MIXTURES OF EXPONENTIAL DISTRIBUTIONS TO DESCRIBE THE DISTRIBUTION OF POISSON MEANS IN ESTIMATING THE NUMBER OF UNOBSERVED CLASSES

A Thesis

Presented to the Faculty of the Graduate School

of Cornell University

in Partial Fulfillment of the Requirements for the Degree of

Master of Science

by

Kathryn Jo-Anne Barger

May 2006

ABSTRACT

In many fields of study scientists are interested in estimating the number of unobserved classes. A biologist may want to find the number of rare species of an animal population in order to conserve, manage, and monitor biodiversity; a library manager may want to know how many non-circulating items are present in a library system; or a clinical investigator may want to determine the number of unseen disease occurrences. A traditional way of estimating an unknown number of classes is by using a negative binomial model (Fisher, Corbet, and Williams 1943). The negative binomial model is based on assuming that the numbers of individuals from each class are independent Poisson samples, and that the means of these Poisson random variables follow a Gamma distribution. This thesis proposes a parametric model where the law of the mean frequency of classes is a finite mixture of exponential distributions. The proposed model has the following advantages: model simplicity, efficient computation using the EM algorithm, and a straightforward interpretation of the fitted model. Also, model assessment by way of a chi-squared goodness of fit procedure can be used, a benefit this parametric model has over other commonly used non-parametric methods.

A main accomplishment of this thesis is providing an efficient computation of maximum likelihood (ML) estimates for the proposed model. Without use of the EM algorithm, finding ML estimates for this model can be difficult and time consuming. The likelihood function is complicated due to high dimensionality and non-identifiability of certain parameters. Within the M step of our algorithm we embed another EM, which can effortlessly maximize the parameters in the finite mixture. We refer to the algorithm as a nested EM. Aitken's

acceleration is used to increase speed of the algorithm.

Microbial samples from the coast of Massachusetts Bay near Nahant, Massachusetts are used to demonstrate data analysis using three different numbers of components in the finite mixture of the model described. It is shown that the model produces reasonable estimates and fits the data satisfactorily. This model has recently been premiered in species richness estimation (Hong *et al.* 2006), and its many advantages show promise for continued use in estimating the number of unobserved classes.

keywords: EM algorithm, finite mixtures, species richness, Aitken's acceleration, microorganisms

BIOGRAPHICAL SKETCH

Kathryn Barger graduated from Carmel High School in Carmel, NY in 1999. She received a Bachelor of Music in violin performance with a second major in mathematics from the State University of New York at Potsdam in 2003.

ACKNOWLEDGMENTS

I would like to thank Cornell University and the Department of Statistical Science for their financial support. I would like to the Cornell Theory Center for their computing resources, especially Linda Woodard for being so helpful and patient. I would also like to thank Slava S. Epstein and all of his associates at the Epstein lab, Marine Science Center, Northeastern University, for the use of their extraordinary data sets. I would like to thank the members of my thesis committee, Dr. Martin Wells and Dr. John Bunge, for dedicating their time in helping me complete my thesis. Finally, I extend sincere appreciation to my thesis advisor, Dr. John Bunge, for all of his assistance and encouragement in my graduate education leading to the writing of this thesis.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1

Introduction: How Many Kinds are There?

Imagine a large population divided into a number of classes. Each member of the population, when polled, is identified with one class. However, the number of existing classes is unknown. It is not possible to survey the entire population; yet we wish to estimate the total number of classes. We must make a guess and try to estimate the number of classes based on an observed sample. Statistics can then be used in order to answer the research question: How many kinds are there?

There are many examples of populations for which researchers are attempting to answer this question. Below are some examples which reveal the varied fields in which this question arises.

1.1 Motivating Examples

In the vast field of biology, the estimation of species richness is a widely studied area. Species richness is used to describe the number of species which reside in a certain biosphere or belong to a particular population. Knowing the number of species can help determine the complexity of an ecosystem. A high level

of species richness can help identify undersampled populations. Measurements over time can help determine the number of rare or extinct species. Knowing the species richness of a population helps biologists understand biodiversity. Biodiversity is related to many other studies in environmental science and ecology. Interest in the number of species is not restricted to specialized fields in biology. Biologists are studying populations of birds (Borgella and Gavin 2005 [5]), spiders (Scharff *et al.* 2003 [28]), fish (Smith and Jones 2005 [29]), flies (Folgarait *et al.* 2005 [18]), bats (Sampaio *et al.* 2003 [26]), fossils (Cobabe and Allmon 1994 [11]), trees (Cannon, Peart, and Leighton 1998 [9]), and bacteria (Hong *et al.* 2006 [19]; Dunbar *et al.* 2002 [13]), in an attempt to assess levels of biodiversity. Due to the numerous biological applications of estimating the number of unobserved classes, the problem is often named the "species problem".

A problem in numismatics is to estimate the size of an ancient coin collection (Stam 1987 [31]). Esty (1986 [15]) describes that numismatists are interested in the original number of coins in an issue, and also the number of dies used to produce an ancient coinage. We can think of a die as a type of coin in the collection. The number of dies found in a sample from an ancient coin issue is used to determine the total number of dies in that coinage, which can be used to estimate the total number of coins in the issue. The size of a coin issue reveals important information regarding the economy of the civilization from which it came.

Estimating the size of an author's vocabulary can also be related to estimating the number of classes. Efron and Thisted (1976 [14]) estimate the number of words in Shakespeare's vocabulary based on word frequencies from his published material. Knowing the size of the vocabulary tells us the number of words Shakespeare knew but did not use. Also, suppose a new work by Shakespeare

is discovered. We can then estimate the number of new words that we expect to find from this source. A linguist may also be interested in comparing the vocabulary size of several authors (Booth 1967 [4]).

Maintaining databases is an area of research related to estimating the number of classes. For example, when a user is searching for an journal article, it is not rare to return several hits on a search query even if the library holds only a single copy of the article. The multiple hits are all for the same published material, however multiple records are on file. These duplicate records are not identified due to small differences, such as a misspelling or an abbreviation in one of the fields. The library database manager wants to know how many duplicate records are in the database, for the size of the database based on the number of records is usually an overestimate of the material contained in the library. This problem reduces to estimating the number of classes. In general, database managers may wish to estimate the number of categories in the database. This is described as estimating the size of projections which is discussed by Olken and Rotem (1995 [24]).

The problem of estimating the number of classes also arises in library systems management. Managers want to be informed of the usage of materials in the library and how much material is not being circulated (Burrell 1988 [8]). Information about which materials are borrowed, and the frequency in which they are loaned out can be used to estimate the amount of non-circulating material.

During the debugging process of computer software, a tally of the number of errors recorded can be used to estimate the number of total faults in the system (Lloyd, Yip, and Chan 1999 [22]). Errors may or may not all be equally likely to be detected. Certain resampling and debugging patterns can be used to estimate the frequency of errors in the system. This information is used to

estimate the number of faults in the entire program.

Ding (1996 [12]) uses AIDS incidence and HIV infection data in order to estimate the size of the AIDS epidemic. Brookmeyer and Gail (1988 [6]) estimate the number of unreported diseases, as well as infection rate of AIDS.

To estimate the size of a illicit drug-using population, Hser (1993 [20]) uses data from drug treatment centers and arrest records. This data is considered a sample from the population of treatment-susceptible criminal users. This population is of interest for its health implications and criminal effect on the larger population. Estimating the size of the drug-using population helps in the planning of institutions invested in helping those with drug problems.

These are just a few of the many examples of estimating the number of unobserved classes. A bibliography of related references exist and are maintained by Dr. John Bunge, Associate Professor, Cornell University. This list of references can be accessed at http://www.stat.cornell.edu/∼bunge/bibliography.

These research problems all ask a common question: How many kinds are there? Whether this question is targeted at spider species, coin dies, or types of words known to an author, it can be answered using the model proposed in the following chapters.

1.2 Modeling the Observed Data

Our sample consists of individuals, each of whom can be identified with a class. In estimating species richness, we classify the individuals into their respective species. Then we can see which classes are the most abundant, which classes are the most rare, and what the frequencies of all class sizes look like. We base the model proposed in this paper on the Poisson distribution. The next sections

introduce the model notation and sampling scheme for the data.

1.2.1 Data Structure and Notation

Consider a population divided into C classes, where $C < \infty$. Suppose we have obtained a sample of n individuals from this population and the value of C is known. Assuming each individual can be identified with only one class, then all of the individuals in the sample can be sorted into their respective classes. Let c be the number of unique classes that have appeared in the sample.

Define Y_j to be the number of individuals in the sample associated with class j where $j = 1, 2, \ldots, C$. We can describe Y_j as a Poisson random variable with mean $\lambda_j > 0$. The probability density function for Y_j is

$$\begin{aligned} p(y_j) &= P[Y_j = y_j] \\ &= \frac{e^{-\lambda_j} \lambda_j^{y_j}}{y_j!}. \end{aligned} \tag{1.1}$$

We assume the Y_j are independent for all j. We will call $\mathbf{Y} = (Y_1, Y_2, \ldots, Y_C)'$ the *non-truncated* data vector.

Notice that $\mathbf{Y}$ has non-negative integer elements. This means zero is a possible value in $\mathbf{Y}$. If $Y_j = 0$, then no individuals from class j were observed in the sample. This feature distinguished the *non truncated* data vector from the *truncated* or *observed* data vector.

1.2.2 Zero-Truncated Poisson Distribution

We will now assume that C is unknown. This means when class j is not found in the sample, we do not know the class exists.

The *truncated* data for each class is X_{i_j}, where $i = 1, 2, \ldots, c$ and c is the number of unique classes observed in the sample. We observe a subset, $\{X_{i_j}\} \subseteq$

$\{Y_j\}$ such that $Y_j > 0$. In other words, $X_{i_j} = Y_j$ if $Y_j > 0$. Denote the truncated (or, more specifically, *zero-truncated*) data vector as $\mathbf{X} = (X_1, X_2, \ldots, X_c)'$. We will drop the double subscript when we do not need to refer to the non-truncated data. Then X_i for $i = 1, \ldots, c$ will represent the observed data.

By properties of conditional probability, we have

$$\begin{aligned} P[X_{i_j} = x_{i_j}] &= P[Y_j = x_{i_j} | Y_j > 0] \\ &= \frac{P[Y_j = x_{i_j}, Y_j > 0]}{P[Y_j > 0]} \\ &= \frac{P[Y_j = x_{i_j}]}{1 - P[Y_j = 0]}, \end{aligned} \tag{1.2}$$

where $x_{i_j} = 1, 2, \ldots,$ and we call the distribution of X_{i_j} a zero-truncated Poisson distribution.

1.2.3 Example: Microbial Species Richness

The data from eight microbial samples will be used to illustrate the model fitting procedure. Each sample contains 1734 total individuals sorted into classes based on eight different methods of classification. We will call each class a species in order to identify with biological examples. A unique species (formally called an operational taxonomic unit) is defined by a 70, 80, 90, 95, 96, 97, 98, and 99% sequence similarity cut-off percentage determined by analyzing 16S rRNA gene sequences. For details on the data generation technique see Hong *et al.* (2006 [19]).

Figure 1.1 shows an example of the observed data for the microbial data at cut-off percentage 99.

Due to difficulties in modeling frequency data with a long right tail, as seen in Figure 1.1 one can split the sample into "rare" and "abundant" species. Rare species are considered the set of X_i such that $X_i \leq \tau$, where τ is fixed prior

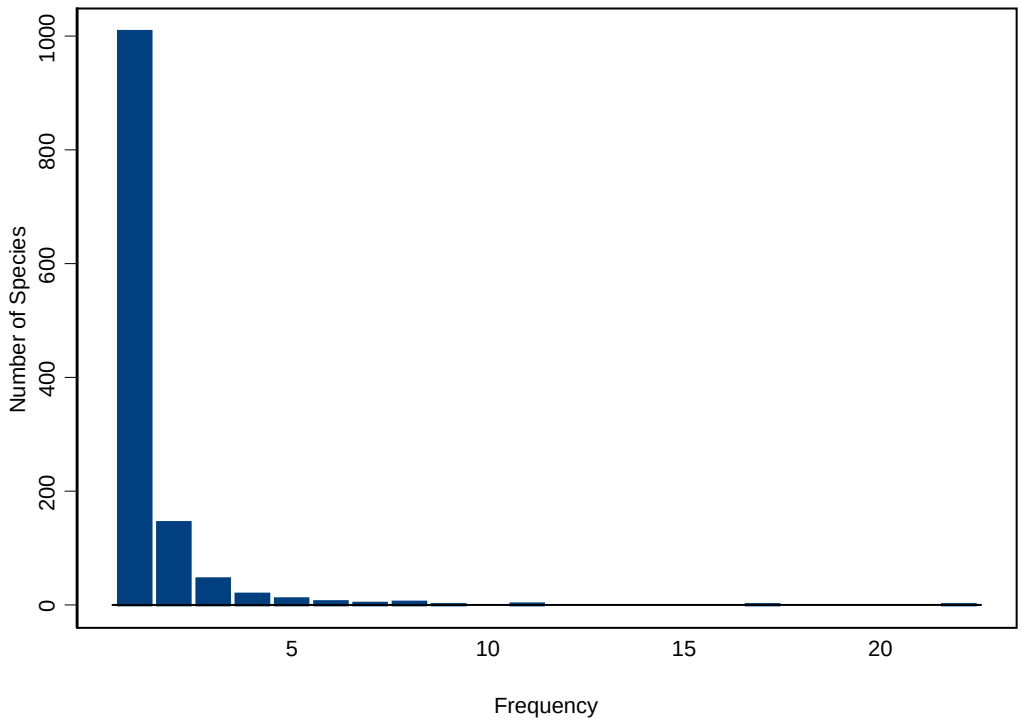

Figure 1.1: Microbial Data (99% cut-off)

to the analysis. In this paper, the choice of τ is important in model selection. Wang and Lindsay (2005 [33]) consider τ a tuning parameter in a nonparametric maximum likelihood approach.

For the microbial data, a higher sequence similarity cut-off is a more stringent rule for determining if two individuals are classified as the same species. Thus, as the cut-off percentage increases, individuals are classified into a larger number of species. The observed data has a shorter right tail and a larger number of singletons ($\sum I\{X_i = 1\}$). Figures 1.2, 1.3, 1.4, 1.5, 1.6, 1.7, and 1.8 show 70, 80, 90, 95, 96, 97, and 98% cut-offs, respectively, for the microbial data, and illustrate how the observed data changes as the cut-off value increases.

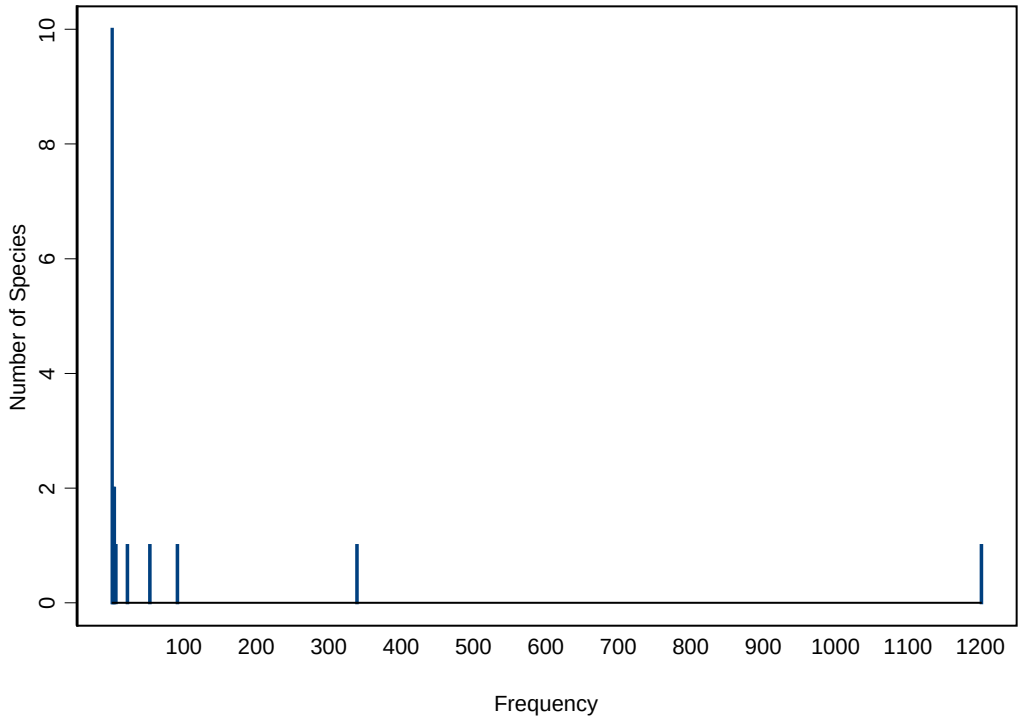

Figure 1.2: Microbial Data (70% cut-off)

