3} \kappa_{U,r,k}^{c_{r,k}} \right) \right] \cdot \left(\underbrace{1 - \text{logit}(\vec{X}_i \cdot \vec{\beta})}_{1-p_i} \right),\end{aligned}$$

recalling that the prior probability of training p_i arises from the logit calculation of the regression, and that $\vec{J}_i$ gives us $\boldsymbol{C}_i$ by the Markov property.

Because we've defined all states that the unnormalized posterior can take, we can normalize these states by dividing through with their sum. Thus, for an example of one of the posterior probabilities, consider the probability of being trained:

$$\pi(Z_i = T|D, \vec{\beta}, \boldsymbol{\kappa}) = \frac{p_i \prod_{r=1}^{3} \prod_{k=1}^{3} \kappa_{T,r,k}^{c_{r,k}}}{p_i \prod_{r=1}^{3} \prod_{k=1}^{3} \kappa_{T,r,k}^{c_{r,k}} + (1 - p_i) \prod_{r=1}^{3} \prod_{k=1}^{3} \kappa_{U,r,k}^{c_{r,k}}}.$$

We've thus fully specified the posterior distribution for the training state. By evaluating this expression, we find the posterior probabilities of training versus untraining, which then makes updating $\vec{Z}$ equivalent to a series of weighted coin tosses.

A.2.4 Compound DTM Algorithm

The basic structure is very similar to the CTM. The regression coefficients are the same as the CTM. The $\boldsymbol{\alpha}$ parameters were set such that no weight was given to illegal moves (located on the off-diagonals, representing a flower change without flying), and most weight went to non-physical transitions (located on the diagonals). Bees overwhelmingly stay in their current state (one second is rather short), so these non-physical transitions typically occur with probability upwards of 90%. While this may be an odd artifact of discretizing a continuous process, it is not difficult to accommodate; here are the Dirichlet priors used to produce the transition matrices described above. (Notice that these priors differ only in $(2, 1)$; we'll come back on this momentarily.)

$$\boldsymbol{\alpha}_U = \begin{bmatrix} 20 & 1 & 0 \\ 1 & 20 & 1 \\ 0 & 1 & 20 \end{bmatrix}, \quad \boldsymbol{\alpha}_T = \begin{bmatrix} 20 & 1 & 0 \\ 4 & 20 & 1 \\ 0 & 1 & 20 \end{bmatrix}.$$

One of the trickiest aspects of mixture models is actually specifying which mixture corresponds to which phenomenon. In other words, how can we inform the computer which set of parameters represents training, and which represents being untrained. We can't do

this after the fact—the regression coefficients change depending on whether the algorithm assigns the proper $\boldsymbol{\kappa}$ parameters or switches them (each of these labeling schemes being equally likely). To that end, we employ a step in the algorithm that we call the *unmixing step*. After the first round of tempering the $\boldsymbol{\kappa}$ parameters with the training states, we inspect the training states of a group of bees whose training states are obvious from inspecting the data. These are called core bees, and only ever visit their trained flower. Overwhelmingly, these bees are all either classified as ones or zeros. In the case that they are zero, we switch them to one, and reassign which $\boldsymbol{\kappa}$ and $\boldsymbol{\alpha}$ parameters go to which training state. This is known as the label switching problem [16].

Recall the transformation of the state space in Chapter 4. When we discuss $(2, 1)$ of a transition matrix, we know that, independent of the training color, we are investigating the probability of going from flying to the trained state. Thus, when our prior assigns more weight to that entry, it reflects our prior belief that a trained bee will choose a trained state more frequently than a novel state. However, should the label switching problem occur (which still happens, but less frequently), we switch the priors as well, because they make the argument that trained and untrained bees behave differently *a priori* of the data.

A.3 Simulation Studies

Note that this these studies are very similar to their continuous-time counterparts. Note that a novel performance metric will be employed to assess the distance in $\boldsymbol{\kappa}$. Because each row of $\boldsymbol{\kappa}$ represents a probability distribution, we calculate the row-wise Jenson-Shannon Divergence (JSD) [23] of each estimated $\boldsymbol{\kappa}$ from the underlying $\boldsymbol{\kappa}$ (as such, the calculated value reflects the sum of six different JSD values). Once again, identical matrices will have JSD equal to zero. The following demonstrates convergence for the DTM (again, parameters located in the appendix for Chapter 5).

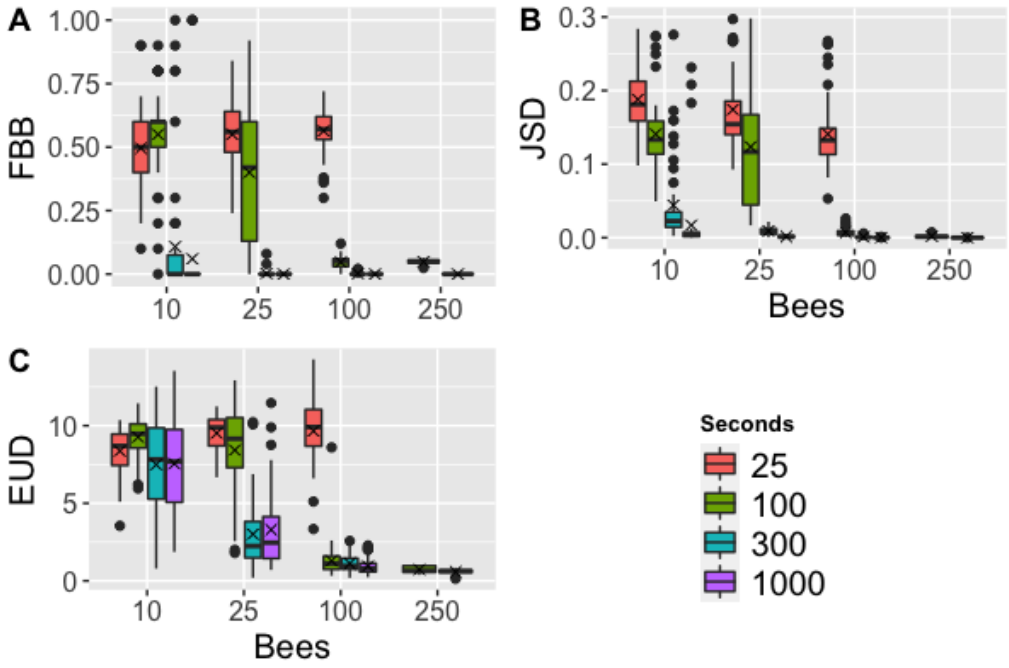


Figure A.3: Sample size increases convergence and performance of the DTM. On the x-axis, we see four different bee sample sizes, each of which contains four different trials with increasing flight durations. Indeed, we can get a rough comparison of sample size with the product of the flight time and bees. Overall, the performance clearly increases with sample size. For example, by the time we see 100 bees with 300 seconds each (the real data has 154 bees with an average of 224 seconds each), the FBB has reached zero, the transition matrix distance is vanishingly small, and the regression vector distance is remarkably small. Meanwhile, the low sample sizes are fraught with outliers and wide IQRs, showing very unreliable performance. Lastly, by the time we reach 1000 bees with 1000 seconds each, the performance is effectively indistinguishable from perfection. Note the black bar indicates the median, and the black '$\times$' indicates the mean. See the appendix for the definition of which $\boldsymbol{\kappa}$ and $\vec{\beta}$ were used (denoted as "stable").

Now that we've established that the algorithm appears to improve with more information, we can specifically probe the role each parameter class plays in performance. To begin, let's consider $\boldsymbol{\kappa}$, the transition matrices. Seeing as this is where the algorithm defines what it means for a bee to be trained or untrained, we can intuitively surmise that the algorithm's performance here is a function of $\boldsymbol{\kappa}$ similarity. In other words, the more similar the transition matrices are, the most difficult it should be for the algorithm to parse which bee is which. As such, we'll define three $\boldsymbol{\kappa}$ pairs of low, medium, and high similarity. Granted, this similarity is based on our human intuition, rather than some universal standard. For full enumeration of these values, refer to the chapter-specific appendix. Note that the following two figures use 100 seconds per flight.