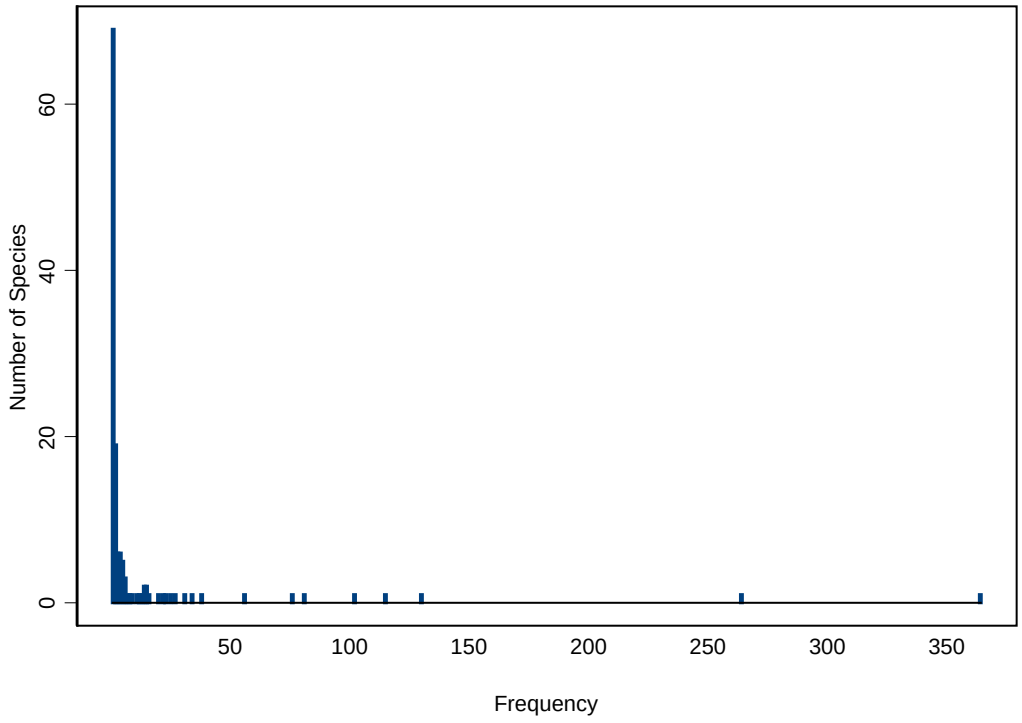

Figure 1.3: Microbial Data (80% cut-off)

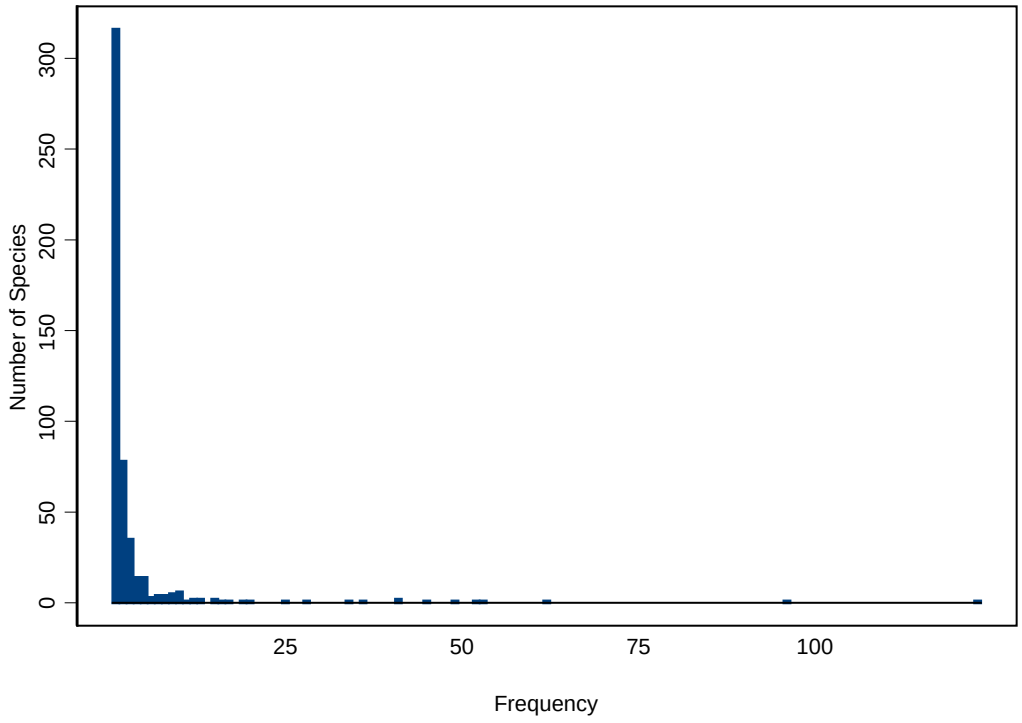

Figure 1.4: Microbial Data (90% cut-off)

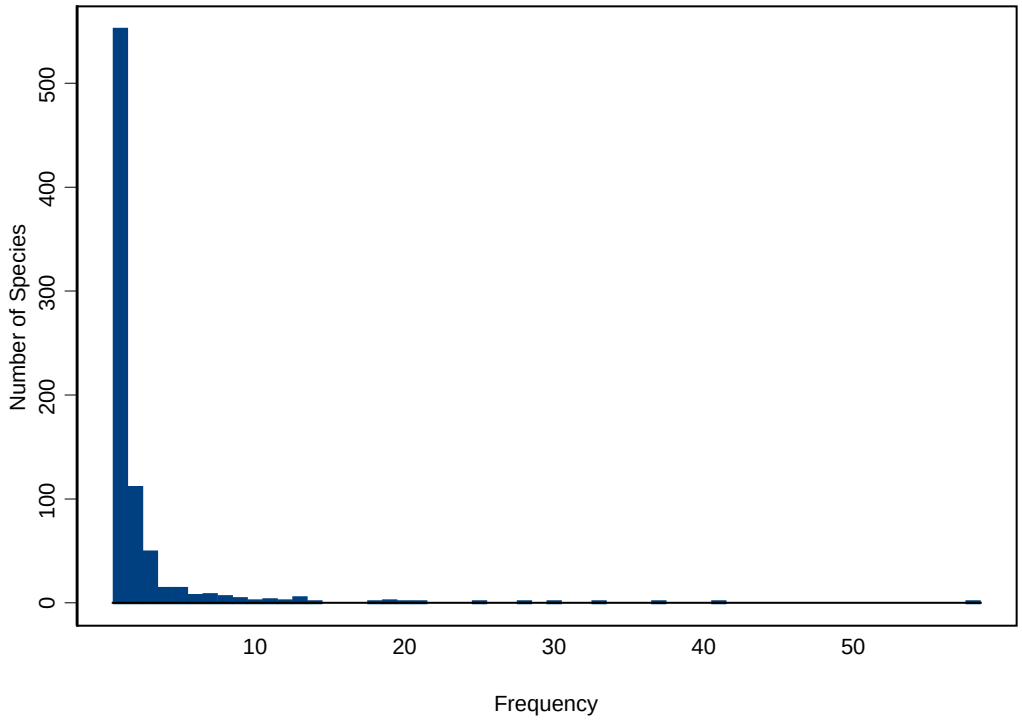

Figure 1.5: Microbial Data (95% cut-off)

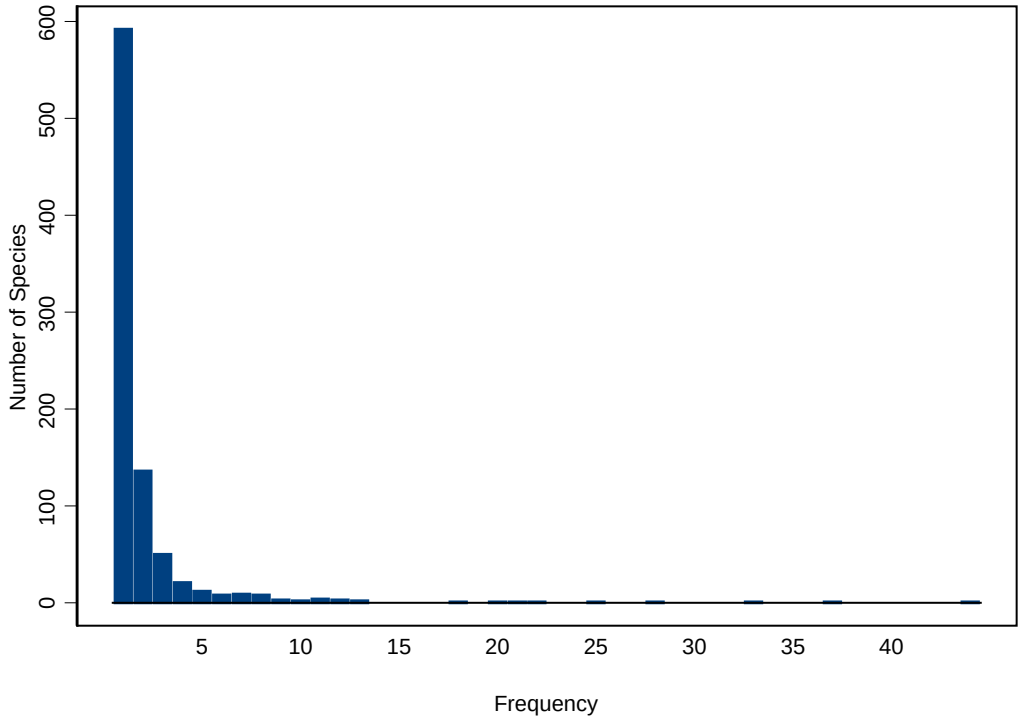

Figure 1.6: Microbial Data (96% cut-off)

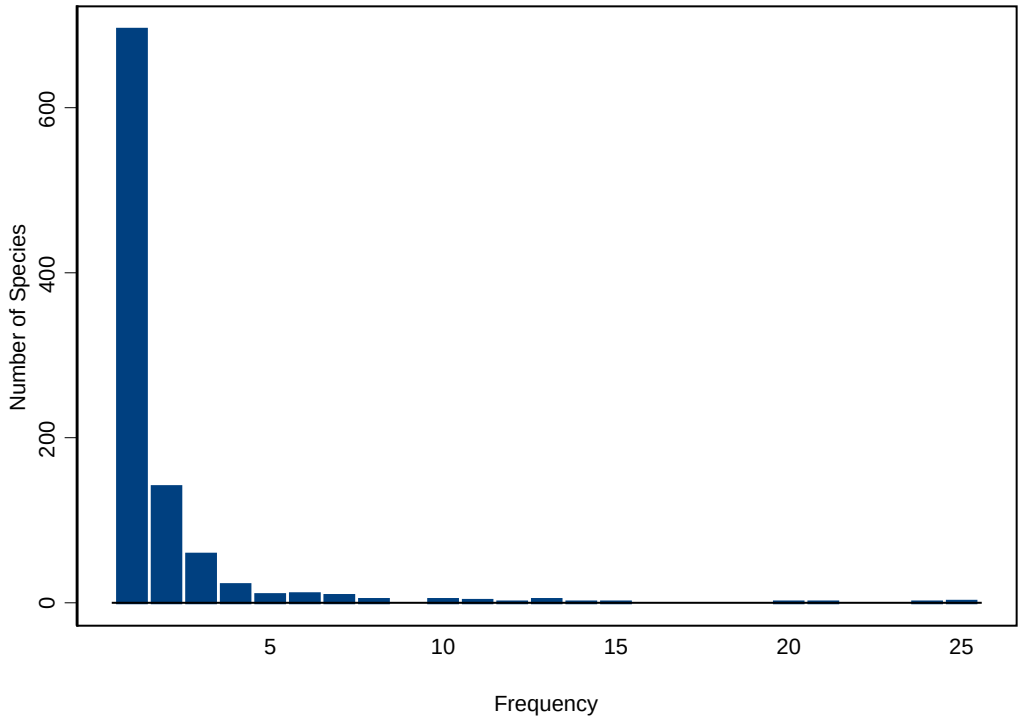

Figure 1.7: Microbial Data (97% cut-off)

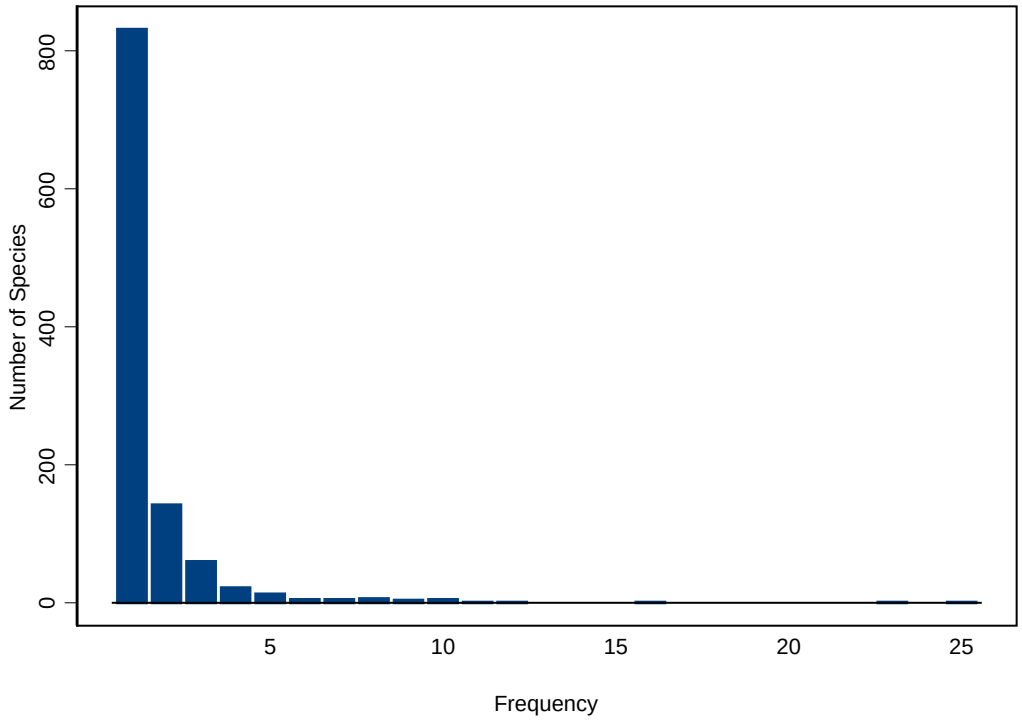

Figure 1.8: Microbial Data (98% cut-off)

1.3 A Hierarchical Model

The mean number of individuals sampled from a class j, $\lambda_j = \mathbb{E}[Y_j]$, can be considered a realization of a random variable with distribution function, F. Assuming $\lambda_j \sim F$ for all j, the resulting marginal distribution of Y_j is often referred to as a F-mixed Poisson distribution. We next discuss possible distributions that can be used for the Poisson means.

1.3.1 The Negative Binomial Model

First, we explain in detail the model proposed by Fisher, Corbet, and Williams (1943 [17]). The authors use this model to estimate species richness. In general, the Negative Binomial distribution is often used to model widely dispersed Poisson data.

Consider Y_j which is the non zero-truncated data for species $j = 1, 2, \ldots, C$, and is a Poisson random variable with mean λ_j which represents the number of individuals from species j in the sample. We assume Y_j are independent for all j.

Suppose that the classes are not equally abundant; that is, $\lambda_j \neq \lambda_k$ for all $j \neq k$. Differences in the means are allowed by assuming the λ_j follow a Gamma distribution given by the following density function with $\alpha, \beta > 0$.

$$p(\lambda_j) = \frac{1}{\Gamma(\alpha)\beta^{\alpha}} \lambda_j^{\alpha-1} e^{-\lambda_j/\beta}. \tag{1.3}$$

We assume that λ_j are independent for all j.

Using equation 1.1 and 1.3, we find the marginal distribution of Y_j by integrating with respect to λ_j. Then Y_j has a negative binomial distribution given by the density function

$$p(y_j) = \frac{\Gamma(\alpha + y_j)}{\Gamma(\alpha)\Gamma(y_j + 1)} \frac{\beta^{y_j}}{(1+\beta)^{\alpha+y_j}}. \tag{1.4}$$

It turns out that our simplest model is a special case of this model.

In practice the Negative Binomial model fails to fit many data sets especially those with a large number of singletons, such as in the microbial data.

1.3.2 Distributions for Poisson Means

The simplest structure for the $\lambda_j's$ is to assume all the means are equal, i.e. $\lambda_j \equiv \lambda$ for all j. However, from observed data as in figure 1.5 this may be an unreasonable assumption in many cases.

The Gamma distribution used in the last section is just one of many choices for a distribution on λ_j. Since λ_j is a nonnegative value, any distribution with support on $(0, \infty)$ is a reasonable choice.

This problem is motivated by the scientific question of estimating the number of unobserved classes. In the realm of parametric models, the Negative Binomial model often fails to fit the data, and a more flexible distribution is sought out. Choosing a more flexible distribution for the λ_j may increase the fit of the model and assuage issues such as choosing an appropriate value for τ.

The basic idea of this thesis is to model the full data as a collection of independent Poisson samples where the means of the Poisson samples follow a mixture of Exponential distributions. The zero-truncated mixed-Exponential mixed Poisson becomes the model for our observed data. We have chosen to use the Exponential distribution for its simplicity. See section 5.2 for discussion.

Chapter 2 introduces the mixed Poisson model with a single Exponential distribution. This is a special case of the Negative Binomial model described in section 1.3.1. The model for the observed data is a zero-truncated mixture of Geometric distributions. Chapter 3 introduces the use of a mixture of two Exponential distributions. Here, we implement a nested EM algorithm to find

the maximum likelihood estimates instead of using the analytic approach in chapter 2. Chapter 4 uses a mixture of three exponentials for the Poisson means. The same nested EM algorithm is used, and slow convergence is expedited by the use of Aitken's acceleration. Chapter 5 presents additional topics which are related to this model and the species problem.

Chapter 2

A Single Exponential Distribution

This chapter begins by introducing the Exponential mixed Poisson distribution in modeling the non-truncated data. The Exponential mixed Poisson distribution is a simple place to start since it is probabilistically equivalent to a Geometric distribution. The model for the observed data becomes a zero-truncated Geometric distribution.