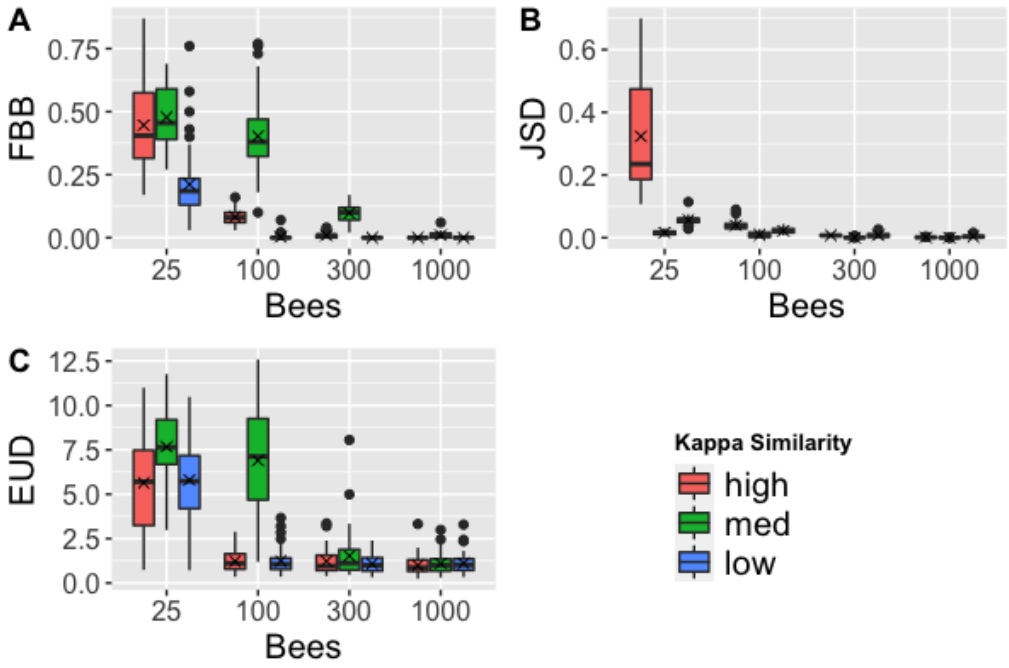


Figure A.4: Mixture distinction increases performance and convergence of the DTM. Kappa similarity refers to how distinct the $\boldsymbol{\kappa}$ transition matrices are. Generally, as similarity decreases, performance increases. See the appendix for Chapter 5 for full specification.

When we compare high to low $\boldsymbol{\kappa}$ similarity, we clearly see what we'd expect: the algorithm generally performs better when the transition matrices differ more. However, what seemed to us like a middle ground between these two supposed extremes actually struggles more than the other two in $\vec{Z}$ and $\vec{\beta}$ inference, but exceeds both in $\boldsymbol{\kappa}$ performance. In other words, the algorithm determines the differences in the transition matrices, but has more difficulty than expected in applying those differences to group the bees.

Lastly, because logistic models can have notoriously flat curvature (which can be difficult to infer with acceptable specificity), we wish to verify that the algorithm performs well under a variety of different $\vec{\beta}$ parameters. Consider the following assortment of distinct $\vec{\beta}$ options (outlined specifically in the appendix).

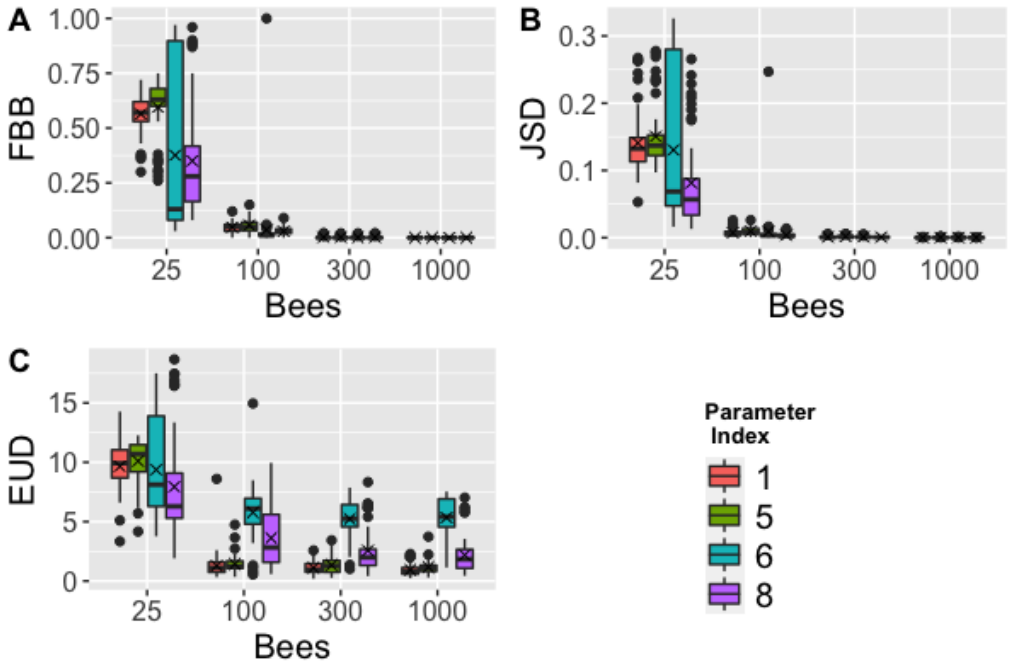


Figure A.5: Convergence achieved with various regression coefficients. The parameter index corresponds to distinct $\vec{\beta}$ values. See the appendix for Chapter 5 for full specification.

Once again, we see the simulations increasing in accuracy and precision as sample size increases. Notice that not all parameterizations behave equally; the two rightmost selections have more difficulty converging, particularly with respect the $\vec{\beta}$, than their leftmost counterparts. Overall, these results are sufficiently encouraging across a variety of parameterization that we can feel confident proceeding to the analysis of the real data.

A.4 Real Data

The first result to check would be the training inference, seeing as we can most intuitively compare those results against the flight paths we observe. Along these lines, recall that the algorithm functions by considering a group of bees that we are confident must be trained, referred to as core bees. (Note again that this doesn't affect the mathematics at play, but rather accounts for the label-switching problem in the mixture model.) Independently of our model, we can find bees by inspection that appear obviously untrained. We then have three categories according to our own eyes: trained, untrained, and ambiguous. While these categories do not affect any math, they offer us a basis from which we can interpret how reasonable the algorithm's assessments are.

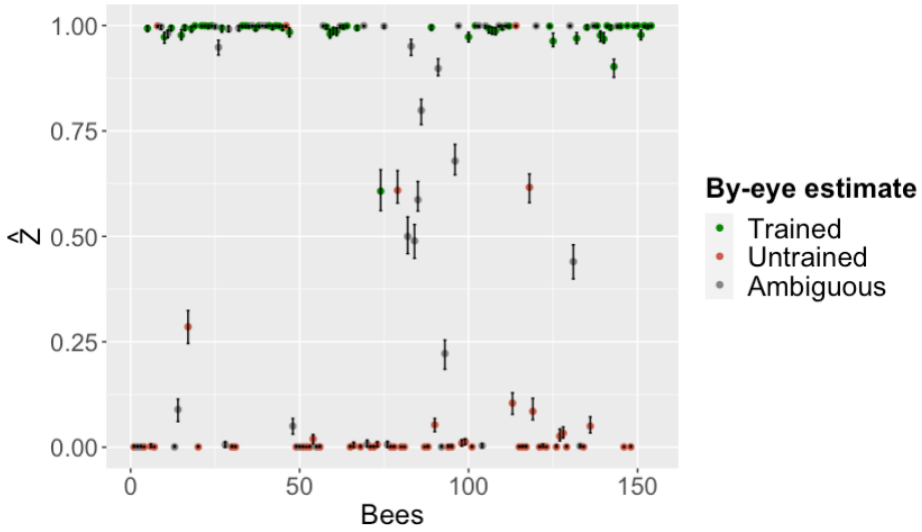


Figure A.6: DTM mixture inference. The colors indicate the by-eye classification of the bees: surely trained in green, surely untrained in red, or ambiguous in grey. The plot shows fraction trained (1 being completely trained, 0 being completely untrained) versus bee index.

Shown above is the algorthm's inference of $\vec{Z}$. Using the aggregate posterior (across all ten separate runs), the training fraction was calculated for each individual bee. This fraction is defined by the number of chain states spent in the trained category (1) divided by the total number of chain states (1 or 0). (The error bars are defined the same way as the CTM.) First, notice that the overwhelming majority of bees occupy one state with remarkable consistency: bees tend to be either completely trained or completely untrained. The algorithm is thus identifying bees with greater confidence than our by-eye estimation. Note further that of the bees that we confidently classified as trained or untrained by eye, all but seven bees fall exactly where we'd expect. Accordingly, the majority of the few bees that do not resoundingly occupy one state were difficult to classify by eye; as such, we are not surprised the algorithm had difficulty defining a single answer as well. Yet, even for those bees that eluded a single classification by eye, the algorithm typically determines a classification with remarkable consistency. Overall, these findings demonstrate that the algorithm reliably parses the mixture model such that each bee receives a consistent training category.