2.1 An Empirical Bayes Model

Consider Y_j, the number of individuals observed from species j, where $j = 1, \ldots, C$ and C is the total number of species. Let $Y_j | \lambda_j \sim Poisson(\lambda_j)$, independently for each j. Now let $\lambda_j \sim Exponential(\theta)$, where θ is the mean of the distribution, i.e. if $p(\lambda_j | \theta)$ represents the probability density for λ_j, then

$$p(\lambda_j | \theta) = \frac{1}{\theta} e^{-\frac{\lambda_j}{\theta}}.$$

We assume that the λ_j are independent for all j.

This probability structure implies that the marginal distribution of Y_j is

$Geometric\left((1+\theta)^{-1}\right)$, that is,

$$\begin{aligned} p\left(y_j\right) &= \int p\left(y_j|\lambda_j\right) p\left(\lambda_j\right) d\lambda_j \\ &= \int \frac{e^{-\lambda_j}\lambda_j^{y_j}}{y_j!}\frac{1}{\theta}\, e^{-\frac{\lambda_j}{\theta}} d\lambda_j \\ &= \frac{1}{\theta y_j!}\int \lambda_j^{y_j} e^{-\lambda_j(1+\theta^{-1})} d\lambda_j \\ &= \frac{1}{1+\theta}\left(\frac{\theta}{1+\theta}\right)^{y_j}, \end{aligned} \tag{2.1}$$

for $y_j = 0, 1, \ldots$. In the above parameterization, $\frac{1}{1+\theta}$ is commonly associated with the probability of success of a Geometrically distributed random variable.

Next we define the model for the observed data. Using equation 1.2, the zero-truncated Geometric distribution is

$$\begin{aligned} P[X_{i_j} = x_{i_j}] &= P[Y_j = x_{i_j} | Y_j > 0] \\ &= \frac{\frac{1}{1+\theta}\left(\frac{\theta}{1+\theta}\right)^{x_{i_j}}}{1-\frac{1}{1+\theta}} \\ &= \left(\frac{\theta}{1+\theta}\right)^{x_{i_j}-1}\frac{1}{1+\theta}, \end{aligned} \tag{2.2}$$

where $x_{i_j} = 1, 2, \ldots$. This is the model as described in equation 1.4 with $\alpha = 1$ and $\beta = \theta$. Thus, $X_{i_j} - 1$ is also a Geometric random variable where $\frac{1}{1+\theta}$ is the probability of success.

2.2 Estimation and Inference

2.2.1 Maximum Likelihood Estimation

Using the zero-truncated Geometric distribution in equation 2.2 to model the observed data, we find parameter estimates by maximum likelihood. Assuming

X_i are independent for all i, the likelihood is

$$\begin{aligned} L(\theta) &= \prod_{i=1}^{c} \left(\frac{\theta}{1+\theta}\right)^{x_i-1} \frac{1}{1+\theta} \\ &= \left(\frac{\theta}{1+\theta}\right)^{\sum_{i=1}^{c}(x_i-1)} \left(\frac{1}{1+\theta}\right)^{c}. \end{aligned}$$

Maximizing with respect to θ gives us the maximum likelihood estimate

$$\hat{\theta} = \bar{x} - 1, \tag{2.3}$$

where $\bar{x} = \frac{1}{c}\sum_{i=1}^{c} x_i$.

Now we estimate the number of unobserved classes. Let $C_0 = C - c$ denote the number of unobserved classes. We estimate C_0 from the model by setting up the equation

$$\begin{aligned} C_0 &= C p_0(\hat{\theta}) \\ &= (C_0 + c)\, p_0(\hat{\theta}). \end{aligned}$$

Thus, the estimate of the number of unobserved classes, $\hat{C}_0$ is

$$\hat{C}_0 = \frac{c\, p_0(\hat{\theta})}{1 - p_0(\hat{\theta})}. \tag{2.4}$$

For the single Exponential model, using equations 2.1 and 2.3 we find

$$\begin{aligned} \hat{C}_0 &= \frac{c\frac{1}{1+\hat{\theta}}}{1 - \frac{1}{1+\hat{\theta}}} \\ &= \frac{c}{\hat{\theta}} \\ &= \frac{c}{\bar{x} - 1}. \end{aligned}$$

2.2.2 Standard Errors

We use a standard asymptotic approach developed by Sanathanan (1972 [27]) to calculate the variance of our estimate in equation 2.4. As $C \to \infty$,

$$\frac{\hat{C} - C}{\sqrt{C}} \Rightarrow^{d} Normal\left[0, \left[\frac{1 - p_0(\theta)}{p_0(\theta)} - \frac{1}{p_0^2(\theta)} \left(\frac{dp\,(0)}{d\theta}\right)^{T} I\,(\theta)^{-1} \left(\frac{dp\,(0)}{d\theta}\right)\right]^{-1}\right],$$

where $\left(\frac{dp(0)}{d\theta}\right)$ is the score function evaluated at $x = 0$ and $I(\theta)$ is the Fisher information matrix.

Since $\hat{C}_0 = \hat{C} - c$ and c is known, we can apply the standard error of $\hat{C}$ to $\hat{C}_0$. Thus, the asymptotic standard error of $\hat{C}_0$ is

$$SE(\hat{C}_0) = \left[\frac{1}{\hat{C}}\left[\frac{1-p_0(\hat{\theta})}{p_0(\hat{\theta})} - \frac{1}{p_0^2(\hat{\theta})}\left(\frac{dp(0)}{d\hat{\theta}}\right)^T I\left(\hat{\theta}\right)^{-1}\left(\frac{dp(0)}{d\hat{\theta}}\right)\right]\right]^{-1/2},$$

where we plug in our parameter estimate, $\hat{\theta}$.

2.2.3 Goodness of Fit

In order to measure departure of the data from the model, an asymptotic χ^2 goodness of fit statistics is used. Let W be

$$W = \sum_{k=1}^{K} \frac{(O_k - E_k)^2}{E_k}, \tag{2.5}$$

where $k = 1, \ldots, K = \max X_i + 1$. Then O_k is the observed count in kth cell and E_k is the expected count in the kth cell. We have,

$$O_k = \sum_{i=1}^{c} I\{X_i = k\}$$

and

$$E_k = \hat{C}P[X_i = k],$$

for $k = 1, \ldots, K-1$; and

$$O_K = 0$$

and

$$E_K = \hat{C}\sum_{m=K-1}^{\infty} P[X_i = m]$$

for the last cell.

The asymptotic distribution of W is χ^2_ν where $\nu = K - p - 1$ is the degrees of freedom for a model with p parameters. If any of the expected cell counts are less than five, cells are binned. Starting from the lowest cell frequency, $\sum_{i=1}^{c} I\{X_i = 1\}$, cells are binned one by one until the expected count is greater than or equal to five. If the last binned cell has an expected count less than five, then the last two cells are binned. A more exact distribution of W can be found using a parametric bootstrap procedure (Tollenaar & Mooijaart 2003 [32]).

2.2.4 Computation

Fitting the Exponential mixed Poisson model is not a difficult task. Maximum likelihood estimates can be computed by hand. Standard errors and goodness of fit statistics can be calculated directly.

2.3 Example

In this section we will focus on the microbial data with cut-off 99%. Table 2.1 shows the results for each value of the tuning parameter, τ.

Figure 2.1 shows the fitted values for this data with $\tau = 22$. It is clear that the estimated values, represented by the asterisks on the plot, do not fit the data well. This agrees with the p-value from the goodness of fit statistic.

Table 2.1: Results for Microbial 99%

τ	$\hat{C}$	$SE(\hat{C})$	*GOF p-value*
4	6,294.0	145.5	0.0000
5	5,716.8	128.1	0.0000
6	5,392.6	118.4	0.0000
7	5,218.6	113.1	0.0000
8	4,918.0	104.0	0.0000
9	4,855.3	102.1	0.0000
11	4,705.7	97.6	0.0000
17	4,592.1	94.1	0.0000
22	4,452.7	89.8	0.0000

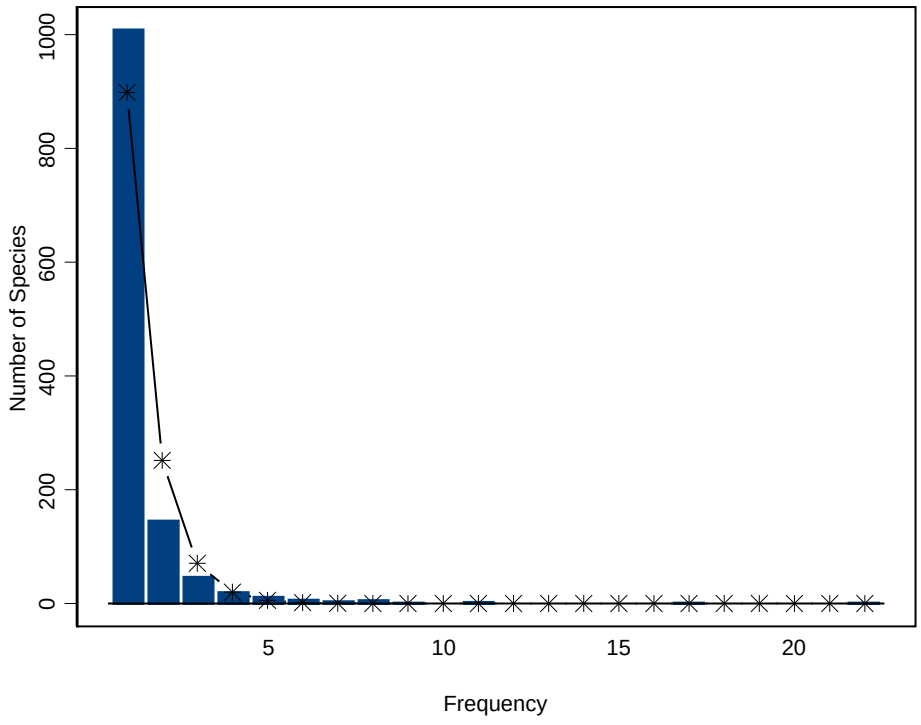

Figure 2.1: Fitted Values with Microbial 99% ($\tau = 22$)

Chapter 3

A Mixture of Two Exponential Distributions

Typically, the zero-truncated Exponential mixed Poisson model is not flexible enough to model the data. The lack of fit can be quantified in the goodness of fit statistic. Searching for a more flexible model is the motivation for looking at mixtures of Exponential distributions.

3.1 Model Description

Consider a three parameter probability density for λ_j,

$$p\left(\lambda_j \middle| \theta_1, \theta_2, \theta_3\right) = \theta_3 f_1 + (1 - \theta_3) f_2,$$

where

$$f_1 = \frac{1}{\theta_1} e^{-\frac{\lambda_j}{\theta_1}}$$

and

$$f_2 = \frac{1}{\theta_2} e^{-\frac{\lambda_j}{\theta_2}}$$

are two Exponential distributions with means θ_1 and θ_2, respectively. Thus

$$p\left(\lambda_j|\,\theta_1,\theta_2,\theta_3\right)=\theta_3\frac{1}{\theta_1}\,e^{-\frac{\lambda_j}{\theta_1}}+(1-\theta_3)\frac{1}{\theta_2}\,e^{-\frac{\lambda_j}{\theta_2}}.$$

Using equation 1.1 the two-mixed-Exponential mixed Poisson distribution is

$$\begin{aligned}
P[Y_j=y_j] &= \int p\left(\lambda_j|\,\theta_1,\theta_2,\theta_3\right)p\left(y_j|\,\lambda_j\right)d\lambda_j \\
&= \theta_3\int f_1 p\left(y_j|\,\lambda_j\right)d\lambda_j+(1-\theta_3)\int f_2 p\left(y_j|\,\lambda_j\right)d\lambda_j \\
&= \theta_3\frac{1}{1+\theta_1}\left(\frac{\theta_1}{1+\theta_1}\right)^{y_j}+(1-\theta_3)\frac{1}{1+\theta_2}\left(\frac{\theta_2}{1+\theta_2}\right)^{y_j}.
\end{aligned}$$

The marginal distribution for Y_j is a weighted sum of two geometric probabilities.

The zero-truncated distribution is

$$\begin{aligned}
P[X_i=x_i] &= \frac{P[X_i=x_i]}{1-P[X_i=0]} \\
&= \frac{\theta_3\frac{1}{1+\theta_1}\left(\frac{\theta_1}{1+\theta_1}\right)^{x_i}+(1-\theta_3)\frac{1}{1+\theta_2}\left(\frac{\theta_2}{1+\theta_2}\right)^{x_i}}{1-\left(\theta_3\frac{1}{1+\theta_1}+(1-\theta_3)\frac{1}{1+\theta_2}\right)}.
\end{aligned}$$

3.2 Computation by the EM Algorithm

We wish to fit the zero-truncated two-mixed-Exponential mixed Poisson distribution to the frequency data. Treating as a missing data problem with the observations $Y_j=0$ as missing data, we can use the Expectation Maximization (EM) algorithm by imputing a value for $\sum I\{Y_j=0\}$ and then maximize the non-zero-truncated distribution. Iterating through these two steps gives us a maximum likelihood estimate for $\theta=(\theta_1,\theta_2,\theta_3)$. The likelihood for the weighted sum of geometric distributions can be maximized by way of the EM algorithm. Embedding another EM into the M step of the outer EM algorithm gives us a nested EM (Böhning and Schön 2005 [3]).

3.2.1 E Step

The first step is to specify an initial value for the expected zero count, $\sum I\{Y_j = 0\}$. Based on the idea that one component of the mixture models the rare species while the other component models the abundant species, we will use the first four frequencies of the data, $\{x_i : x_i \leq 4\}$, to estimate the starting value with a single Exponential mixed Poisson distribution truncated on the left and right; that is, $X = Y|0 < Y \leq 4$ where Y is a Geometric random variable with success probability p. Let $n^{(0)}$ be the number of observations in $\{x_i : x_i \leq 4\}$. The likelihood for $\{x_i : x_i \leq 4\}$ is

$$\begin{aligned} L(p) &= \prod_{i=1}^{n^{(0)}} P[Y = x_i | Y \leq 4] \\ &= \prod_{i=1}^{n^{(0)}} \frac{P[Y = x_i]}{\sum_{j=1}^{4} P[Y = j]}. \end{aligned}$$

The log likelihood is

$$\begin{aligned} l(p) &= \sum_{i=1}^{n^{(0)}} \log P[Y = x_i] - n^{(0)} \log \left(\sum_{j=1}^{4} P[Y = j] \right) \\ &= \sum_{i=1}^{n^{(0)}} \log \left[(1-p)^{x_i} \, p \right] - n^{(0)} \log \left(\sum_{j=1}^{4} (1-p)^j \, p \right) \\ &= \log (1-p) \sum_{i=1}^{n^{(0)}} x_i - n^{(0)} \log \sum_{j=1}^{4} (1-p)^j. \end{aligned}$$

Taking a derivative with respect to p, we have

$$\frac{l(p)}{dp} = \frac{-\sum_{i=1}^{n^{(0)}} x_i}{1-p} - n^{(0)} \frac{\sum_{j=1}^{4} j(1-p)^{j-1}}{\sum_{j=1}^{4} (1-p)^j}.$$

The maximum likelihood estimate for p solves

$$\frac{l(p)}{dp} = 0.$$

The estimated probability $Y = 0|Y \leq 4$ is

$$\hat{p}_0 = \frac{\hat{p}}{\sum_{j=0}^{4}(1-\hat{p})^j \hat{p}}.$$

The expected zero count is

$$\hat{C}_{0(0)} = \frac{\hat{p}_0 n^{(0)}}{1-\hat{p}_0}.$$

The zero in parenthesis indicates the iteration number for the outer EM, indexed by k.

We treat the unobserved zero count as missing data. This step imputes a value for $\hat{C}_{0(k+1)}$. The expected zero count for the kth iteration given the current parameter estimates, $\hat{\theta}_{(k)}$ is

$$\hat{C}_{0(k+1)} = p_0(\hat{\theta}_{(k)}) \left(c + \hat{C}_{0(k)} \right),$$

where $p_0(\hat{\theta}_{(k)})$ is the probability $Y = 0$ under the non zero-truncated model.

3.2.2 M Step

Next, we want to update our parameter estimates. We can now maximize the non-zero-truncated model, which is a mixture of Geometric distributions, using the EM algorithm. We will refer to this embedded EM as the inner EM. The EM used to impute the expected zero counts will be called the outer EM.

The first step is to specify initial values for the parameters $(\theta_1, \theta_2, \theta_3)$. Consider the non truncated data as contributions from the two exponential mixing distributions. To obtain rough starting values, we use the low and high frequencies to estimate the mean of each mixing distribution. Let $Y_i^{(1)}$ be a subset of the data such that $\min Y_i \leq Y_i \leq \lfloor \frac{2}{3} \max Y_i \rfloor + 1$ where $\lfloor y \rfloor$ is the greatest integer less than or equal to y. Let $Y_i^{(2)}$ be a subset of the data such that

$\lfloor \frac{1}{3} \max Y_i \rfloor + 1 \leq Y_i \leq \max Y_i$. Also, let $n^{(1)}$ and $n^{(2)}$ be the number of elements in the sets $Y_i^{(1)}$ and $Y_i^{(2)}$ respectively. The initial values for θ_1 and θ_2 are

$$\theta_{1(0)} = \bar{Y}^{(1)} = \frac{1}{n^{(1)}} \sum_{i=1}^{n^{(1)}} Y_i^{(1)}$$

and

$$\theta_{2(0)} = \bar{Y}^{(2)} = \frac{1}{n^{(2)}} \sum_{i=1}^{n^{(2)}} Y_i^{(2)}.$$

We let $\theta_{3(0)} = 1/2$. The subscript in parenthesis indicates the iteration number for the inner EM, indexed by l.

Nested E Step

Within one iteration of the outer EM, there is one imputed value of C_0, which together with the observed data, is the non-truncated data for the lth iteration. This data can be modeled by the non-truncated two-mixed-Exponential distribution. We apply the EM to maximize the parameters in this finite mixture as described in McLachlan and Peel (2000 [23] section 2.8).

Let Z_{1i} be a random variable which indicates Y_i originated from the first component and $Z_{2i} = 1 - Z_{1i}$ be a random variable which indicates Y_i originated from the second component. Using the current parameter estimates, $\theta_{(l-1)}$, we impute the expected values for $Z_{1i(l)}$ and $Z_{2i(l)}$. For each Y_i, we find the conditional expectation given the data. Bayes formula gives us

$$\begin{aligned}
\mathbb{E}[Z_{1i(l)}] &= P[Z_{1i} = 1 | Y_i = y_i] \\
&= \frac{P[Y_i = y_i | Z_{1i} = 1] P[Z_{1i} = 1]}{P[Y_i = y_i | Z_{1i} = 1] P[Z_{1i} = 1] + P[Y_i = y_i | Z_{1i} = 0] P[Z_{1i} = 0]} \\
&= \frac{\left(\frac{\theta_{1(l-1)}}{1+\theta_{1(l-1)}}\right)^{y_i} \frac{1}{1+\theta_{1(l-1)}} \theta_{3(l-1)}}{\left(\frac{\theta_{1(l-1)}}{1+\theta_{1(l-1)}}\right)^{y_i} \frac{1}{1+\theta_{1(l-1)}} \theta_{3(l-1)} + \left(\frac{\theta_{2(l-1)}}{1+\theta_{2(l-1)}}\right)^{y_i} \frac{1}{1+\theta_{2(l-1)}} (1 - \theta_{3(l-1)})}
\end{aligned}$$

and

$$\begin{aligned}\mathbb{E}[Z_{2(l)}] &= P[Z_{1i} = 0|Y_i = y_i] \\ &= 1 - P[Z_{1i} = 1|Y_i = y_i].\end{aligned}$$

Nested M Step

The parameter estimates for θ_1 and θ_2 have a closed form following equation 2.33 in (Mclachlan and Peel 2000 [23]) given by

$$\theta_{1(l)} = \frac{\sum \mathbb{E}[Z_{1i(l)}]Y_i}{\sum \mathbb{E}[Z_{1i(l)}]}$$

and

$$\theta_{2(l)} = \frac{\sum \mathbb{E}[Z_{2i(l)}]Y_i}{\sum \mathbb{E}[Z_{2i(l)}]}$$

The estimate for $\theta_{3(l)}$ is the sample average of the $\mathbb{E}[Z_{1i(l)}]$,

$$\theta_{3(l)} = \frac{\sum \mathbb{E}[Z_{1i(l)}]}{c + c_{0(k)}}.$$

The sum, $\sum \mathbb{E}[Z_{1i(l)}] = \sum P[Z_{1i} = 1|Y_i = y_i]$, is calculated by adding all $\mathbb{E}[Z_{1i(l)}]$ terms for $Y_i > 0$ with $\mathbb{E}[Z_{1i(l)}]\hat{C}_0$ for $Y_i = 0$.

Convergence Criterion

Convergence is determined when the likelihood for the incomplete data is not decreased after an EM iteration, for both the outer and inner EM. For the outer EM, iterations are ceased when

$$1 - \varepsilon \leq \frac{\hat{C}_{0(k)}}{\hat{C}_{0(k-1)}} \leq 1 + \varepsilon$$

for small, positive ε. For the inner EM, iterations are ceased when all parameter estimates meet the criteria

$$1-\varepsilon \leq \frac{\hat{\theta}_{1(l)}}{\hat{\theta}_{1(l-1)}} \leq 1+\varepsilon,$$
$$1-\varepsilon \leq \frac{\hat{\theta}_{2(l)}}{\hat{\theta}_{2(l-1)}} \leq 1+\varepsilon,$$

and

$$1-\varepsilon \leq \frac{\hat{\theta}_{3(l)}}{\hat{\theta}_{3(l-1)}} \leq 1+\varepsilon.$$

In the examples in sections 3.3 and 4.3, ε is set equal to 10^{-6}.

3.3 Example

Again we focus on the microbial data with cut-off 99%. Table 3.1 shows the results from fitting the zero-truncated two mixed-exponential mixed Poisson model for each value of the tuning parameter, τ.

Figure 3.1 shows the fitted values for this data with $\tau = 22$. This model shows a better fit than the predicted values from the single Exponential model quantified by the large goodness of fit p-values from the fourth column in table 3.1. We would like to use as much data as possible, meaning the largest value

Table 3.1: Results for Microbial 99%

τ	$\hat{C}$	$SE(\hat{C})$	*GOF p-value*
6	13,966.7	6,191.6	0.1339
7	12,849.4	4,315.5	0.5210
8	11,236.2	2,472.6	0.5574
9	10,958.7	2,220.8	0.7218
11	10,256.5	1,738.4	0.8063
17	9,678.9	1,410.6	0.7638
22	9,063.0	1,127.4	0.6493

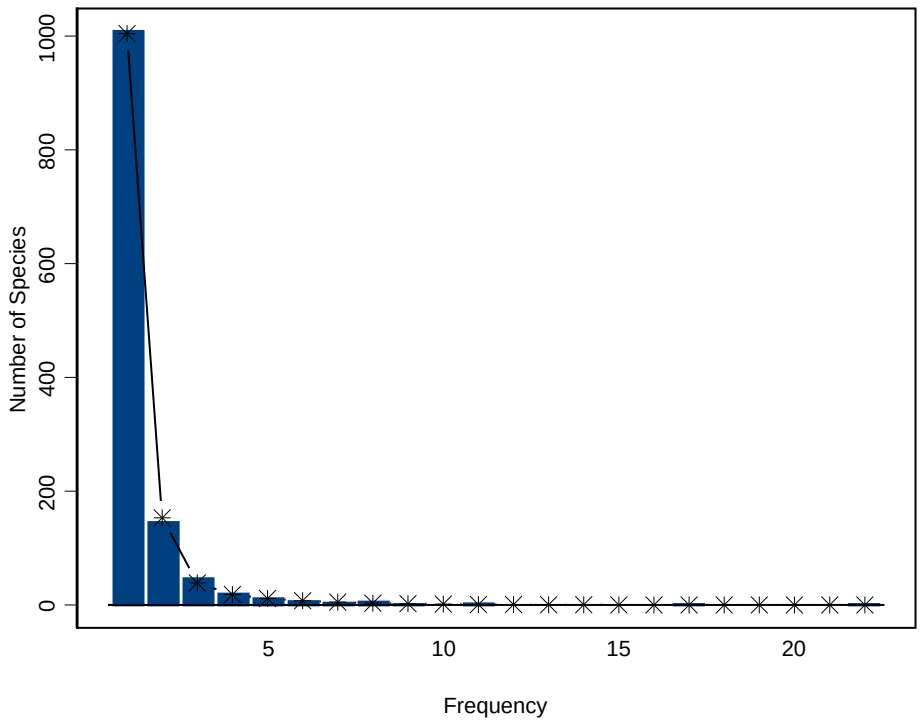

Figure 3.1: Fitted Values with Microbial 99% ($\tau = 22$)

of τ. With the two mixed-Exponential model, we can use all of the available data ($\tau = 22$) with model agreement.

Figure 3.2 shows the fitted values for the microbial data at cut-off percentage 98 ($\tau = 16$); table 3.2 shows the results; and table 3.3 shows the parameter estimates.

Tables 3.4, 3.5, 3.6, 3.7, 3.8, and 3.9 show results for all other cutoff percentages for the microbial data. An NA appears for the GOF p-value when there are not enough degrees of freedom to calculate the goodness of fit statistic after binning cells.

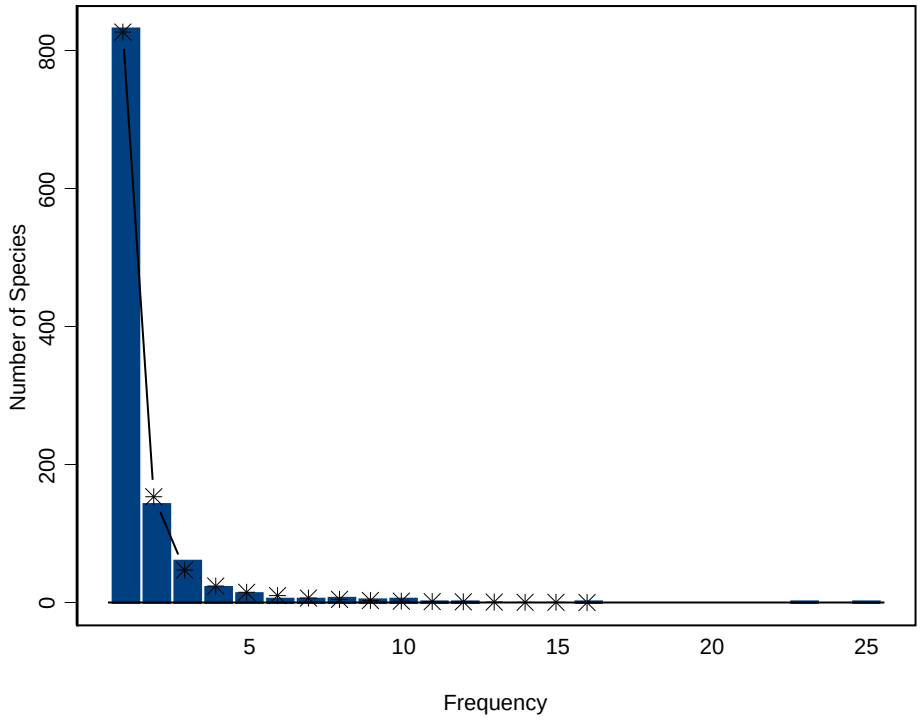

Figure 3.2: Fitted Values with Microbial 98% ($\tau = 16$)

Table 3.2: Results for Microbial 98%

τ	$\hat{C}$	$SE(\hat{C})$	*GOF p-value*
6	11,967.7	10,019.9	0.0964
7	11,243.1	6,675.4	0.3828
8	9,322.8	3,137.2	0.0673
9	8,155.5	1,903.9	0.0517
10	7,213.8	1,214.3	0.0100
11	7,061.6	1,127.4	0.0205
12	6,896.1	1,040.3	0.0285
16	6,631.3	906.4	0.0941
23	6,250.1	745.7	0.0973
25	5,935.5	637.2	0.0677

Table 3.3: Parameter Estimates for Microbial 98%

τ	$\hat{\theta}_1$	$\hat{\theta}_2$	$\hat{\theta}_3$
6	0.05542	0.73886	0.89974
7	0.06476	0.87155	0.91299
8	0.09018	1.11146	0.92398
9	0.11324	1.33931	0.93235
10	0.13986	1.69176	0.94281
11	0.14508	1.77538	0.94481
12	0.15109	1.87431	0.94701
16	0.16141	2.04080	0.95050
23	0.17828	2.33765	0.95586
25	0.19439	2.68506	0.96070

Table 3.4: Results for Microbial 70%

τ	$\hat{C}$	$SE(\hat{C})$	*GOF p-value*
6	202.7	1,413.5	NA
22	96.9	114.6	NA
53	40.9	13.7	NA
91	40.0	13.6	NA
339	38.4	13.8	NA

Table 3.5: Results for Microbial 80%

τ	$\hat{C}$	$SE(\hat{C})$	*GOF p-value*
6	827.3	2,071.3	NA
7	793.0	1,612.5	0.0725
8	737.7	1,139.2	0.1135
9	679.7	785.4	0.1563
11	614.7	506.9	0.2372
12	569.0	362.6	0.4997
13	536.2	278.6	0.5494
14	496.1	200.3	0.6007
15	474.4	166.7	0.4459
16	466.4	155.1	0.4386
20	453.5	141.0	0.4549
22	441.2	127.8	0.1526
23	431.2	117.9	0.1473
25	421.7	109.4	0.1522
27	412.8	102.1	0.1210
31	402.8	94.7	0.1399
34	393.2	88.3	0.1437
38	383.5	81.5	0.2340
56	366.9	71.8	0.2668
76	347.1	61.6	0.3548
81	332.7	54.4	0.2426
102	319.3	48.4	0.2440
115	309.3	45.2	0.1801
130	301.8	43.3	0.1092
264	289.7	39.6	0.0507
364	280.7	37.2	0.0189

Table 3.6: Results for Microbial 90%

τ	$\hat{C}$	$SE(\hat{C})$	*GOF p-value*
6	3,612.7	6,289.2	0.0323
7	3,428.4	3,851.3	0.1196
8	2,827.9	1,615.4	0.2381
9	2,348.5	714.2	0.0706
10	2,115.2	442.8	0.0138
11	2,087.6	414.0	0.0351
12	2,028.9	367.4	0.0486
13	1,975.7	326.7	0.0685
15	1,917.5	288.7	0.1232
16	1,891.2	274.1	0.1426
17	1,865.2	260.8	0.1584
19	1,836.3	244.4	0.2780
20	1,809.4	232.4	0.2878
25	1,773.1	215.3	0.3398
28	1,737.0	199.9	0.3337
34	1,696.4	184.3	0.3114
36	1,661.4	171.9	0.2669
41	1,600.4	152.6	0.0952
45	1,575.1	145.4	0.0608
49	1,551.7	139.2	0.0392
52	1,530.9	132.8	0.0165
53	1,513.7	125.0	0.0313
62	1,495.1	122.0	0.0210
96	1,466.3	117.5	0.0029
123	1,436.4	112.2	0.0018

Table 3.7: Results for Microbial 95%

τ	$\hat{C}$	$SE(\hat{C})$	*GOF p-value*
6	6,808.7	6,142.4	0.0065
7	5,650.8	2,496.6	0.0199
8	4,875.1	1,355.4	0.0086
9	4,528.2	993.8	0.0302
10	4,372.2	865.7	0.0615
11	4,160.1	708.5	0.0793
12	4,036.8	633.8	0.1016
13	3,810.2	504.3	0.1532
14	3,774.1	487.6	0.0999
18	3,711.8	461.6	0.2525
19	3,603.9	415.5	0.2473
20	3,557.2	398.9	0.2804
21	3,513.2	380.1	0.2910
25	3,459.2	362.9	0.3206
28	3,404.1	343.0	0.4110
30	3,352.2	328.4	0.3843
33	3,301.5	311.9	0.3486
37	3,250.4	296.5	0.3005
41	3,200.8	285.4	0.2248
58	3,132.6	265.8	0.0653

Table 3.8: Results for Microbial 96%

τ	$\hat{C}$	$SE(\hat{C})$	*GOF p-value*
6	5,755.3	4,199.5	0.0874
7	4,713.5	1,408.4	0.0634
8	4,293.3	830.6	0.0105
9	4,177.5	721.4	0.0784
10	4,091.5	646.4	0.1870
11	3,935.1	536.0	0.2520
12	3,839.7	481.6	0.4189
13	3,780.4	447.0	0.5353
18	3,720.8	421.1	0.6265
20	3,659.1	396.4	0.6758
21	3,602.3	371.2	0.6928
22	3,550.5	353.3	0.6694
25	3,495.1	331.9	0.6977
28	3,438.9	312.1	0.6758
33	3,377.7	295.5	0.5380
37	3,318.4	277.9	0.4489
44	3,256.0	261.2	0.3286

Table 3.9: Results for Microbial 97%

τ	$\hat{C}$	$SE(\hat{C})$	*GOF p-value*
6	8,237.0	6,482.1	0.0979
7	6,768.1	2,533.3	0.0054
8	6,131.7	1,656.1	0.0254
10	5,450.8	1,025.1	0.1563
11	5,106.4	785.9	0.1999
12	5,004.0	727.0	0.2224
13	4,677.6	559.7	0.0215
14	4,613.4	532.5	0.2936
15	4,548.1	501.3	0.1271
20	4,443.4	463.3	0.2015
21	4,351.5	428.4	0.1860
24	4,257.1	399.3	0.1710
25	4,111.6	355.3	0.1463

Chapter 4

A Mixture of Three Exponential Distributions

We would like to explore mixtures of Exponential Distributions with more than two components. Situations may arise where the two-mixed-Exponential model is not flexible enough to fit the data. This chapter explores this five parameter model.

4.1 Model Description

Consider a five parameter probability density for λ_j,

$$p\left(\lambda_j|\,\theta_1,\theta_2,\theta_3,\theta_4,\theta_5\right) = \theta_4 f_1 + \theta_5 f_2 + (1-\theta_4-\theta_5) f_3,$$

where

$$f_1 = \frac{1}{\theta_1}\, e^{-\frac{\lambda_j}{\theta_1}},$$
$$f_2 = \frac{1}{\theta_2}\, e^{-\frac{\lambda_j}{\theta_2}},$$

and

$$f_3 = \frac{1}{\theta_3}\, e^{-\frac{\lambda_j}{\theta_3}}.$$

are three Exponential distributions with means θ_1, θ_2, and θ_3, respectively.