Now that we've established that the algorithm can parse the mixture, we can turn to the parameters of most biological interest: the $\vec{\beta}$ regression.

Posterior Results		
Parameters	90% credible interval	median
β Intercept	-1.1 to 0.17	-0.47
β Blue?	1.0 to 2.5	1.7
β EtOH?	-1.6 to 0.023	-0.75
β Caff?	-0.82 to 0.78	-0.0061

Table A.1: DTM Parametric inference of each posterior distribution.

With that in mind, we can calculate the odds ratios for various different treatment regimens (using medians for calculations).

Odds of training...	Calculation	Value
alcohol over control	$\exp(\beta_3)$	0.48
caffeine over control	$\exp(\beta_4)$	1
blue over white	$\exp(\beta_2)$	5.47

Table A.2: Odds ratio interpretations of DTM regression coefficients

Clearly, these coefficients imply that color controls training hugely. At the same time, caffeine seems to not significantly affect training, while alcohol roughly halves training efficacy. We can see these ideas spatially in the following plots.

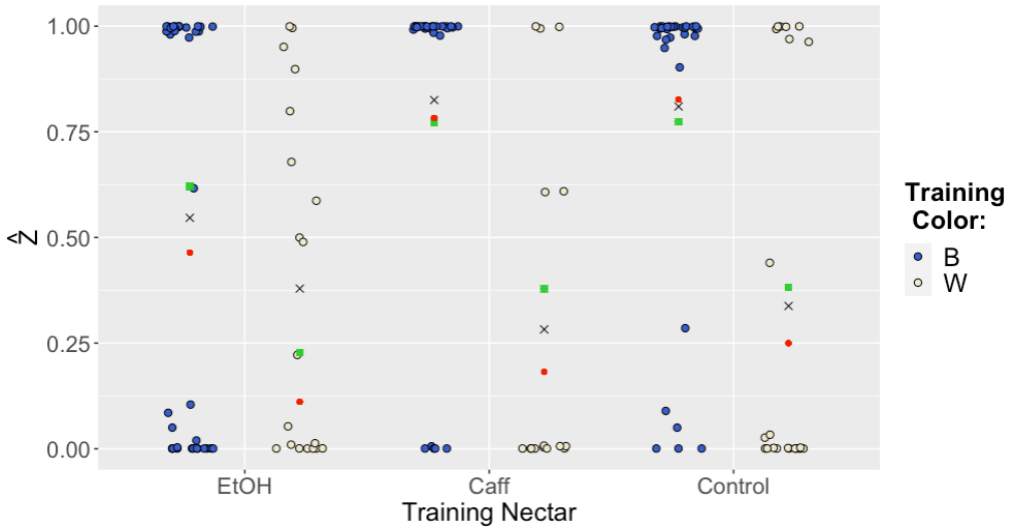


Figure A.7: DTM regression and training summary. Here we see the fraction trained of each bee, as inferred by the algorithm. We've mapped the structure of the design matrix onto the grouping scheme: note the three groups of two columns, reflecting nectar and color. The black cross indicates the mean proportion of trained bees for each grouping, according to the algorithm; in other words, this statistic averages the $\vec{Z}$ estimates. We can compare this to the baseline probability of training occurring for each entry of the design matrix, as calculated by the regression coefficients (green point). To compare this result to what we expect by eye, we calculated a rough proportion of trained individuals beforehand (shown in red). Note, these points are quite close together for each entry, showing that the inferred parameters imply training probabilities that meet our expectations from inspecting the data, and are internally consistent with the algorithm's $\vec{Z}$ classification.

The implication of these probabilities in terms of how the bees spend their time can be seen graphically with the following violin plot.

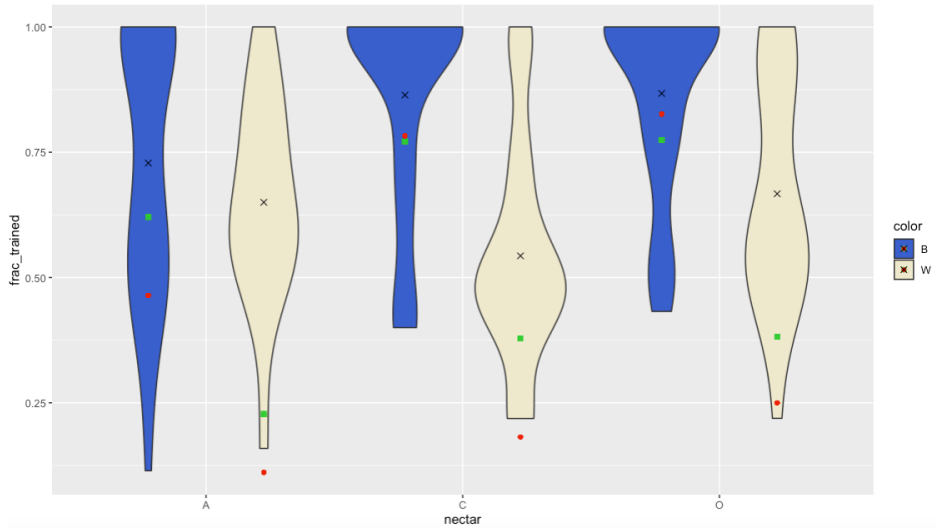


Figure A.8: The fraction trained metric on the y-axis reflects the total number of seconds spent in the trained state divided by the total number of seconds spent in either the trained or untrained state. Effectively, this gives us better resolution on how the bees' training manifests in their state preference. Trained blue individuals, particularly in the control and caffeine groups, appear to spend far more time in the trained state, as shown by the clustering of mass toward the top of the graph. This is supported by the higher predicted (green) and observed (red) training frequencies. In contrast, the white individuals cluster more mass toward the 50% mark, indicating a lack of preference for either state. Once again, the predicted and observed point estimates suggest that these bees train less effectively. Recall, from the table above, the over five-fold increase in training odds for blue over white.

A.5 Model Checking

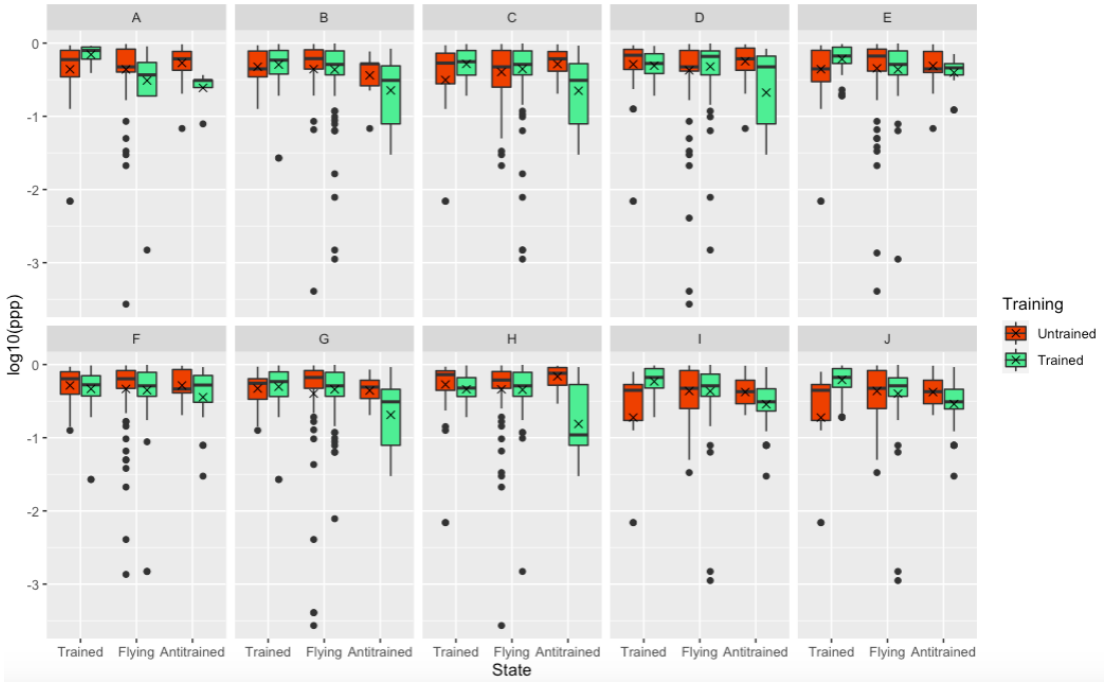


Figure A.9: Shown above we have the posterior predictive p-values (ppp) for each experimentally observed dwell-time. Across all colonies, treatments, and trainings, we find the real data lying within the 90% credible interval. Notice, this isn't quite true with some of the antitrained dwell-times of trained bees. For these bees, the real data are more extreme than the other cases. This is to be expected; since trained bees spend very little time in novel states, we have the least amount of data for these cases. In a similar vein, we see that many flying dwell-times are remarkably extreme relative to the posterior predictive distribution (although the IQRs seem to always fall well within 90% credibility). This is even less concerning, seeing as the time spent flying is less important to the biology in question than the flower preferences of the bees. Indeed, for those floral dwell-times, we see excellent fit, with means and medians of the p-values falling well within the bulk of the distribution. Lastly, note that the colony-specific distinctions appear highly unremarkable, especially when we consider the differential sample sizes at play. This supports our modeling decision to avoid an explicit random effect for colony.