The three-mixed-Exponential mixed Poisson distribution is a mixture of Geometric distributions with success probabilities $1/(1+\theta_1)$, $1/(1+\theta_2)$, and $1/(1+\theta_3)$, respectively. Thus, the non-zero-truncated distribution for Y_j is a weighted sum of three geometric probabilities.

The zero-truncated distribution is

$$\begin{aligned} P[X_{i_j} = x_{i_j}] &= \frac{P[Y_j = x_{i_j}]}{1 - P[Y_j = 0]} \\ &= \frac{\theta_4 \frac{1}{1+\theta_1}\left(\frac{\theta_1}{1+\theta_1}\right)^{x_{i_j}} + \theta_5 \frac{1}{1+\theta_2}\left(\frac{\theta_2}{1+\theta_2}\right)^{x_{i_j}} + (1-\theta_4-\theta_5)\frac{1}{1+\theta_3}\left(\frac{\theta_3}{1+\theta_3}\right)^{x_{i_j}}}{1 - \left(\theta_4 \frac{1}{1+\theta_1} + \theta_5 \frac{1}{1+\theta_2} + (1-\theta_4-\theta_5)\frac{1}{1+\theta_3}\right)}. \end{aligned}$$

4.2 Computation

Implementation of the EM algorithm is an extension of the two-mixed-Exponential fitting algorithm.

4.2.1 Starting Values

The starting value for the expected zero count, $\hat{C}_{0(k)}$ is found using the same method from section 3.2.1.

The starting values for the parameter estimates are found in a similar manner to section 3.2.2. We split the data into three sections, estimate a geometric distribution for each of the subsets, and then use these estimates for starting values for the mixture distribution.

Let $Y_i^{(1)}$ be a subset of the data such that $\min Y_i \le Y_i \le \lfloor \frac{2}{4} \max Y_i \rfloor + 1$ where $\lfloor y \rfloor$ is the greatest integer less than or equal to y. Let $Y_i^{(2)}$ be a subset of the data such that $\lfloor \frac{1}{4} \max Y_i \rfloor + 1 \le Y_i \le \lfloor \frac{3}{4} \max Y_i \rfloor + 1$. Let $Y_i^{(3)}$ be a subset of the data such that $\lfloor \frac{2}{4} \max Y_i \rfloor + 1 \le Y_i \le \max Y_i$. Let $n^{(1)}$, $n^{(2)}$, and $n^{(3)}$ be the number of elements in the sets $Y_i^{(1)}$, $Y_i^{(2)}$, and $Y_i^{(3)}$ respectively. The initial

values for θ_1, θ_2, and θ_3 are

$$\theta_{1(0)} = \bar{Y}^{(1)} = \frac{1}{n^{(1)}} \sum_{i=1}^{n^{(1)}} Y_i^{(1)}$$

$$\theta_{2(0)} = \bar{Y}^{(2)} = \frac{1}{n^{(2)}} \sum_{i=1}^{n^{(2)}} Y_i^{(2)}$$

$$\theta_{3(0)} = \bar{Y}^{(3)} = \frac{1}{n^{(3)}} \sum_{i=1}^{n^{(3)}} Y_i^{(3)}.$$

We let $\theta_{4(0)} = \theta_{5(0)} = 1/3$.

4.2.2 E Step

Using the parameter estimates from the M step, $\hat{\theta}_k$, we impute a new value for the expected zero count, $\hat{C}_{0(k+1)}$ by

$$\hat{C}_{0(k+1)} = p_0(\hat{\theta}_{(k)}) \left(c + \hat{C}_{0(k)} \right),$$

where $p_0(\hat{\theta}_{(k)})$ is the probability $Y = 0$ under the non zero-truncated model, that is

$$p_0(\hat{\theta}_{(k)}) = \hat{\theta}_{4(k)} \frac{1}{1 + \hat{\theta}_{1(k)}} + \hat{\theta}_{5(k)} \frac{1}{1 + \hat{\theta}_{2(k)}} + (1 - \hat{\theta}_{4(k)} - \hat{\theta}_{5(k)}) \frac{1}{1 + \hat{\theta}_{3(k)}}.$$

4.2.3 M Step

Again we nest an EM algorithm for fitting the missing component mixture distribution inside the M step for the truncated distribution. We use the starting values for the five parameters as defined in section 4.2.1.

Nested E Step

Let Z_{1i} be a random variable which indicates if Y_i originated from the first component, Z_{2i} indicating the second component, and Z_{3i} indicating the third

component. Using the current parameter estimates, $\theta_{(jl1)}$, we impute the expected values for $Z_{1i(l)}$ and $Z_{2i(l)}$ for $i = 1, 2, \ldots, n$. For each Y_i, Bayes formula gives us

$$\begin{aligned}\mathbb{E}[Z_{1i(l)}] &= P[Z_{1i}=1, Z_{2i}=0, Z_{3i}=0|Y_i=y_i] \\ &= \frac{P[Y_i=y_i|Z_{1i}=1, Z_{2i}=0, Z_{3i}=0]P[Z_{1i}=1, Z_{2i}=0, Z_{3i}=0]}{D},\end{aligned}$$

where

$$\begin{aligned}D &= P[Y_i=y_i|Z_{1i}=1, Z_{2i}=0, Z_{3i}=0]P[Z_{1i}=1, Z_{2i}=0, Z_{3i}=0] \\ &\quad +P[Y_i=y_i|Z_{1i}=0, Z_{2i}=1, Z_{3i}=0]P[Z_{1i}=0, Z_{2i}=1, Z_{3i}=0] \\ &\quad +P[Y_i=y_i|Z_{1i}=0, Z_{2i}=0, Z_{3i}=1]P[Z_{1i}=0, Z_{2i}=0, Z_{3i}=1].\end{aligned}$$

Thus,

$$\mathbb{E}[Z_{1i(l)}] = \frac{\left(\frac{\theta_{1(l-1)}}{1+\theta_{1(l-1)}}\right)^{y_i} \frac{1}{1+\theta_{1(l-1)}}\,\theta_{4(l-1)}}{D},$$

where

$$\begin{aligned}D &= \left(\frac{\theta_{1(l-1)}}{1+\theta_{1(l-1)}}\right)^{y_i} \frac{1}{1+\theta_{1(l-1)}}\,\theta_{4(l-1)} + \left(\frac{\theta_{2(l-1)}}{1+\theta_{2(l-1)}}\right)^{y_i} \frac{1}{1+\theta_{2(l-1)}}\theta_{5(l-1)} \\ &\quad + \left(\frac{\theta_{3(l-1)}}{1+\theta_{3(l-1)}}\right)^{y_i} \frac{1}{1+\theta_{3(l-1)}}(1-\theta_{4(l-1)}-\theta_{5(l-1)}.\end{aligned}$$

Similarly,

$$\begin{aligned}\mathbb{E}[Z_{2i(l)}] &= P(Z_{1i}=0, Z_{2i}=1, Z_{3i}=0|Y_i=y_i) \\ &= \frac{P(Y_i=y_i|Z_{1i}=0, Z_{2i}=1, Z_{3i}=0)P(Z_{1i}=0, Z_{2i}=1, Z_{3i}=0)}{D} \\ &= \frac{\left(\frac{\theta_{2(l-1)}}{1+\theta_{2(l-1)}}\right)^{y_i} \frac{1}{1+\theta_{2(l-1)}}\,\theta_{5(l-1)}}{D}\end{aligned}$$

and

$$\begin{aligned}\mathbb{E}[Z_{3i(l)}] &= P(Z_{1i}=0, Z_{2i}=0, Z_{3i}=1|Y_i=y_i) \\ &= \frac{P(Y_i=y_i|Z_{1i}=0, Z_{2i}=0, Z_{3i}=1)P(Z_{1i}=0, Z_{2i}=0, Z_{3i}=1)}{D} \\ &= \frac{\left(\frac{\theta_{3(l-1)}}{1+\theta_{3(l-1)}}\right)^{y_i} \frac{1}{1+\theta_{3(l-1)}} \left(1-\theta_{4(l-1)}-\theta_{5(l-1)}\right)}{D}.\end{aligned}$$

Nested M Step

Following equation 2.33 in (Mclachlan and Peel 2000 [23]), the parameter estimates for θ are

$$\theta_{1(l)} = \frac{\sum \mathbb{E}[Z_{1i(l)}]Y_i}{\sum \mathbb{E}[Z_{1i(l)}]},$$
$$\theta_{2(l)} = \frac{\sum \mathbb{E}[Z_{2i(l)}]Y_i}{\sum \mathbb{E}[Z_{2i(l)}]},$$

and

$$\theta_{3(l)} = \frac{\sum \mathbb{E}[Z_{3i(l)}]Y_i}{\sum \mathbb{E}[Z_{3i(l)}]}.$$

The estimate for the weight parameters are

$$\theta_{3(l)} = \frac{\sum \mathbb{E}[Z_{1i(l)}]}{c + c_{0(k)}}$$

and

$$\theta_{4(l)} = \frac{\sum \mathbb{E}[Z_{2i(l)}]}{c + c_{0(k)}}.$$

Convergence Criterion

We maintain the same convergence criterion for each EM. Iterations for the outer EM are ceased when

$$1-\varepsilon \leq \frac{\hat{C}_{0(k)}}{\hat{C}_{0(k-1)}} \leq 1+\varepsilon$$

for small, positive ε. For the inner EM, iterations are ceased when all parameter estimates meet the criteria

$$1-\varepsilon \leq \frac{\hat{\theta}_{1(l)}}{\hat{\theta}_{1(l-1)}} \leq 1+\varepsilon$$
$$1-\varepsilon \leq \frac{\hat{\theta}_{2(l)}}{\hat{\theta}_{2(l-1)}} \leq 1+\varepsilon$$
$$1-\varepsilon \leq \frac{\hat{\theta}_{3(l)}}{\hat{\theta}_{3(l-1)}} \leq 1+\varepsilon$$
$$1-\varepsilon \leq \frac{\hat{\theta}_{4(l)}}{\hat{\theta}_{4(l-1)}} \leq 1+\varepsilon$$
$$1-\varepsilon \leq \frac{\hat{\theta}_{5(l)}}{\hat{\theta}_{5(l-1)}} \leq 1+\varepsilon.$$

4.2.4 Aitken's Acceleration

The EM algorithm sometimes requires an immense number of iterations to converge. This slowness is mainly due to the small changes of the imputed zero count and the parameter estimates between iterations in the primary EM. In order to increase the speed of the algorithm we utilize an acceleration method which attempts to skip iterations in the sequence. The method used is a version of Aitken's acceleration (McLachlan and Peel 2000 [23]) with the number of skipped iterations determined by the change in the imputed zero count. This method is applied to the outer EM.

We assume the change in the value of the imputed zero count, $\Delta_{(k)} = \left|\hat{C}_{0(k)} - \hat{C}_{0(k-1)}\right|$, is decreasing with constant rate. Thus

$$f(k) = \Delta_{(k)} + \left(\Delta_{(k)} - \Delta_{(k-1)}\right) k$$

describes the sequence of $\Delta_{(k)}$. To find the iteration where $\Delta_{(k)} = 0$, we solve

$$0 = \Delta_{(k)} + \left(\Delta_{(k)} - \Delta_{(k-1)}\right) k$$

for k, obtaining $k = \frac{\Delta_{(k)}}{\left(\Delta_{(k-1)} - \Delta_{(k)}\right)}$.

The value of the next imputed zero count, $\hat{C}_{0(k+1)}$ is

$$\begin{aligned}
\hat{C}_{0(k)} + k\Delta_{(k)} &= \hat{C}_{0(k)} + \left(\frac{\Delta_{(k)}}{\Delta_{(k-1)} - \Delta_{(k)}}\right)\Delta_{(k)} \\
&= \hat{C}_{0(k)} + \frac{\left(\hat{C}_{0(k)} - \hat{C}_{0(k-1)}\right)^2}{\left(\hat{C}_{0(k-1)} - \hat{C}_{0(k-2)}\right) - \left(\hat{C}_{0(k)} - \hat{C}_{0(k-1)}\right)} \\
&= \hat{C}_{0(k)} - \frac{\left(\hat{C}_{0(k)} - \hat{C}_{0(k-1)}\right)^2}{\hat{C}_{0(k)} - 2\hat{C}_{0(k-1)} + \hat{C}_{0(k-2)}}.
\end{aligned}$$

This acceleration step is used after 10 successful EM iterations. Since the acceleration requires three consecutive estimates for the unobserved number of classes, $\hat{C}_{0(k)}$, $\hat{C}_{0(k-1)}$, and $\hat{C}_{0(k-2)}$, the acceleration can be used at most on every third iteration. We also do not implement the acceleration step if the change in consecutive estimates is not decreasing.

4.3 Example

Again we focus on the microbial data with cut-off 99%. Table 4.1 shows the results from fitting the zero-truncated three mixed-exponential mixed Poisson model for each value of the tuning parameter, τ. The NA's appearing in the fourth column are due to an uncalculable p-value due to insufficient degrees of freedom after binning cells.

Figure 4.1 shows the fitted values for this data with $\tau = 22$. It is hard to see any deviation from the observed and fitted values from this model. Large goodness of fit statistics in table 4.1, show that the models would not reject a goodness of fit hypothesis test. However, with a five parameter model there is danger in overfitting the model with too many components. Table 4.2 shows the parameter estimates for all of the values of τ on the same data set.

Table 4.1: Results for Microbial 99%

τ	$\hat{C}$	$SE(\hat{C})$	*GOF p-value*
7	26,284.8	26,770.4	NA
8	11,885.0	2,862.6	NA
9	11,490.6	2,533.5	0.2344
11	10,586.7	1,878.9	0.3240
17	9,906.0	1,490.1	0.3002
22	9,216.3	1,170.8	0.3131

Table 4.2: Parameter Estimates for Microbial 99%

τ	$\hat{\theta}_1$	$\hat{\theta}_2$	$\hat{\theta}_3$	$\hat{\theta}_4$	$\hat{\theta}_5$
7	0.03114	0.03114	0.75556	0.62935	0.32838
8	0.09183	0.09183	1.15288	0.62148	0.33306
9	0.09690	0.09690	1.22649	0.62738	0.32946
11	0.11029	0.11029	1.44236	0.66230	0.30028
17	0.12235	0.12235	1.66437	0.68001	0.28725
22	0.13680	0.13680	1.99606	0.69979	0.27260

The estimates for the first two components are the same. We have rounded the estimates in the table to five decimal places; however, the estimates are identical up to 32 significant digits. This means the algorithm is fitting a two mixed-Exponential model. However, we are losing precision in our estimates since we are estimating unnecessary parameters. Looking back at table 3.1 from the two-mixed-Exponential model, corresponding values of τ have similar estimates for the number of unobserved classes; however, the two mixed-Exponential model's estimate has a lower standard error.

Figure 4.2 shows the fitted values for the microbial data at cut-off percentage 98 ($\tau = 16$) and table 4.3 shows the results. Table 4.4 shows parameter estimates

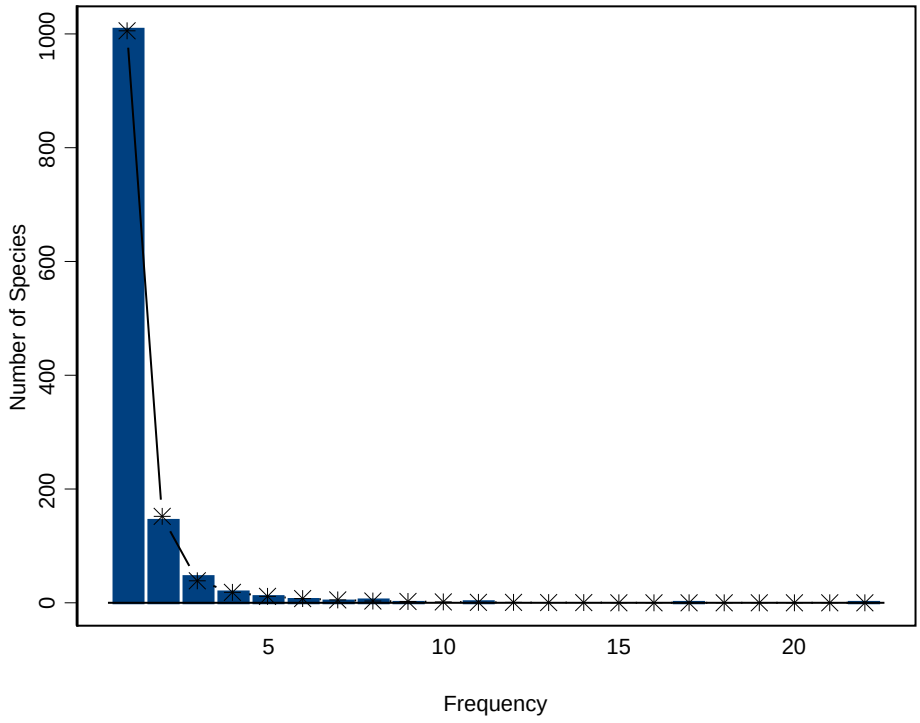

Figure 4.1: Fitted Values with Microbial 99% ($\tau = 22$)

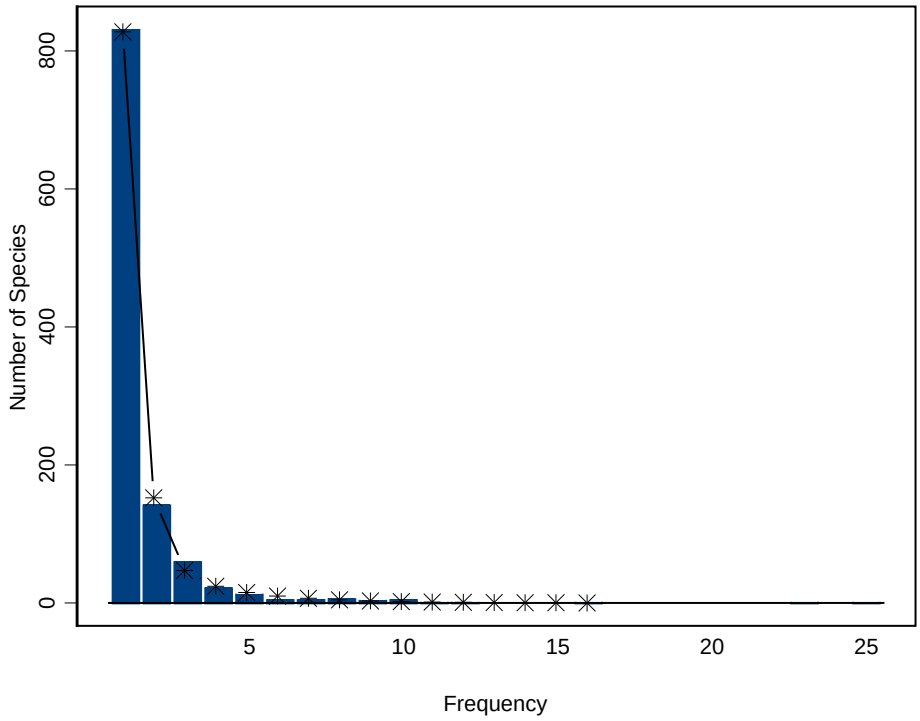

Figure 4.2: Fitted Values with Microbial 98% ($\tau = 16$)

for the microbial 98% data. As τ increases, the estimates for θ_1 and θ_2 become distinct. As we use more data, we are able to fit more components.

Tables 4.5, 4.6, 4.7, 4.8, 4.9, and 4.10, show results for all other cutoff percentages for the microbial data.

Table 4.3: Results for Microbial 98%

τ	$\hat{C}$	$SE(\hat{C})$	*GOF p-value*
7	3,772,351.4	NA	NA
8	9,425.4	NA	0.0076
9	8,142.0	2,149.7	0.0055
10	7,209.0	35,540.0	0.0013
11	7,082.3	31,242.0	0.0030
12	6,893.9	26,053.5	0.0045
16	6,666.1	20,603.1	0.0243
23	10,994.7	15,838.9	0.2193
25	11,050.5	13,836.0	0.2473

Table 4.4: Parameter Estimates for Microbial 98%

τ	$\hat{\theta}_1$	$\hat{\theta}_2$	$\hat{\theta}_3$	$\hat{\theta}_4$	$\hat{\theta}_5$
7	0.00013	0.00013	0.75922	0.65655	0.34310
8	0.08892	0.08892	1.11111	0.59783	0.32679
9	0.11364	0.11365	1.34384	0.60661	0.32603
10	0.13992	0.14030	1.69443	0.61921	0.32373
11	0.14435	0.14488	1.77461	0.64117	0.30373
12	0.15089	0.15185	1.87660	0.64144	0.30566
16	0.15986	2.03478	0.16105	0.67288	0.04954
23	0.06197	0.55887	3.06595	0.87978	0.10670
25	0.06186	0.60542	3.71956	0.88451	0.10508

Table 4.5: Results for Microbial 70%

τ	$\hat{C}$	$SE(\hat{C})$	*GOF p-value*
22	198.8	1,482.5	NA
53	204.6	1,550.8	NA
91	234.6	2,114.6	NA
339	213.6	1,676.7	NA

Table 4.6: Results for Microbial 80%

τ	$\hat{C}$	$SE(\hat{C})$	*GOF p-value*
7	792.8	1,933.4	NA
8	736.6	10,996.2	NA
9	680.1	6,300.3	NA
11	640.4	28,771.8	NA
12	573.5	12,323.9	NA
13	537.7	7,292.3	NA
14	496.1	3,718.8	NA
15	478.2	2,666.5	NA
16	466.5	2,219.1	NA
20	453.7	1,780.5	NA
22	512.7	548.4	0.0202
23	538.5	665.1	0.0182
25	535.5	632.2	0.0218
27	544.0	653.8	0.0252
31	551.7	668.3	0.0362
34	559.0	675.7	0.0433
38	565.5	677.9	0.1232
56	573.9	654.8	0.1088
76	575.6	592.4	0.1779
81	571.0	528.3	0.2511
102	555.8	431.1	0.2070
115	428.9	120.0	0.2012
130	410.4	100.0	0.2010
264	394.8	86.3	0.2014
364	387.8	81.2	0.1877

Table 4.7: Results for Microbial 90%

τ	$\hat{C}$	$SE(\hat{C})$	*GOF p-value*
7	10,050,106.2	3,170.1	NA
8	2,834.5	83,226.7	NA
9	2,352.9	18,040.7	0.0080
10	2,125.6	7,458.1	0.0019
11	2,096.2	6,628.6	0.0057
12	2,029.6	5,109.5	0.0084
13	1,975.5	4,072.6	0.0166
15	2,223.5	3,299.5	0.0403
16	2,481.8	6,064.4	0.0538
17	3,303.0	18,917.6	0.0688
19	2,918.5	9,765.8	0.1662
20	3,582.4	26,157.1	0.1811
25	2,953.2	7,747.6	0.2195
28	2,880.1	6,256.2	0.2984
34	2,994.9	5,751.9	0.3162
36	3,300.7	6,798.6	0.1109
41	1,877.3	303.6	0.2353
45	1,876.5	298.2	0.2748
49	1,875.9	294.2	0.2735
52	1,876.0	291.8	0.2681
53	1,877.1	291.4	0.2542
62	1,875.9	288.4	0.1809
96	1,861.6	275.3	0.2791
123	1,859.7	267.8	0.3235

Table 4.8: Results for Microbial 95%

τ	$\hat{C}$	$SE(\hat{C})$	*GOF p-value*
7	5,718.8	NA	NA
8	4,902.0	49,106.2	0.0006
9	4,774.3	38,234.3	0.0041
10	4,378.5	21,723.8	0.0113
11	4,168.2	14,973.6	0.0153
12	4,040.4	11,878.7	0.0209
13	3,813.9	7,704.0	0.0449
14	3,778.0	7,171.4	0.0308
18	3,713.8	6,298.1	0.0989
19	5,602.8	15,385.3	0.2021
20	5,538.0	14,895.1	0.2338
21	5,620.0	14,049.7	0.3049
25	5,655.3	10,193.6	0.3781
28	5,641.3	8,118.3	0.5296
30	5,501.3	6,113.4	0.3699
33	5,033.9	3,371.1	0.1310
37	3,989.4	811.1	0.2311
41	3,862.7	647.3	0.0641
58	3,777.7	548.2	0.0889

Table 4.9: Results for Microbial 96%

τ	$\hat{C}$	$SE(\hat{C})$	*GOF p-value*
7	4,715.2	NA	NA
8	4,301.1	17,113.5	0.0008
9	4,291.3	15,520.3	0.0131
10	4,096.9	10,957.1	0.0463
11	3,935.2	7,757.2	0.0689
12	3,843.0	6,317.2	0.1741
13	3,779.9	5,515.2	0.2516
18	3,731.9	4,898.6	0.3568
20	3,674.3	3,989.7	0.4049
21	3,877.1	3,230.2	0.4502
22	4,063.5	3,624.4	0.4453
25	4,231.5	4,039.2	0.4750
28	4,385.6	4,047.5	0.4705
33	3,714.8	481.9	0.3664
37	3,725.7	473.8	0.3564
44	3,723.4	462.2	0.1227

Table 4.10: Results for Microbial 97%

τ	$\hat{C}$	$SE(\hat{C})$	*GOF p-value*
7	6,781.0	NA	NA
8	6,162.4	26,646.1	0.0023
10	NA	NA	NA
11	5,106.5	15,334.6	0.0502
12	5,008.0	13,202.4	0.0578
13	4,680.3	7,856.9	0.0042
14	6,483.8	19,277.8	0.0150
15	6,705.7	18,363.9	0.0214
20	7,047.5	15,920.2	0.0556
21	7,532.4	14,425.2	0.1247
24	6,982.8	7,502.0	0.1316
25	6,510.2	4,198.5	0.1222

Chapter 5

Discussion and Additional Topics

5.1 General Mixtures of Exponentials

Increasing the number of components in the model explores more complicated finite mixtures of Exponentials. It is easy to extend the model from chapter 4. For instance, a four-mixed-Exponential mixed Poisson model could be parameterized as

$$\begin{aligned} P[Y_j = y_j] &= \theta_5 \left(\frac{\theta_1}{1+\theta_1}\right)^{y_j} \left(\frac{1}{1+\theta_1}\right) + \theta_6 \left(\frac{\theta_2}{1+\theta_2}\right)^{y_j} \left(\frac{1}{1+\theta_2}\right) \\ &\quad +\theta_7 \left(\frac{\theta_3}{1+\theta_3}\right)^{y_j} \left(\frac{1}{1+\theta_3}\right) \\ &\quad +(1-\theta_5-\theta_6-\theta_7) \left(\frac{\theta_4}{1+\theta_4}\right)^{y_j} \left(\frac{1}{1+\theta_4}\right), \end{aligned}$$

where θ_1, θ_2, θ_3, and θ_4 are component parameters and θ_5, θ_6, and θ_7 are weight parameters. This model has seven parameters, and may be an overfit for frequency data shown in figure 1.1. As seen with using the three mixed-Exponential distribution in section 4.3, three components is more than enough to model some data.

Feldmann and Whitt (1998 [16]) show that any monotone pdf can be ap-

proximated by a finite mixture of Exponentials. Thus, mixtures of Exponentials is a reasonable and flexible choice for the distribution of class means. They also illustrate that this is especially useful in modelling heavy tailed data without some of the mathematical complications of distributions such as Pareto and Weibull.

Example

For the three models proposed in this paper (zero-truncated Geometric, zero-truncated two-mixed-Geometric, and zero-truncated three-mixed-Geometric) the two-mixed-Geometric performs best in terms of goodness of fit and precision of the estimates. Contrasting tables 2.1, 3.1, and 4.1, we see that the Geometric model does not fit the data well. The two and three mixed-Geometric models fit the data well, although the estimates from the two-mixed-Geometric model have a smaller standard error.

Among the analyses from the two-mixed-Geometric model we also can choose a value for τ in order to attain better model fit. At $\tau = 22$, the goodness of fit p-value is 0.6493. Since it is possible to model all of the data, we will choose this largest value of τ. Thus we estimate 9063.0 existing microbial species based on a 99% sequence similarity cut-off percentage. This estimate has a standard error of 1127.4. Hence, a normal confidence interval would be $(6853.296, 11272.7)$. We use the normal approximation based on Sanathanan's results (1972 [27]).

5.2 Other Mixtures

In generalizing the model, a first step would be to use mixtures of Gamma distributions for the distribution of Poisson means. However, there is no restriction on the form of the distributions. Mixtures of Exponential distributions were

chosen for their simplicity.

When using mixture distributions to model the Poisson means, a useful result for more complicated components, shown by Böhning and Kuhnert (2005 [2]), is that each zero-truncated mixture distribution can be written as a mixture of zero-truncated count distributions. This is a useful result since a mixture of zero-truncated count distributions usually has a more tractable analytic form. The zero-truncated mixture distribution is often more complicated due to the denominator in equation 1.2. Writing the model as a mixture of zero-truncated count distributions eliminates the need for a nested EM, and a single stage EM can be used on the new finite mixture model. However, maximizing the likelihood in the M step may still be difficult depending on the form of the zero-truncated components.

5.3 Estimating Number of Components

A usual way of comparing models is using a likelihood ratio test statistic ,

$$2\{\log L(\hat{\theta}_1) - \log L(\hat{\theta}_0)\}, \tag{5.1}$$

where Ł($\hat{\theta}_1$) and Ł($\hat{\theta}_0$) are the likelihoods given the MLE estimates under an alternative and null model, respectively. However, due to the non-identifiability of the parameters in the mixture model, the usual distributional results do not hold. However, there are some ways to modify the likelihood ratio test statistic as described in McLachlan and Peel (2000 [23] section 6.5).

Another approach is by a parametric bootstrap (McLachlan and Peel 2000 [23]). Bootstrap samples can be generated under the null model with the maximum likelihood parameter estimates. The value of $2\{\log L(\hat{\theta}_1) - \log L(\hat{\theta}_0)\}$ can be calculated for each bootstrap sample, and the original value of $2\{\log L(\hat{\theta}_1) -$

$\log L(\hat{\theta}_0)\}$ from the data can be assessed by seeing where it lies in the distribution of the statistic from the bootstrap samples. One disadvantage of this approach is that it takes a substantial amount of computation time.

Other common methods use Akaike's Information Criterion, Cross Validation Based Information Criterion, and Schwarz's Bayesian Information Criterion. These criteria are often used to evaluate the number of components in a mixture model, even though they often do not meet regularity conditions, often due to unidentifiable parameters.

5.4 A Fully Bayesian Model

Bayesians may categorize the model proposed in this paper as an empirical Bayes model. The Exponential distribution is used as a prior on the nuisance parameters, λ_j, for $j = 1, \ldots, C$. Additionally, if we specify a prior distribution for the unknown parameter, C, then the model can be viewed as a fully Bayesian model.

Rodrigues, Milan, and Leite (2001 [25]) consider a noninformative and a Poisson prior for C. A Gamma prior is considered for the distribution of λ_j. Lewins and Joanes (1984 [21]) specify a zero-truncated negative binomial distribution for the number of classes. The distribution of the abundance proportions, given the number of classes, is described with a Dirichlet distribution. Solow (1994 [30]) discusses several possibilities for the distribution of C including a noninformative prior, a zero-truncated negative binomial distribution, and a sequential broken stick model. The Dirichlet distribution is used as a prior on the abundance proportions.

Rodrigues, Milan, and Leite (2001 [25]) compare the empirical Bayes ap-

proach with the fully Bayesian approach. They describe how the empirical Bayes approach approximates to the fully Bayesian approach, with a noninformative prior, for large values of C. They cite (Berger *et al.* 1999 [1]) for discussion on choosing between the two approaches. The authors point out that the use of noninformative priors gives justification to Sanathanan's (1972 [27]) asymptotic results.

5.5 Model Assumptions

We assume that the number of individuals each class contributes to the sample is from a mixture of Geometric random variables, and that each of these random variables is independent of each other. In terms of the mixed Poisson model, we assume each class contributes a number of individuals to the sample independently of all other classes. However, this may not be true in applied situations. Consider a biological example where the numbers of individuals from two species are correlated due to symbiotic relationships between the species. In a predator-prey or competitive relationship, the numbers of individuals in the sample may be negatively correlated. When the existence of one species benefits the other, the numbers may be positively correlated. Also, sampling techniques may invalidate the independence assumption.

Another important assumption is that the frequency distribution is monotonically decreasing. Other distributions such as the negative binomial model from section 1.3.1 are able to have a non-zero mode, showing more flexibility in the abundance distribution.

5.6 Use of Other Computer Software

The computer code appears in the Appendix and currently runs with Maple version 10. The use of Maple is helpful for other mixed Poisson models which have more complicated forms than a mixture of Exponentials. Because of the simplicity of the algorithm and simple form of the Exponential mixed Poisson probability density function, it is possible for the code to be ported to other user friendly programs, such as Microsoft Excel.

Appendix A

Maple Code

This appendix contains Maple code to fit the model for mixtures of two and three exponential distributions. In the version presented here, Aitken's acceleration is used in both the two-mixed-Exponential and three mixed-Exponential models. This program has been used in the data analysis for Hong *et al.* (2006 [19]).

The analysis described in this paper can be used to estimate the number of unobserved classes for any frequency data set. The user only needs to save the data in an appropriate format and fill in the appropriate program specifications labeled "user_defined" in the code below. The frequency data is read in as a two column tab delimited file with frequencies in the first column and number of classes observed of that frequency in the second column. This data format represents the distribution of the number of individuals observed from each class. The other specifications that are defined by the user include the following.

1. data_file: Path name indicating location of the data file

2. output_fits_file: Path name indicating location where fits file is to be saved (See below for content of fits file.)

3. output_analysis_file: Path name indicating location where analysis file is to be saved (See below for content of analysis file.)

4. f_min: Smallest value of τ for analysis

5. f_max: Largest value of τ for analysis

6. Digits: Number of significant digits used (Default= 16)

7. mle_nonzt_iter: Maximum number of iterations for both outer and inner EM (Default= 5000)

8. mle_nonzt_tol: Convergence tolerance for EM in negative powers of ten (Default= 6 indicating $\varepsilon = 10^{-6}$)

9. std_err_tol: Convergence tolerance for standard error series computation in negative powers of ten (Default= 2)

The program creates two output files. One is an analysis file containing all of the relevant summary statistics. The second is a fits file which contains fitted values for the model. The output of the program contains several fields many of which are not of direct interest in this report. For completeness, we will define all fields in the program output.