Appendix B

Chapter-Specific Appendices

Appendix for Chapter 1: Background

Theorem 1.1 (**Exponential Memorylessness**). *If X is a positive continuous random variable with the memorylessness property, then $X \sim \text{EXPONENTIAL}(\lambda)$ for some positive λ.*

Proof: Let F be the CDF of X, and $G(x) = \mathbb{P}(X > x) = 1 - F$. Recall that memorylessness can be represented as

$$\mathbb{P}(s + t > x | s > x) = \mathbb{P}(t > x).$$

If we think in terms of dwell-times, this is equivalent to saying that the probability of waiting for $x + t$ seconds, given that we've already waited for s seconds, is equal to the probability of waiting t seconds. Now, suppose $G(x)$ is memoryless. Then,

$$G(s + t) = \mathbb{P}(s + t > x) = \mathbb{P}(s + t > x | s > x)\mathbb{P}(s > x) = \mathbb{P}(t > x)\mathbb{P}(s > x).$$

Now, we can clearly see that we wish to solve for $G(x)$, a class of functions, such that

$$G(s + t) = G(s)G(t).$$

To this end, let's consider some values of t. Let $s = t$. Then, $G(2t) = G(t)^2$. By simple induction, this quickly translates to $G(kt) = G(t)^k$ for all $k \in \mathbb{N}$. Consider next $G\left(\frac{t}{2}\right)$. If we recall that $G(2t) = G(t)^2$, and then substitute $\frac{t}{2}$ for t, we find that $G(t) = G\left(\frac{t}{2}\right)^2$, or $G\left(\frac{t}{2}\right) = G(t)^{1/2}$. By a parallel induction, our k can now be the reciprocal of a natural number, $k \in \frac{1}{\mathbb{N}} \cup \mathbb{N}$. Taking this further, for natural numbers M, N, we see

$$G\left(\frac{M}{N}t\right) = G(t)^M G(t)^{\frac{1}{N}} = G(t)^{\frac{M}{N}}.$$

In other words, $k \in \mathbb{Q}$. However, because the rationals are dense in the reals, we actually know $G(xt) = G(t)^x$, for all $x \in \mathbb{R}_{>0}$.

Now, suppose $t = 1$. Then,

$$G(x) = G(1)^x = e^{x \ln G(1)}.$$

However, recall that G is a probability distribution, meaning it is contained in the closed interval of $[0, 1]$ on the real line. The logarithm function maps the points of this region to negative real numbers, meaning $\ln([0, 1]) = -\lambda$, such that $\lambda \in \mathbb{R}_{>0}$. Thus, $G(x) = e^{-\lambda x} = 1 - F(x)$, or equivalently, $F(x) = 1 - e^{-\lambda x}$. We have recovered the CDF of an exponential, proving the claim.[24] □

Theorem 1.2 (**Metropolis-Hastings Algorithm**). *If the proposition distribution $\boldsymbol{Q}$ is ergodic, the MHA builds a Markov chain X with a stationary distribution of π^*.*

Proof: We first note that $X_0, X_1, \ldots$ form a Markov chain, as the Markov property holds (the selection process for the next state depends only on the current state). Define $\boldsymbol{P}$ as the transition matrix for the chain. Our task is to demonstrate that detailed balance holds for $\boldsymbol{P}$ and π^*.

Consider an arbitrary transition from state i to state j. We will be using the fact that $P_{ij} = Q_{ij} \cdot \mathbb{P}(U \leq \alpha_{ij})$, which reflects the probabilities of being proposed and accepted respectively, linked by the basic principle of counting. Additionally, recall the ratio definition of α from the algorithm, which will be referenced frequently. Relying on the trichotomy of the real numbers, we'll consider the following three cases.

1. Suppose $\pi_i^* Q_{ij} < \pi_j^* Q_{ji}$. Then, $\alpha_{ij} > 1$ and $\mathbb{P}(U \leq \alpha_{ij}) = 1$; by complementary logic, $\alpha_{ji} < 1$, and $\mathbb{P}(U \leq \alpha_{ji}) = \alpha_{ji}$. Now, consider

$$\begin{aligned}
\pi_i^* P_{ij} &= \pi_i^* Q_{ij} \cdot \mathbb{P}(U \leq \alpha_{ij}) \\
&= \pi_i^* Q_{ij} = \pi_j^* Q_{ji} \cdot \left(\frac{\pi_i^* Q_{ij}}{\pi_j^* Q_{ji}} \right) \\
&= \pi_j^* Q_{ji} \cdot \alpha_{ji} = \pi_j^* Q_{ji} \cdot \mathbb{P}(U \leq \alpha_{ji}) \\
&= \pi_j^* P_{ji}.
\end{aligned}$$

2. Suppose $\pi_i^* Q_{ij} > \pi_j^* Q_{ji}$. Then, $\alpha_{ij} < 1$ and $\mathbb{P}(U \leq \alpha_{ij}) = \alpha_{ij}$; furthermore, $\alpha_{ji} > 1$, so $\mathbb{P}(U \leq \alpha_{ji}) = 1$. Now, consider

$$\begin{aligned}
\pi_i^* P_{ij} &= \pi_i^* Q_{ij} \cdot \mathbb{P}(U \leq \alpha_{ij}) \\
&= \pi_i^* Q_{ij} \cdot \alpha_{ij} = \pi_i^* Q_{ij} \cdot \frac{\pi_j^* Q_{ji}}{\pi_i^* Q_{ij}} \\
&= \pi_j^* Q_{ji} = \pi_j^* Q_{ji} \cdot \mathbb{P}(U \leq \alpha_{ji}) \\
&= \pi_j^* P_{ji}.
\end{aligned}$$

3. Suppose $\pi_i^* Q_{ij} = \pi_j^* Q_{ji}$. Then, $\alpha_{ij} = 1 = \alpha_{ji}$, and $\mathbb{P}(U \leq \alpha_{ij}) = 1 = \mathbb{P}(U \leq \alpha_{ji})$. Finally, consider

$$\begin{aligned}
\pi_i^* P_{ij} &= \pi_i^* Q_{ij} \cdot \mathbb{P}(U \leq \alpha_{ij}) \\
&= \pi_i^* Q_{ij} \cdot \frac{\pi_j^* Q_{ji}}{\pi_j^* Q_{ji}} \\
&= \pi_j^* Q_{ji} \mathbb{P}(U \leq \alpha_{ji}) \\
&= \pi_j^* P_{ji}. \quad \square
\end{aligned}$$

Appendix for Chapter 3: Model

Theorem 3.1 (**Minimum of Exponentials**). *The minimum of two independent exponential random variables is itself exponentially distributed, with a new rate equal to the sum of the two rates.*