In the analysis file the following statistics are reported from a single analysis:

1. Tuning parameter: τ

2. Maximum likelihood parameter estimates, $\hat{\theta}$: $\hat{\theta}_1 =$"t1", $\hat{\theta}_2 =$"t2", ...

3. Estimated non-coverage: $P[Y_i = 0|\hat{\theta}]$

4. Estimate for the number of unobserved classes: $\hat{C}_0$

5. Estimate for the total number of classes based on the subset of the data defined by the tuning parameter: $\hat{C}$

6. Estimate for the total number of classes based on full data:

$$\hat{C} + \sum I[X_i > \tau]$$

7. Asymptotic standard error of $\hat{C}$: $SE(\hat{C})$

8. Lower bound for standard error of $\hat{C}_0$

$$\left(\hat{C}\frac{p_0\,(0)}{1-p_0\,(0)}\right)^{1/2}$$

9. Naïve goodness of fit statistic (non-binned cells)

10. Asymptotically correct goodness of fit statistic (binned cells)

11. Program error report: Error is 1 if either EM loop has reached maximum number of iterations; 0 otherwise.

The second output file contains the fitted values for each value of the tuning parameter used. The first column contains the values of X_i from the observed data. The second column contains the observed frequency counts O_k, for $k = 1, \ldots, K = \max X_i$. The following columns contain the fitted values for ascending values of the tuning parameter.

A.1 Two-Mixed-Exponential

```
# DEFINE DATA and OUTPUT FILES
data_file:="user_defined";
output_fits_file:="user_defined";
output_analysis_file:="user_defined";

#DEFINE FREQUENCY RANGE FOR ANALYSIS
# minimum frequency for analysis
f_min:=user_defined;
# maximum frequency for analysis
```

```
f_max:=user_defined;


# significant digits for computations
Digits:=user_defined;

# maximum iterations for EM algorithm
mle_nonzt_iter:=user_defined;
# convergence tolerance for EM algorithm
mle_nonzt_tol:=user_defined;

# convergence tolerance for standard error series computation
std_err_tol:=user_defined;

# SET MAPLE INTERFACE SCREEN DISPLAY DIMENSION
interface(rtablesize=infinity):



### ANALYTICAL MATH
# Ordinary mixed exponential mixed poisson density
p:=t3*(t1/(1+t1))^j*1/(1+t1)+(1-t3)*(t2/(1+t2))^j*1/(1+t2):

# probability j=0
p0:=eval(p,j=0):

# Zero-truncated mixed exponential mixed poisson
p_zt:=p/(1-p0):

# Pre-information matrix and vectors for ordinary mixed expl
p_t1t1:=-simplify(diff(diff(ln(p),t1),t1)):
p_t1t2:=-simplify(diff(diff(ln(p),t1),t2)):
p_t1t3:=-simplify(diff(diff(ln(p),t1),t3)):
p_t2t2:=-simplify(diff(diff(ln(p),t2),t2)):
p_t2t3:=-simplify(diff(diff(ln(p),t2),t3)):
p_t3t3:=-simplify(diff(diff(ln(p),t3),t3)):
v_t1:=simplify(diff(p0,t1)):
v_t2:=simplify(diff(p0,t2)):
v_t3:=simplify(diff(p0,t3)):


### DATA INPUT
entrydata:=ImportMatrix(data_file):
```

```
rowdim:=LinearAlgebra[RowDimension](entrydata):
fmaxdata:=entrydata[rowdim,1]:
freqdata:=<<seq(i,i=1..fmaxdata)>|<seq(0,i=1..fmaxdata)>>:
for n from 1 to fmaxdata do:
  for m from 1 to rowdim do:
    if entrydata[m,1]=n then freqdata[n,2]:=entrydata[m,2]: break:
    fi:
  od:
od:
unassign('n'): unassign('m'):

#print(freqdata):


### SET FMIN & FMAX FOR ANALYSIS
fmin:=max(5,f_min);
fmax:=min(fmaxdata,f_max);


### SET UP LOOP ON CUTOFFS OF INTEREST
for t from fmin to fmax do:
  if freqdata[t,2]=0 then next
  fi:

  # SET ERROR CONDITIONS
  tracking[t]:=0:

  # BASIC STATISTICS AT UPPER NONEMPTY CUTOFFS FROM FMIN TO FMAX
  s[t]:=add(freqdata[i,2],i=1..t):
  sum1:=evalf(add(freqdata[i,1]*freqdata[i,2],i=1..t)):

  # initial value for s0
    # first find geometric parameter based on
    # first four frequencies

      # create raw data
      raw_data:=\
         <<[seq(1,i=1..freqdata[1,2]),seq(2,i=1..freqdata[2,2]),\
         seq(3,i=1..freqdata[3,2]),seq(4,i=1..freqdata[4,2])]>>;

      # mle for right and left truncated geometric
      geom_par:=fsolve((-1*add(raw_data[i,1],\
```

```
       i=1..LinearAlgebra[RowDimension](raw_data))/(1-par))\
       +LinearAlgebra[RowDimension](raw_data)*\
       add(j*(1-par)^(j-1),j=1..4)/add((1-par)^j,j=1..4),par);
    s0:=(geom_par/(add(geom_par*(1-geom_par)^j,j=0..4))\
       *add(raw_data[i,1],i=1..LinearAlgebra[RowDimension]\
       (raw_data)))/\
       (1-geom_par/(add(geom_par*(1-geom_par)^j,j=0..4)));

# initialize values for t1, t2, t3
t1_init:=1:t2_init:=1:t3_init:=1/2:


### append s0 to freqdata to make a new full data set
fulldata:=<<seq(x,x=0..t)>|<[s0,seq(0,x=1..t)]>>;
for x from 1 to t do:
  fulldata[x+1,2]:=freqdata[x,2]:

od:
print("full data",fulldata);

### obtain starting values for t1, t2
## subset fulldata for low frequencies
low_freq:=LinearAlgebra[SubMatrix]\
  (fulldata,1..floor(2*(t+1)/3)+1,1..2);
# print("low_freq",low_freq);

# fist find sample size
n_low_freq:=add(low_freq[i,2],i=1..floor(2*(t+1)/3)+1);

# next calculate sample mean
low_freq_mean:=evalf(add(low_freq[i,1]*low_freq[i,2],\
  i=1..floor(2*(t+1)/3)+1)/n_low_freq);

# calculate geometric success probability, which is 1/mean
geom_success_prob:=1/(low_freq_mean+1);

# calculate starting value for t1 using mle of low_feq data
t1_init:=(1-geom_success_prob)/geom_success_prob;


## subset freqdata for high frequencies
high_freq:=LinearAlgebra[SubMatrix]\
```

```
  (fulldata,floor(1*(t+1)/3)+1..t+1,1..2);
# print("high_freq",high_freq);

# find row dim of high_freq
high_freq_dim:=LinearAlgebra[RowDimension](high_freq);

# fist find sample size
n_high_freq:=add(high_freq[i,2],i=1..high_freq_dim);

# next calculate sample mean
high_freq_mean:=evalf(add(high_freq[i,1]*high_freq[i,2],\
  i=1..high_freq_dim)/n_high_freq);

# calculate geometric success probability
geom_success_prob:=1/(high_freq_mean+1-(floor(1*(t+1)/3)-1));

# calculate starting value for t2 using mle of high_feq data
t2_init:=(1-geom_success_prob)/geom_success_prob;


## define mles with starting values
mles:={t1=t1_init,t2=t2_init,t3=t3_init}:
print("init mles",mles);

### initialize number of consecutive em iterations
ncem:=0;

### start loop for imputed zero count

for h from 1 to mle_nonzt_iter do;

## append s0 to freqdata to make a new full data set
fulldata:=<<seq(x,x=0..t)>|<[s0,seq(0,x=1..t)]>>;
for x from 1 to t do:
  fulldata[x+1,2]:=freqdata[x,2]:

od:
# print("full data",fulldata);


##### start a new loop(i) here for EM for non-zt data
for i from 1 to mle_nonzt_iter do:
```

```
##
## E step
##

# initialize imputed values vectors
z1_impute:=<<seq(0,k=1..t+1)>>;
z2_impute:=<<seq(0,k=1..t+1)>>;

# define geometric density
geom_den:=(theta/(1+theta))^(j)*(1/(1+theta));

for w from 0 to t do:

  # calculate pr(z=1|x=w) =
  # pr(x=w|z=1)pr(z=1)/[pr(x=w|z=1)pr(z=1)+pr(x=w|z=0)pr(z=0)]
  z1_impute[w+1,1]:=subs(j=w,theta=eval(t1,mles),geom_den)\
    *eval(t3,mles)/\
    (\
    subs(j=w,theta=eval(t1,mles),geom_den)*eval(t3,mles)+\
    subs(j=w,theta=eval(t2,mles),geom_den)*(1-eval(t3,mles))
    );
od; unassign('w');

# use constraint to define z2
for w from 0 to t do:
  z2_impute[w+1,1]:=1-z1_impute[w+1,1];
od; unassign('w');


# print imputed values for each iteration of EM
# print("z1_impute",z1_impute);
# print("z2_impute",z2_impute);


##
## M step
##

### calculate t3 estimate
# initialize vector of products; we will weight each
```

```
# imputed value by its count
z1_count_prod:=<<seq(0,i=1..t+1)>>;
for r from 0 to t do;
  z1_count_prod[r+1,1]:=z1_impute[r+1,1]*fulldata[r+1,2];
od;
unassign('r');
# print("z1 count prod",z1_count_prod);

z2_count_prod:=<<seq(0,i=1..t+1)>>;
for r from 0 to t do;
  z2_count_prod[r+1,1]:=z2_impute[r+1,1]*fulldata[r+1,2];
od;
unassign('r');
# print("z2 count prod",z2_count_prod);


# sum the z_count_prod vector and divide by sample size(s)
# then solve for t3
t3_new:=add(z1_count_prod[i,1],i=1..t+1)/(s[t]+s0);


### now estimate t1, t2 given z_impute

# closed form for estimates of t1, t2 derived from
# eq. (2.33) McLachlan
t1_new:=add(z1_count_prod[j,1]*(fulldata[j,1]),j=1..t+1)/\
  add(z1_count_prod[j,1],j=1..t+1);
t2_new:=add(z2_count_prod[j,1]*(fulldata[j,1]),j=1..t+1)/\
  add(z2_count_prod[j,1],j=1..t+1);


# define mles_try as current iteration mles
mles_try:={t1=t1_new,t2=t2_new,t3=t3_new};

# print mles_try at each iteration
# print(i,"mle estimates","t1=",eval(t1,mles),\
# "t2=",eval(t2,mles),"t3=",eval(t3,mles));

# break for loop if convergence is reached
if    eval(t1,mles_try)/eval(t1,mles) > 1-10^(-mle_nonzt_tol)
  and eval(t1,mles_try)/eval(t1,mles) < 1+10^(-mle_nonzt_tol)
  and eval(t2,mles_try)/eval(t2,mles) > 1-10^(-mle_nonzt_tol)
```

```
    and eval(t2,mles_try)/eval(t2,mles) < 1+10^(-mle_nonzt_tol)
    and eval(t3,mles_try)/eval(t3,mles) > 1-10^(-mle_nonzt_tol)
    and eval(t3,mles_try)/eval(t3,mles) < 1+10^(-mle_nonzt_tol)
    then break;
  fi;

  # if mles not converged, update parameter estimates
  mles:=mles_try;


  # track error if EM step reaches max # of iterations
  if i=mle_nonzt_iter+1
    then tracking[t]:=1;
    fi;

# end EM loop
od;

#### Impute a new value for s0
s0_new:=evalf(eval(p0,mles))*(s[t]+s0);
print(t,h,"s0_new",s0_new);

# if new s0 has converged, break out of loop
    if  s0/s0_new > 1-10^(-mle_nonzt_tol)
    and s0/s0_new < 1+10^(-mle_nonzt_tol)
    then break;
  fi;


### keep track of number of consecutive em iterations
ncem:=ncem+1;
print("ncem",ncem,"so_old",s0_old,"s0",s0,"s0_new",s0_new);

#### acceleration step


print("max i",i);
if h>10 and abs(s0-s0_old)>abs(s0_new-s0) and ncem>1
  then s0_step:=s0_new-s0;
  num_step:=(s0_new-s0)/((s0-s0_old)-(s0_new-s0));
  print("s0_step",s0_step,"num_step",num_step);
  s0_new:=s0_new+num_step*s0_step;
```

```
    # reset nem
    ncem:=0;
    fi;

  #### end of acceleration step


  ### save s0 as old s0
  s0_old:=s0;

  ### save new s0 if no convergence
  s0:=s0_new;

  # track error if s0 loop reaches max # of iterations
    if h=mle_nonzt_iter+1
      then tracking[t]:=2;
      fi;

## end outer loop
od;

  # save converged mles
  mles_final[t]:=mles;
  print("mles final","t1=",eval(t1,mles_final[t]),\
    "t2=",eval(t2,mles_final[t]),"t3=",eval(t3,mles_final[t]));


  ### COMPUTE FITTED VALUES
  # we will export these in the fits file later
  for j from 1 to t do;
    fitzt[j,t]:=evalf(s[t]*eval(p_zt,mles_final[t]));
  od;
  unassign('j');


  ### GOODNESS OF FIT BASED ON FINAL MLES

  accum_obs:=0: accum_fit:=0:
  # k is concatenated cell number
  k:=0:

  # add fits and observations for each frequency
```

```
for j from 1 to t do:
  accum_obs:=accum_obs+freqdata[j,2]:
  accum_fit:=accum_fit+fitzt[j,t]:
  # if fits >=5 then assign to cell[k]
  if accum_fit >= 5 then
    k:=k+1: cell_obs[k]:=accum_obs:
    cell_fit[k]:=accum_fit: j_stop:=j: k_stop:=k:
    accum_obs:=0: accum_fit:=0:

    # print concatenated cells with fits >= 5
    # print("k",k,"cell_fit[k]",cell_fit[k]):
  fi:
od:

# if fits in last cells do not sum to 5,
# then add them to the previous cell
if j_stop < t then
  cell_obs[k]:=cell_obs[k]+add(freqdata[l,2],l=j_stop+1..t):
  cell_fit[k]:=cell_fit[k]+add(fitzt[l,t],l=j_stop+1..t):
fi:
unassign('j'):
unassign('k'):

# print all concatenated cells with observed values
for k from 1 to k_stop do:
  print("k",k,"cell_fit[k]",cell_fit[k],\
    "cell_obs[k]",cell_obs[k]):
od:

# print freq where last fits>=5, and number of
# concatenated cells
# print("j_stop",j_stop,"k_stop",k_stop):
unassign('k'):

# calculate chi-sq statistic and p-value for concatenated cells
if k_stop>4 then
  chistat_cell:=add((cell_obs[k]-cell_fit[k])^2/\
    cell_fit[k],k=1..k_stop) + s[t]-add(fitzt[i,t],i=1..t);
  GOF_mles_cell[t]:=\
    evalf(1-stats[statevalf,cdf,chisquare[k_stop+1-3-1]]\
    (chistat_cell));
```

```
    # print GOF chi-sq statistic and p-value
    print(t,"cells chi-sq stat", chistat_cell,\
      "concatenated cells GOF",GOF_mles_cell[t]):
  fi:

  # calculate (naive) chi-sq statistic and p-value for all cells
  chistat_all:=add((freqdata[i,2]-fitzt[i,t])^2/\
    fitzt[i,t],i=1..t)+s[t]-add(fitzt[i,t],i=1..t);
  GOF_mles_all[t]:=\
    evalf(1-stats[statevalf,cdf,chisquare[t+1-3-1]](chistat_all)):

  # print GOF (naive) chi-sq stat and p-value
  print(t,"all cells chi-sq stat",chistat_all,\
    "all cells GOF",GOF_mles_all[t]):



### SET UP ORDINARY NON-ZERO-TRUNCATED INFORMATION MATRIX,
### ASSOCIATED VECTOR, & EMPIRICAL QUANTITIES
   p_mles_final[t]:=eval(p,mles_final[t]):
   p0_mles_final[t]:=evalf(eval(p0,mles_final[t])):
   s0_mles_final[t]:=evalf(s[t]*p0_mles_final[t]/\
     (1-p0_mles_final[t])):
   s_hat[t]:=s[t]+s0_mles_final[t]:
   p_t1t1_mles_final:=eval(p_t1t1,mles_final[t]):
   p_t1t2_mles_final:=eval(p_t1t2,mles_final[t]):
   p_t1t3_mles_final:=eval(p_t1t3,mles_final[t]):
   p_t2t2_mles_final:=eval(p_t2t2,mles_final[t]):
   p_t2t3_mles_final:=eval(p_t2t3,mles_final[t]):
   p_t3t3_mles_final:=eval(p_t3t3,mles_final[t]):
   v_t1_mles_final:=eval(v_t1,mles_final[t]):
   v_t2_mles_final:=eval(v_t2,mles_final[t]):
   v_t3_mles_final:=eval(v_t3,mles_final[t]):

### EVALUATE STANDARD ERROR
   inf_t1t1_mles_final:=0:
   inf_t1t2_mles_final:=0:
   inf_t1t3_mles_final:=0:
   inf_t2t2_mles_final:=0:
   inf_t2t3_mles_final:=0:
   inf_t3t3_mles_final:=0:
   std_err_try:=0:
```

```
  std_err[t]:=1:
  for j from 0 to 1000 do;
     inf_t1t1_mles_final:=inf_t1t1_mles_final+\
       evalf(p_mles_final[t]*p_t1t1_mles_final):
     inf_t1t2_mles_final:=inf_t1t2_mles_final+\
       evalf(p_mles_final[t]*p_t1t2_mles_final):
     inf_t1t3_mles_final:=inf_t1t3_mles_final+\
       evalf(p_mles_final[t]*p_t1t3_mles_final):
     inf_t2t2_mles_final:=inf_t2t2_mles_final+\
       evalf(p_mles_final[t]*p_t2t2_mles_final):
     inf_t2t3_mles_final:=inf_t2t3_mles_final+\
       evalf(p_mles_final[t]*p_t2t3_mles_final):
     inf_t3t3_mles_final:=inf_t3t3_mles_final+\
       evalf(p_mles_final[t]*p_t3t3_mles_final):
     infomat_mles_final:=evalf(Matrix(3,3,[\
        [inf_t1t1_mles_final,inf_t1t2_mles_final,\
          inf_t1t3_mles_final],\
        [inf_t1t2_mles_final,inf_t2t2_mles_final,\
          inf_t2t3_mles_final],\
        [inf_t1t3_mles_final,inf_t2t3_mles_final,\
          inf_t3t3_mles_final]])):
     v_mles_final:=evalf(Matrix(3,1,[[v_t1_mles_final],\
       [v_t2_mles_final],[v_t3_mles_final]])):
     if LinearAlgebra[Rank](infomat_mles_final) = 3
        then std_err_try:=evalf(sqrt(s_hat[t]*p0_mles_final[t])*\
          (1-p0_mles_final[t]-(1/p0_mles_final[t])*LinearAlgebra\
          [Multiply](LinearAlgebra[Multiply]\
          (LinearAlgebra[Transpose](v_mles_final),\
        LinearAlgebra[MatrixInverse]\
          (infomat_mles_final)),v_mles_final)[1,1])^(-1/2))
        else next fi:
     if std_err_try>0
        and std_err_try/std_err[t] > 1-10^(-std_err_tol)
        and std_err_try/std_err[t] < 1+10^(-std_err_tol)\
        then break
        else std_err[t]:=std_err_try: next: fi:
  od:

# print(t,j):
 unassign('j'):
```

```
# end t loop
od;


### OUTPUT FITTED VALUES

# first, find max non-zero frequency in interval (fmin,fmax)
nonzero:=0:
for u from fmin to fmax do:
  if freqdata[u,2]=0 then next
  fi:
  # nonzero is the max non-zero frequency in interval (fmin,fmax)
  nonzero:=nonzero+1:
od:
unassign('u'):

fits:=Matrix(fmax,nonzero+2):

for w from 1 to fmax do:
  # first column lists frequencies
  fits[w,1]:=w:
  # second column lists counts
  fits[w,2]:=freqdata[w,2]:
od:

# rest of columns fill in fitted values for non-zero t values
u:=0:
for z from fmin to fmax do:
  if freqdata[z,2]=0 then next
  fi:
  u:=u+1:
  for v from 1 to z do:
    fits[v,u+2]:=fitzt[v,z]:
  od:
od:
unassign('w'):
unassign('u'):
unassign('v'):
unassign('z'):

# print fitted values
print("fits",fits):
```

```
ExportMatrix(output_fits_file,fits):


### OUTPUT ANALYSIS FILE
results:=Matrix(nonzero,11):
u:=0:
for t from fmin to fmax do:
  if freqdata[t,2]=0 then next
  fi:
  u:=u+1:
  # right truncation point
  results[u,1]:=t:

  # final mle estimates
  results[u,2]:=mles_final[t]:

  # non-coverage
  results[u,3]:=p0_mles_final[t]:

  # estimate of unobserved species
  results[u,4]:=s0_mles_final[t]:

  # estimate of total species based on subset
  results[u,5]:=s_hat[t]:

  # estimate of total species based on full data
  results[u,6]:=add(freqdata[i,2],i=1..fmaxdata)\
    +s0_mles_final[t]:

  # standard error
  results[u,7]:=std_err[t]:

  # lower bound for standard error
  results[u,8]:=sqrt(s_hat[t]*p0_mles_final[t]/\
    (1-p0_mles_final[t])):

  # naive GOF
  results[u,9]:=GOF_mles_all[t]:

  # concatenated GOF
  results[u,10]:=GOF_mles_cell[t]:
```

```
  # error tracking
  results[u,11]:=tracking[t]:
od:

# print analysis file
print(results):

# export analysis file
ExportMatrix(output_analysis_file, results):
```