Suppose we have $Z = \min\{X, Y\}$, where

$$X \overset{i.i.d}{\sim} \textsc{Exp}(\lambda), \quad Y \overset{i.i.d}{\sim} \textsc{Exp}(\mu),$$

and X is independent of Y. Then, $\mathbb{P}(Z \geq t) = \mathbb{P}(X \geq t, Y \geq t)$ because Z is the minimum of X and Y. Thus, we find

$$\begin{aligned}
\mathbb{P}(Z > t) &= \mathbb{P}(X > t, Y, > t) \\
&= \mathbb{P}(X > t)\mathbb{P}(Y > t) && \text{(by independence)} \\
&= [1 - \mathbb{P}(X \leq t)][1 - \mathbb{P}(Y \leq t)] \\
&= \left[1 - (1 - e^{-\lambda t})\right]\left[1 - (1 - e^{-\mu t})\right] && \text{(CDF of exponential)} \\
\mathbb{P}(Z > t) &= e^{-(\lambda+\mu)t} \\
\mathbb{P}(Z \leq t) &= 1 - e^{-(\lambda+\mu)t},
\end{aligned}$$

which is the CDF of an exponential with a rate equal to the sum of the smaller rates. (Note, this generalizes readily through induction, but for our purposes we need only the base case.) □

As a **corollary** to this theorem, consider (without loss of generality) the probability that a minimum $Z = t$ resulted from X and not Y. In other words, what's the probability that the X alarm clock detonates before the Y alarm clock? By construction, note that $\mathbb{P}(Z = t) = \mathbb{P}(X = t, Y > t) + \mathbb{P}(X > t, Y = t)$. Considering one of these components,

$$\begin{aligned}
\mathbb{P}(X = t, Y > t) &= \mathbb{P}(X = t)\mathbb{P}(Y > t) && \text{(by independence)} \\
&= \mathbb{P}(X = t)(1 - \mathbb{P}(Y \leq t)) \\
&= \left(\lambda e^{-\lambda t}\right)\left(1 - \left[1 - e^{-\mu t}\right]\right) \\
&= \lambda e^{-(\lambda+\mu)t}.
\end{aligned}$$

The same argument applies to the other component. Thus, by the law of total probability, we find

$$\begin{aligned}
\mathbb{P}(X = t | Z = t) = \frac{\mathbb{P}(X = t, Z = t)}{\mathbb{P}(Z = t)} &= \frac{\mathbb{P}(X = t, Y > t)}{\mathbb{P}(X = t, Y > t) + \mathbb{P}(X > t, Y = t)} \\
&= \frac{\lambda e^{-(\lambda+\mu)t}}{\lambda e^{-(\lambda+\mu)t} + \mu e^{-(\lambda+\mu)t}} \\
&= \frac{\lambda}{\lambda + \mu}.
\end{aligned}$$

Thus, we found that the probability of the minimum of exponentials resulting from one of its component exponentials is proportional to the rate of that component. □

Appendix for Chapter 4: Inference

Note, the following theorem only pertains to the discrete-time algorithm.

Theorem 4.1 (**Dirichlet-Multinomial Conjugacy**). *Suppose we have three k-dimensional vectors, $\vec{\kappa}$, $\vec{\alpha}$, and $\vec{c}$. If $\vec{\kappa} \sim \textsc{Dirichlet}(\vec{\alpha})$, and $\vec{c} \sim \textsc{Multinomial}(\vec{c}|\vec{\kappa})$, then $\vec{\kappa}|\vec{c} \sim \textsc{Dirichlet}(\vec{\alpha} + \vec{c})$.*

In other words, we wish to show that the Dirichlet prior and multinomial likelihood give rise to a joint posterior Dirichlet with the update rule $\alpha_i + c_i$ for all $i \in \{1, \ldots, k\}$. We know that Bayes's Law implies that the posterior is proportional to the likelihood and the prior,

$$\begin{aligned}\pi(\Theta|D) &= \frac{\pi(D|\Theta)\pi(\Theta)}{\int_\Theta \pi(D|\Theta)\pi(\Theta)\ d\Theta} \\ &\propto \pi(D|\Theta)\pi(\Theta).\end{aligned}$$

Thus, our goal is to verify this proportionality for the given distributions. Considering the RHS, we have

$$\begin{aligned}\pi(D|\Theta)\pi(\Theta) &= \pi(\vec{c}|\vec{\kappa}) \cdot \pi(\vec{\kappa}|\vec{\alpha}) \\ &= \left[\left(\sum_{i=1}^k c_i\right)! \prod_{i=1}^k \frac{\kappa_i^{c_i}}{c_i!}\right] \cdot \left(\frac{\Gamma(\sum_{i=1}^k \alpha_i)}{\prod_{i=1}^k \Gamma(\alpha_i)} \prod_{k=1}^k \kappa_i^{\alpha_i - 1}\right) \\ &= \underbrace{\left[\left(\sum_{i=1}^k\right)! \left(\frac{1}{\Gamma(\alpha_i)c_i!}\right) \Gamma\left(\sum_{i=1}^k \alpha_i\right)\right]}_{\text{constant with respect to } \vec{\kappa}} \cdot \prod_{i=1}^k \kappa_i^{\alpha_i + c_i - 1}\end{aligned}$$

We can then see that the joint posterior is of the form of a Dirichlet:

$$\begin{aligned}\pi(\vec{\kappa}|\vec{c}) &\propto \pi(\vec{c}|\vec{\kappa})\pi(\kappa|\vec{\alpha}) \\ \pi(\vec{\kappa}|\vec{c}) &\propto \prod_{i=1}^k \kappa_i^{\alpha_i + c_i - 1} = \prod_{i=1}^k \kappa_i^{\alpha_i' - 1},\end{aligned}$$

where $\alpha_i' = \alpha_i + c_i$ for all $i \in \{1, \ldots, k\}$. This demonstrates that $(\vec{\kappa}|\vec{c}) \sim \textsc{Diric}(\alpha_i')$, with the update rule of $\alpha_i' = \alpha_i + c_i$, as desired. □

Appendix for Chapter 5: Simulation Studies

The following are the parameter choices for the discrete-time simulation studies. Consider first the various regression vectors:

Description	$\beta_{\text{intercept}}$	β_{blue}	β_{EtOH}	β_{caff}
stable inference	-0.74	1.5	-1.5	0.02
low training	-2.5	-2.2	2.4	-2
medium-low training	-2	0.5	1.75	1.2
medium-high training	-2.4	4.8	2.7	0.3
high training	2.8	2.8	1.15	-2.9

Next, consider the various pairs of transition matrices. Notice, each graph represents 1000 seconds of flight duration, and the JSD represents the row-wise Jenson-Shannon distance between the two transition matrices.

The following table outlines the different parameter combinations used in the studies.

Parameter Index	β training	κ similarity
1	stable	stable
2	stable	high
3	stable	medium
4	stable	low
5	medium-low	stable
6	high	stable
7	low	stable
8	medium-high	stable

$$\begin{bmatrix} .892 & .108 & 0 \\ .02 & .921 & .059 \\ 0 & .108 & .892 \end{bmatrix} \qquad \begin{bmatrix} .596 & .404 & 0 \\ .009 & .94 & .051 \\ 0 & .404 & .596 \end{bmatrix}$$

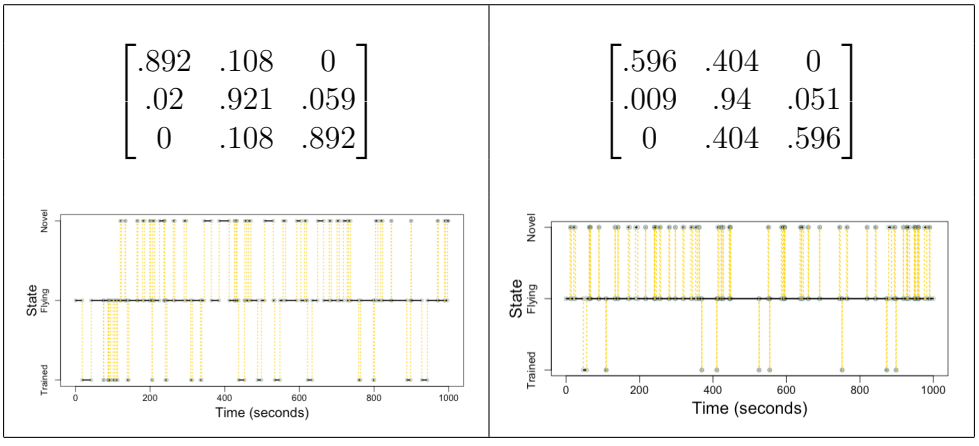


Table 7.1: Stable $\boldsymbol{\kappa}$'s. JSD = 0.1220.

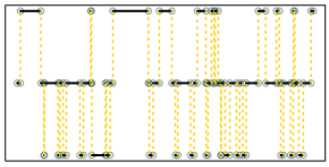
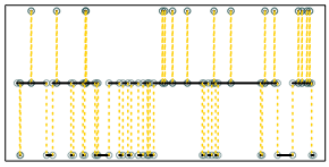
$\begin{bmatrix} .95 & .05 & 0 \\ .03 & .95 & .02 \\ 0 & .05 & .95 \end{bmatrix}$	$\begin{bmatrix} .95 & .05 & 0 \\ .02 & .95 & .03 \\ 0 & .95 & .05 \end{bmatrix}$
	

Table 7.2: High $\boldsymbol{\kappa}$ similarity. JSD = 0.4956.

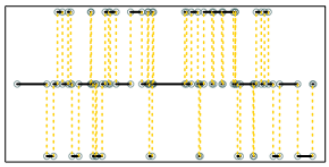
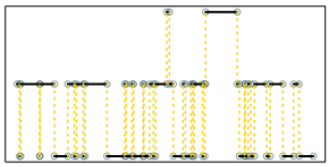
$\begin{bmatrix} .95 & .05 & 0 \\ .025 & .95 & .025 \\ 0 & .05 & .95 \end{bmatrix}$	$\begin{bmatrix} .95 & .05 & 0 \\ .045 & .95 & .005 \\ 0 & .05 & .95 \end{bmatrix}$
	

Table 7.3: Medium $\boldsymbol{\kappa}$ similarity. JSD = 0.0051.

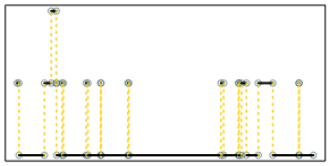
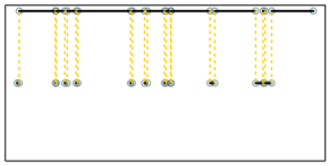
$\begin{bmatrix} .99 & .01 & 0 \\ .099 & .9 & .001 \\ 0 & .1 & .9 \end{bmatrix}$	$\begin{bmatrix} .9 & .1 & 0 \\ .001 & .9 & .099 \\ 0 & .01 & .99 \end{bmatrix}$
	

Table 7.4: Low $\boldsymbol{\kappa}$ similarity. JSD = 0.1086.

As for the continuous-time simulation studies, we used the stable $\vec{\beta}$ parameters for each (seeing as the $\vec{\lambda}$ and $\vec{Z}$ inference were the only new parts to focus on).

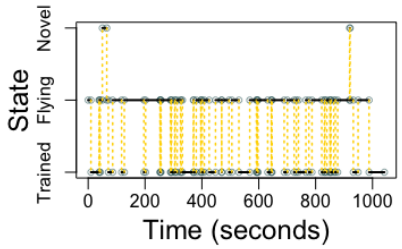
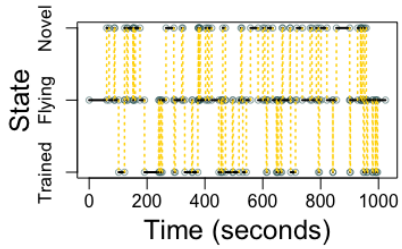
$\begin{bmatrix} -0.1 & 0.1 & 0 \\ 0.075 & -0.081 & 0.006 \\ 0 & 0.1 & -0.1 \end{bmatrix}$	$\begin{bmatrix} -0.1 & 0.1 & 0 \\ 0.05 & -0.1 & 0.05 \\ 0 & 0.1 & -0.1 \end{bmatrix}$
	

Table 7.5: Stable $\vec{\lambda}$ values. EUD = 0.0541.

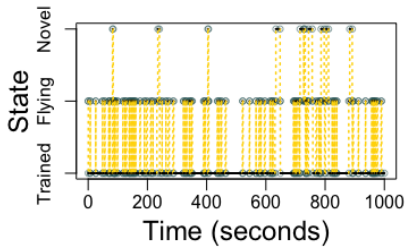
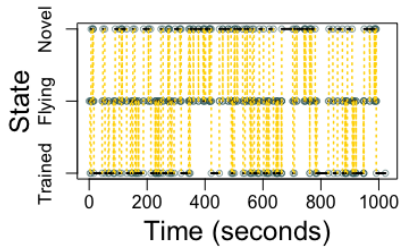
$\begin{bmatrix} -0.1 & 0.1 & 0 \\ 0.75 & -0.81 & 0.06 \\ 0 & 0.1 & -0.1 \end{bmatrix}$	$\begin{bmatrix} -0.1 & 0.1 & 0 \\ 0.5 & -1 & 0.5 \\ 0 & 0.1 & -0.1 \end{bmatrix}$
	

Table 7.6: High transition frequency $\vec{\lambda}$ values. EUD = 0.5406.

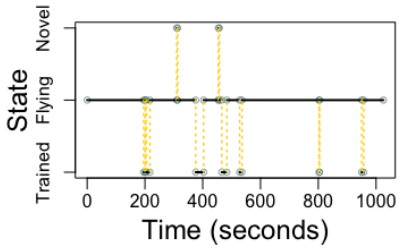
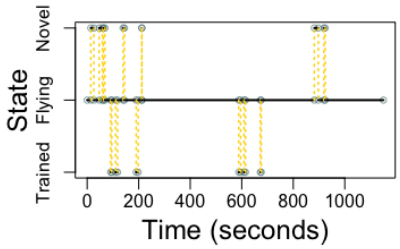
$\begin{bmatrix} -0.1 & 0.1 & 0 \\ 0.0075 & -0.0081 & 0.0006 \\ 0 & 0.1 & -0.1 \end{bmatrix}$	$\begin{bmatrix} -0.1 & 0.1 & 0 \\ 0.005 & -0.01 & 0.005 \\ 0 & 0.1 & -0.1 \end{bmatrix}$
	

Table 7.7: Low transition frequency $\vec{\lambda}$ values. EUD = 0.0054

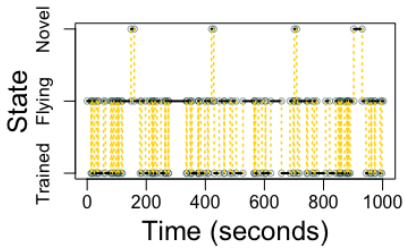
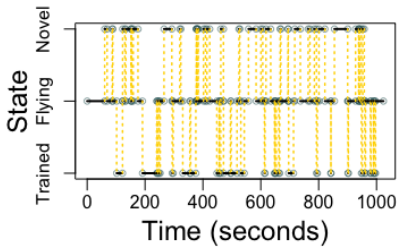
$\begin{bmatrix} -0.1 & 0.1 & 0 \\ 0.112 & -0.116 & 0.004 \\ 0 & 0.1 & -0.1 \end{bmatrix}$	$\begin{bmatrix} -0.1 & 0.1 & 0 \\ 0.05 & -0.1 & 0.05 \\ 0 & 0.1 & -0.1 \end{bmatrix}$
	

Table 7.8: Distinct transition probabilities $\vec{\lambda}$ values. EUD = 0.07884

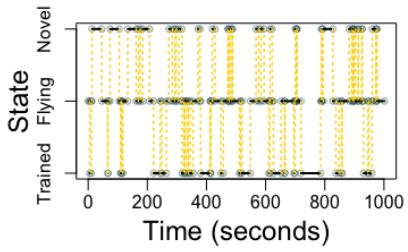
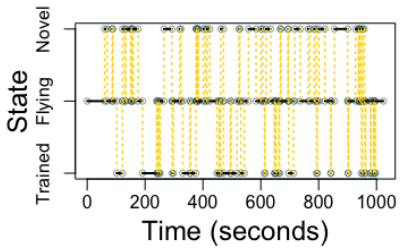
$\begin{bmatrix} -0.1 & 0.1 & 0 \\ 0.035 & -0.1 & 0.065 \\ 0 & 0.1 & -0.1 \end{bmatrix}$	$\begin{bmatrix} -0.1 & 0.1 & 0 \\ 0.05 & -0.1 & 0.05 \\ 0 & 0.1 & -0.1 \end{bmatrix}$
	

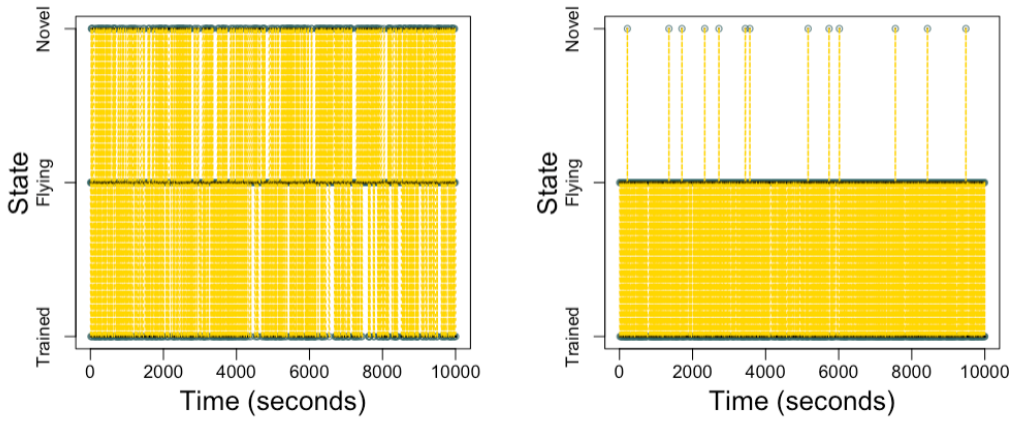
Table 7.9: Similar transition probabilities $\vec{\lambda}$ values. EUD = 0.0212

Appendix for Chapter 6: Data Analysis

Recall from Chapter 4 the discussion on the prior used in the CTM real data runs. The 10 chains (each with randomly sampled training bees) seemed to converge quite well on the following values:

$$\boldsymbol{Q}_U = \begin{bmatrix} -0.1567 & 0.1567 & 0 \\ 0.07821 & -0.1515 & 0.07333 \\ 0 & 0.1565 & -0.1565 \end{bmatrix}, \quad \boldsymbol{Q}_T = \begin{bmatrix} -0.1536 & 0.1536 & 0 \\ 0.1201 & -0.1216 & 0.001520 \\ 0 & 1.439 & -1.439 \end{bmatrix}.$$

These values correspond to the medians of medians: for each run, the median $\vec{\lambda}$ values were calculated, and here we have the median of those values. (The standard deviations were also calculated, and deemed encouraging enough to continue.) To understand the implications of these values, consider a graphical representation below.



Simulating two flight paths of 10,000 seconds each from the two matrices, we can approximate a graphical representation of training. The untrained status shows effectively identical time spent in either state, and the trained status shows a very strong preference for the trained state. Transitions to the novel state are still clearly possible, but very unlikely (consider the timescale).

To convert these into a prior, I devised shape and rate parameters of the gamma distribution such that these values were the means, and the curvature included nearby values. Recall the standard gamma distribution notation:

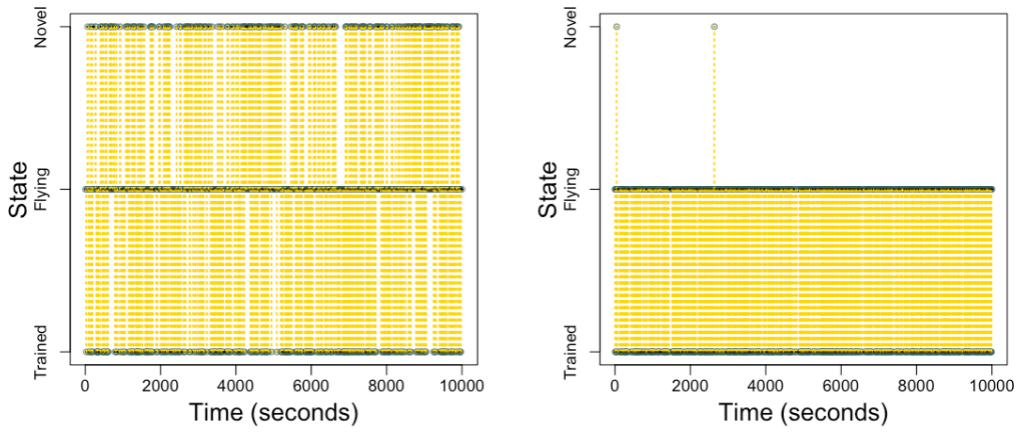
$$\text{rate} = \beta, \quad \text{shape} = \alpha,$$
$$\text{mean} = \frac{\alpha}{\beta}, \quad \text{variance} = \frac{\alpha}{\beta^2}.$$

As such, I fixed the rate to be 100, and solved for shape such that the mean matched the inferred values. Note, for the (2,3) entry of $\boldsymbol{Q}_T$ (which is far smaller than the others) I used a rate of 1,000, otherwise the parameter would collapse to zero in subsequent sampling. This procedure defined the prior distribution on each parameter.

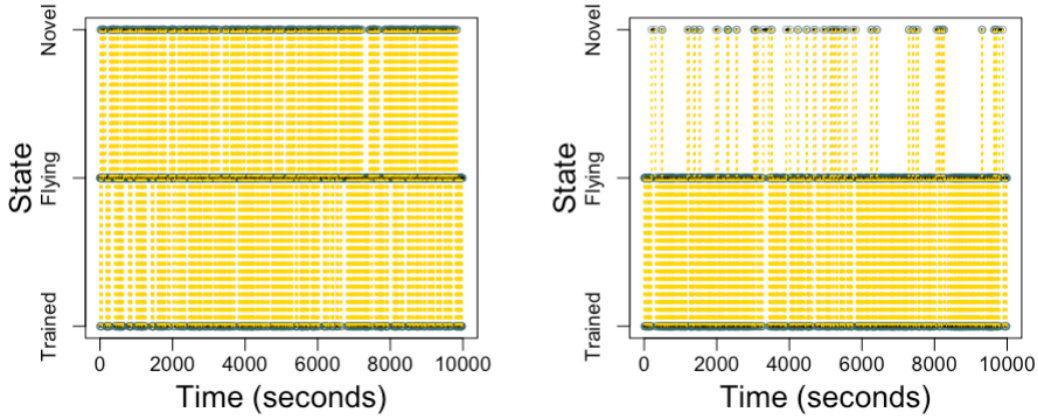
After full inference, we find the final $\vec{\lambda}$ to define the following rate matrices:

$$\hat{\boldsymbol{Q}}_U = \begin{bmatrix} -0.1137 & 0.1137 & 0 \\ 0.06921 & -0.1158 & 0.04660 \\ 0 & 0.1068 & -0.1068 \end{bmatrix}, \quad \hat{\boldsymbol{Q}}_T = \begin{bmatrix} -0.1120 & 0.1120 & 0 \\ 0.1001 & -0.1008 & 0.0007117 \\ 0 & 1.373 & -1.373 \end{bmatrix}.$$

Again, we can glimpse this visually by simulation.

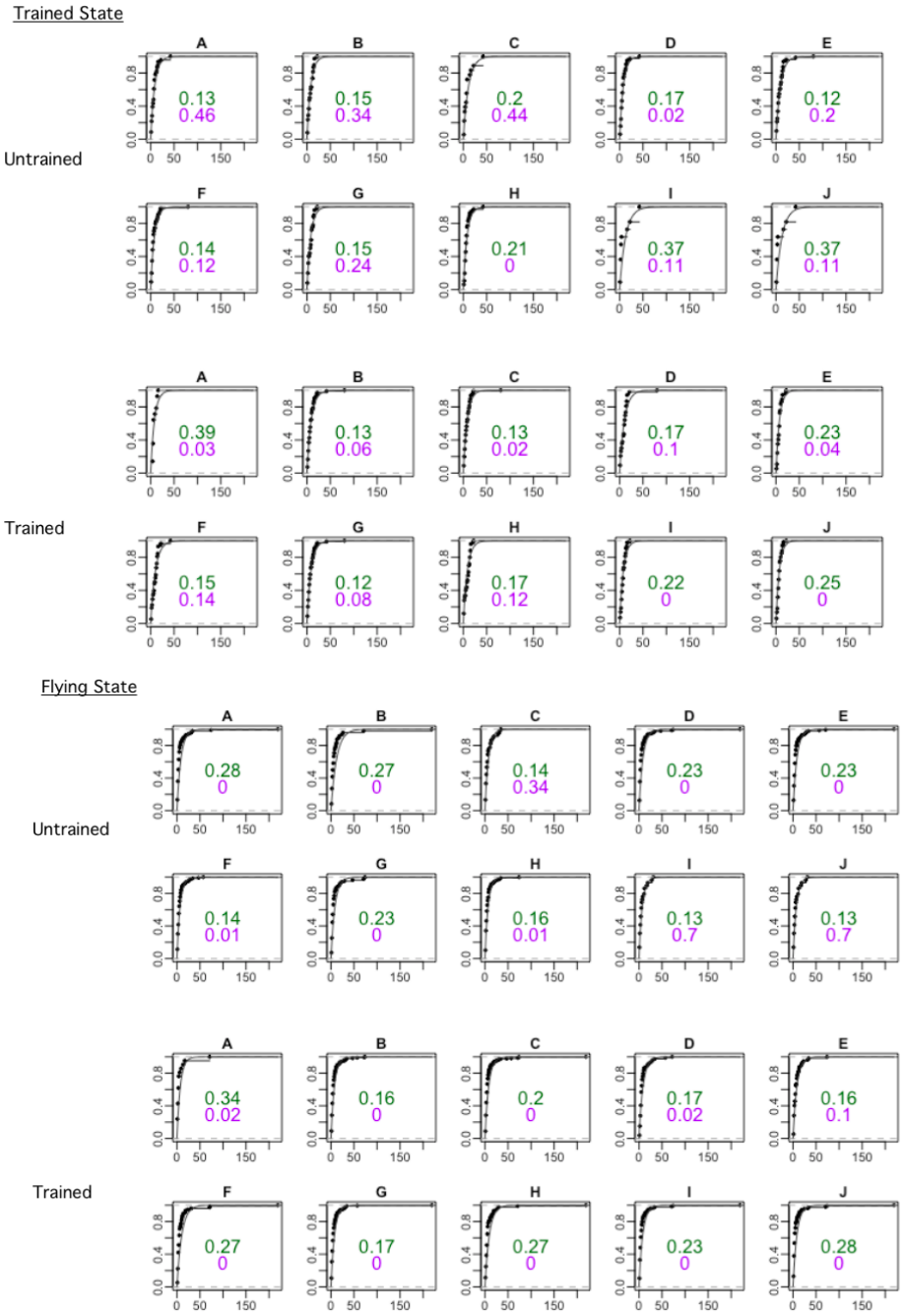


The algorithm appears to have doubled down on the infrequency of novel transitions in a trained bee. This helps explain why untrained bees are classified as such with such high certainty: any novel transition appears to offer very strong evidence against training. For context, compare these results to the $\boldsymbol{\kappa}$ implications from the DTM.



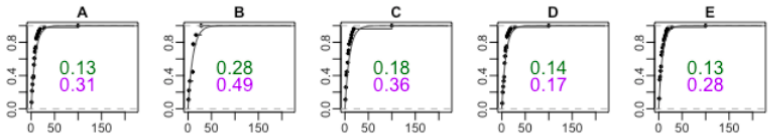
Clearly, the DTM trained state is far more tolerant to novel transitions than its CTM counterpart. Once again, this visual understanding of the rate and transition matrices sheds light on the DTM and CTM mixture inferences.

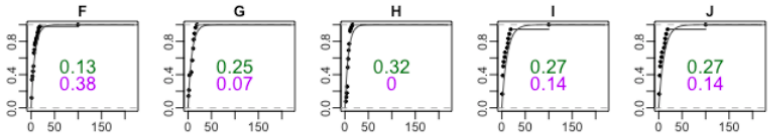
Shown below are the observed dwell-times fit to the curve of an exponential CDF (parameterized by MLE). The green, top number shows the KS-test D statistic, and the purple, bottom number shows the corresponding p-value.



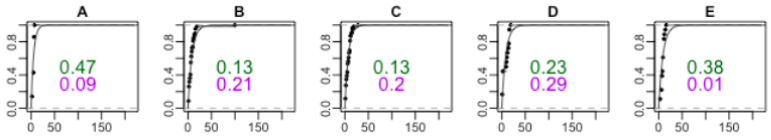
Novel State

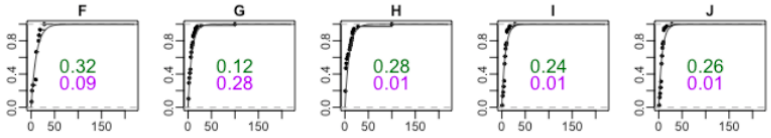
Untrained





Trained





Bibliography

1. Ross, S. M. *A First Course in Probability* Fifth. ISBN: 0137463146 9780137463145 013895772X 9780138957728 (Prentice Hall, Upper Saddle River, N.J., 1998).
2. Dobrow, R. P. *Introduction to stochastic processes with R* eng. ISBN: 1118740718 (John Wiley and Sons, Inc., Hoboken, New Jersey, 2016).
3. Sigman, K. *Lecture notes on Stochastic Modeling I* 2009. http://www.ieor.columbia.edu/~sigman/stochastic-I.html.
4. Whitt, W. Continuous-time Markov chains. *Dept. of Industrial Engineering and Operations Research, Columbia University, New York* (2006).
5. Hitchcock, D. B. A History of the Metropolis-Hastings Algorithm. *The American Statistician* **57,** 254–257. ISSN: 00031305. http://www.jstor.org/stable/30037292 (2003).
6. Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H. & Teller, E. Equation of State Calculations by Fast Computing Machines. *The Journal of Chemical Physics* **21,** 1087–1092. https://doi.org/10.1063/1.1699114 (1953).
7. Hastings, W. K. Monte Carlo Sampling Methods Using Markov Chains and Their Applications. *Biometrika* **57,** 97–109. ISSN: 00063444. http://www.jstor.org/stable/2334940 (1970).
8. Gundersen, G. Feb. 2020. http://gregorygundersen.com/blog/2020/02/23/gibbs-sampling/.
9. Mathis, A. *et al.* Markerless tracking of user-defined features with deep learning. *CoRR* **abs/1804.03142.** arXiv: 1804.03142. http://arxiv.org/abs/1804.03142 (2018).
10. *Introduction to Generalized Linear Mixed Models* https://stats.idre.ucla.edu/other/mult-pkg/introduction-to-generalized-linear-mixed-models/.
11. Mcculloch, C. E. & Neuhaus, J. M. in *Wiley StatsRef: Statistics Reference Online* (American Cancer Society, 2014). ISBN: 9781118445112. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/9781118445112.stat07540. https://onlinelibrary.wiley.com/doi/abs/10.1002/9781118445112.stat07540.
12. Visual acuity of the honey bee retina and the limits for feature detection. *Scientific Reports* **7,** 45972. https://doi.org/10.1038/srep45972 (2017).
13. Osborne, J. L. *et al.* Bumblebee flight distances in relation to the forage landscape. *Journal of Animal Ecology* **77,** 406–415. eprint: https://besjournals.onlinelibrary.wiley.com/doi/pdf/10.1111/j.1365-2656.2007.01333.x. https://besjournals.onlinelibrary.wiley.com/doi/abs/10.1111/j.1365-2656.2007.01333.x (2008).
14. R Core Team. *R: A Language and Environment for Statistical Computing* R Foundation for Statistical Computing (Vienna, Austria, 2019). https://www.R-project.org/.
15. Wickham, H. *et al.* Welcome to the tidyverse. *Journal of Open Source Software* **4,** 1686 (2019).

16. Jasra, A., Holmes, C. C. & Stephens, D. A. Markov Chain Monte Carlo Methods and the Label Switching Problem in Bayesian Mixture Modeling. *Statistical Science* **20,** 50–67. https://doi.org/10.1214/088342305000000016 (2005).

17. Hoff, P. D. *A First Course in Bayesian Statistical Methods* 1st. ISBN: 0387922997 (Springer Publishing Company, Incorporated, 2009).

18. Endres, D. M. & Schindelin, J. E. A new metric for probability distributions. *IEEE Transactions on Information Theory* **49,** 1858–1860 (2003).

19. Gelman, A., Carlin, J. B., Stern, H. S. & Rubin, D. B. *Bayesian Data Analysis* 2nd ed. (Chapman and Hall/CRC, 2004).

20. Meng, X.-L. Posterior Predictive p-Values. *The Annals of Statistics* **22,** 1142–1160. ISSN: 00905364. http://www.jstor.org/stable/2242219 (1994).

21. Massey, F. J. The Kolmogorov-Smirnov Test for Goodness of Fit. *Journal of the American Statistical Association* **46,** 68–78. ISSN: 01621459. http://www.jstor.org/stable/2280095 (1951).

22. Betancourt, M. *A Conceptual Introduction to Hamiltonian Monte Carlo* 2018. arXiv: 1701.02434 [stat.ME].

23. Wong, A. K. C. & You, M. Entropy and Distance of Random Graphs with Application to Structural Pattern Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **PAMI-7,** 599–609 (1985).

24. Joe Blitzstein. *Lecture 17: Moment Generating Functions. Harvard Stat 110* https://www.youtube.com/watch?v=N8O6zd6vTZ8&t=586s. Online; accessed 14 April 2021. 2006.