A.2 Three-Mixed-Exponential

```
# DEFINE DATA and OUTPUT FILES
data_file:="user_defined";
output_fits_file:="user_defined";
output_analysis_file:="user_defined";

#DEFINE FREQUENCY RANGE FOR ANALYSIS
# minimum frequency for analysis
f_min:=user_defined;
# maximum frequency for analysis
f_max:=user_defined;


# significant digits for computations
Digits:=user_defined;

# maximum iterations for EM algorithm
mle_nonzt_iter:=user_defined;
# convergence tolerance for EM algorithm
mle_nonzt_tol:=user_defined;

# convergence tolerance for standard error series computation
std_err_tol:=user_defined;

# SET MAPLE INTERFACE SCREEN DISPLAY DIMENSION
interface(rtablesize=infinity):
```

```
### Mixture of three exponentials

### ANALYTICAL MATH
# Ordinary three mixed exponential mixed poisson density
p:=t4*(t1/(1+t1))^j*1/(1+t1)+t5*(t2/(1+t2))^j*1/(1+t2)+(1-t4-t5)\
  *(t3/(1+t3))^j*1/(1+t3):

# probability j=0
p0:=eval(p,j=0):

# Zero-truncated mixed exponential mixed poisson
p_zt:=p/(1-p0):

# Pre-information matrix and vectors for ordinary mixed expl
p_t1t1:=-simplify(diff(diff(ln(p),t1),t1)):
p_t1t2:=-simplify(diff(diff(ln(p),t1),t2)):
p_t1t3:=-simplify(diff(diff(ln(p),t1),t3)):
p_t1t4:=-simplify(diff(diff(ln(p),t1),t4)):
p_t1t5:=-simplify(diff(diff(ln(p),t1),t5)):
p_t2t2:=-simplify(diff(diff(ln(p),t2),t2)):
p_t2t3:=-simplify(diff(diff(ln(p),t2),t3)):
p_t2t4:=-simplify(diff(diff(ln(p),t2),t4)):
p_t2t5:=-simplify(diff(diff(ln(p),t2),t5)):
p_t3t3:=-simplify(diff(diff(ln(p),t3),t3)):
p_t3t4:=-simplify(diff(diff(ln(p),t3),t4)):
p_t3t5:=-simplify(diff(diff(ln(p),t3),t5)):
p_t4t4:=-simplify(diff(diff(ln(p),t4),t4)):
p_t4t5:=-simplify(diff(diff(ln(p),t4),t5)):
p_t5t5:=-simplify(diff(diff(ln(p),t5),t5)):
v_t1:=simplify(diff(p0,t1)):
v_t2:=simplify(diff(p0,t2)):
v_t3:=simplify(diff(p0,t3)):
v_t4:=simplify(diff(p0,t4)):
v_t5:=simplify(diff(p0,t5)):


### DATA INPUT
entrydata:=ImportMatrix(data_file):
rowdim:=LinearAlgebra[RowDimension](entrydata):
fmaxdata:=entrydata[rowdim,1]:
freqdata:=<<seq(i,i=1..fmaxdata)>|<seq(0,i=1..fmaxdata)>>:
for n from 1 to fmaxdata do:
```

```
  for m from 1 to rowdim do:
    if entrydata[m,1]=n then freqdata[n,2]:=entrydata[m,2]: break:
    fi:
  od:
od:
unassign('n'): unassign('m'):

print(freqdata):


### SET FMIN & FMAX FOR ANALYSIS
fmin:=max(7,f_min);
fmax:=min(fmaxdata,f_max);


### SET UP LOOP ON CUTOFFS OF INTEREST
for t from fmin to fmax do:
  if freqdata[t,2]=0 then next
  fi:

  # SET ERROR CONDITIONS
  tracking[t]:=0:

  # BASIC STATISTICS AT UPPER NONEMPTY CUTOFFS FROM FMIN TO FMAX
  s[t]:=add(freqdata[i,2],i=1..t):
  sum1:=evalf(add(freqdata[i,1]*freqdata[i,2],i=1..t)):

  # initial value for s0
   # first find geometric parameter based on
   # first four frequencies
   # create raw data
   raw_data:=\
      <<[seq(1,i=1..freqdata[1,2]),seq(2,i=1..freqdata[2,2]),\
      seq(3,i=1..freqdata[3,2]),seq(4,i=1..freqdata[4,2])]>>;
   # mle for right and left truncated geometric
   geom_par:=fsolve((-1*add(raw_data[i,1],\
      i=1..LinearAlgebra[RowDimension](raw_data))/(1-par))\
      +LinearAlgebra[RowDimension](raw_data)*\
      add(j*(1-par)^(j-1),j=1..4)/add((1-par)^j,j=1..4),par);
   s0:=(geom_par/(add(geom_par*(1-geom_par)^j,j=0..4))\
*add(raw_data[i,1],i=1..LinearAlgebra[RowDimension](raw_data)))/\
(1-geom_par/(add(geom_par*(1-geom_par)^j,j=0..4)));
```

```
# initialize values for t1, t2, t3
t1_init:=1:t2_init:=1:t3_init:=1:t4_init:=1/3:t5_init:=1/3:


## append s0 to freqdata to make a new full data set
fulldata:=<<seq(x,x=0..t)>|<[s0,seq(0,x=1..t)]>>;
for x from 1 to t do:
  fulldata[x+1,2]:=freqdata[x,2]:  od:
print("full data",fulldata);

### obtain starting values for t1, t2, t3
## subset fulldata for low frequencies
low_freq:=LinearAlgebra[SubMatrix]\
   (fulldata,1..floor(2*(t+1)/4)+1,1..2);
print("low_freq",low_freq);

# fist find sample size
n_low_freq:=add(low_freq[i,2],i=1..floor(2*(t+1)/4)+1);

# next calculate sample mean
low_freq_mean:=evalf(add(low_freq[i,1]*low_freq[i,2],\
   i=1..floor(2*(t+1)/4)+1)/n_low_freq);

# calculate geometric success probability
geom_success_prob:=1/(low_freq_mean+1);

# calculate starting value for t1 using mle of low_feq data
t1_init:=(1-geom_success_prob)/geom_success_prob;


## subset fulldata for middle frequencies
mid_freq:=LinearAlgebra[SubMatrix](fulldata,\
   floor((t+1)/4)+1..floor(3*(t+1)/4)+1,1..2);
print("mid_freq",mid_freq);

# find row dim of mid_freq
mid_freq_dim:=LinearAlgebra[RowDimension](mid_freq);

# fist find sample size
n_mid_freq:=add(mid_freq[i,2],i=1..mid_freq_dim);
```

```
# next calculate sample mean
mid_freq_mean:=evalf(add(mid_freq[i,1]*mid_freq[i,2],\
   i=1..mid_freq_dim)/n_mid_freq);

# calculate geometric success probability
geom_success_prob:=1/(mid_freq_mean+1-(floor((t+1)/4)-1));

# calculate starting value for t2 using mle of low_feq data
t2_init:=(1-geom_success_prob)/geom_success_prob;


## subset freqdata for high frequencies
high_freq:=LinearAlgebra[SubMatrix](fulldata,\
   floor(2*(t+1)/4)+1..t+1,1..2);
print("high_freq",high_freq);

# find row dim of high_freq
high_freq_dim:=LinearAlgebra[RowDimension](high_freq);

# fist find sample size
n_high_freq:=add(high_freq[i,2],i=1..high_freq_dim);

# next calculate sample mean
high_freq_mean:=evalf(add(high_freq[i,1]*high_freq[i,2],\
   i=1..high_freq_dim)/n_high_freq);

# calculate geometric success probability
geom_success_prob:=1/(high_freq_mean+1-(floor(2*(t+1)/4)-1));

# calculate starting value for t2 using mle of high_feq data
t3_init:=(1-geom_success_prob)/geom_success_prob;


## define mles with starting values
mles:={t1=t1_init,t2=t2_init,t3=t3_init,t4=t4_init,t5=t5_init}:
print("init mles",mles);

### initialize number of em iterations
ncem:=0;

### start loop for imputed zero count
```

```
for h from 1 to mle_nonzt_iter do;

## append s0 to freqdata to make a new full data set
fulldata:=<<seq(x,x=0..t)>|<[s0,seq(0,x=1..t)]>>;
for x from 1 to t do:
  fulldata[x+1,2]:=freqdata[x,2]:  od:
#print("full data",fulldata);


##### start a new loop(i) here for EM1 for non zt data
for i from 1 to mle_nonzt_iter do;


  ##
  ## E step
  ##

  # initialize imputed value vectors
  z1_impute:=<<seq(0,k=1..t+1)>>;
  z2_impute:=<<seq(0,k=1..t+1)>>;
  z3_impute:=<<seq(0,k=1..t+1)>>;

  # define geometric density
  geom_den:=(theta/(1+theta))^(j)*(1/(1+theta));

  for w from 0 to t do:

    # calculate E[Z=(1,0,0)|X=w] = P(Z=(1,0,0)|X=w) =
    # P(X=w|Z=(1,0,0))P(Z=(1,0,0))/\
    # [P(X=w|Z=(1,0,0))P(Z=(1,0,0))+\
    #  P(X=w|Z=(0,1,0))P(Z=(0,1,0))+\
    #  P(X=w|Z=(0,0,1))P(Z=(0,0,1))]
    z1_impute[w+1,1]:=(subs(j=w,theta=eval(t1,mles),geom_den)\
      *eval(t4,mles))/\
      (\
      (subs(j=w,theta=eval(t1,mles),geom_den)*eval(t4,mles))+\
      (subs(j=w,theta=eval(t2,mles),geom_den)*eval(t5,mles))+\
      (subs(j=w,theta=eval(t3,mles),geom_den)*(1-eval(t4,mles)\
         -eval(t5,mles)))
      );
  od; unassign('w');
```

```
for w from 0 to t do:

  # calculate E[Z=(0,1,0)|X=w] = P(Z=(0,1,0)|X=w) =
  # P(X=w|Z=(0,1,0))P(Z=(0,1,0))/
  # [P(X=w|Z=(1,0,0))P(Z=(1,0,0))+\
  #  P(X=w|Z=(0,1,0))P(Z=(0,1,0))+\
  #  P(X=w|Z=(0,0,1))P(Z=(0,0,1))]
  z2_impute[w+1,1]:=(subs(j=w,theta=eval(t2,mles),geom_den)\
    *eval(t5,mles))/\
    (\
    (subs(j=w,theta=eval(t1,mles),geom_den)*eval(t4,mles))+\
    (subs(j=w,theta=eval(t2,mles),geom_den)*eval(t5,mles))+\
    (subs(j=w,theta=eval(t3,mles),geom_den)*(1-eval(t4,mles)\
    -eval(t5,mles)))
    );
od; unassign('w');


# use constraint to define z3
for w from 0 to t do:
  z3_impute[w+1,1]:=1-z1_impute[w+1,1]-z2_impute[w+1,1];
od; unassign('w');


# print imputed values for each iteration of EM
# print("z1_impute",z1_impute);
# print("z2_impute",z2_impute);
# print("z3_impute",z3_impute);


##
## M step
##

### calculate t4 and t5 estimates
# initialize vector of products; we will weight each
# imputed value by its count
z1_count_prod:=<<seq(0,i=1..t+1)>>;
for r from 0 to t do;
  z1_count_prod[r+1,1]:=z1_impute[r+1,1]*fulldata[r+1,2];
```

```
od;
unassign('r');
# print("z1 count prod",z1_count_prod);

z2_count_prod:=<<seq(0,i=1..t+1)>>;
for r from 0 to t do;
  z2_count_prod[r+1,1]:=z2_impute[r+1,1]*fulldata[r+1,2];
od;
unassign('r');
# print("z2 count prod",z2_count_prod);

z3_count_prod:=<<seq(0,i=1..t+1)>>;
for r from 0 to t do;
  z3_count_prod[r+1,1]:=z3_impute[r+1,1]*fulldata[r+1,2];
od;
unassign('r');
# print("z3 count prod",z3_count_prod);


# sum the z_count_prod vector and divide by sample size(s)
# then solve for t4, t5
t4_new:=add(z1_count_prod[i,1],i=1..t+1)/(s[t]+s0);
t5_new:=add(z2_count_prod[i,1],i=1..t+1)/(s[t]+s0);


### now estimate t1, t2, t3 given z_impute

# closed form for estimates of t1, t2, t3 derived from
# eq. (2.33) McLachlan
t1_new:=add(z1_count_prod[j,1]*(fulldata[j,1]),j=1..t+1)/\
   add(z1_count_prod[j,1],j=1..t+1);
t2_new:=add(z2_count_prod[j,1]*(fulldata[j,1]),j=1..t+1)/\
   add(z2_count_prod[j,1],j=1..t+1);
t3_new:=add(z3_count_prod[j,1]*(fulldata[j,1]),j=1..t+1)/\
   add(z3_count_prod[j,1],j=1..t+1);


# define mles_try as current iteration mles
mles_try:={t1=t1_new,t2=t2_new,t3=t3_new,t4=t4_new,t5=t5_new};

# print mles_try at each iteration
# print(i,"mle estimates","t1=",eval(t1,mles_try),\
```

```
  #   "t2=",eval(t2,mles_try),"t3=",eval(t3,mles_try),\
  #   "t4=",eval(t4,mles_try),"t5=",eval(t5,mles_try));

  # break for loop if convergence is reached
  if    eval(t1,mles_try)/eval(t1,mles) > 1-10^(-mle_nonzt_tol)
    and eval(t1,mles_try)/eval(t1,mles) < 1+10^(-mle_nonzt_tol)
    and eval(t2,mles_try)/eval(t2,mles) > 1-10^(-mle_nonzt_tol)
    and eval(t2,mles_try)/eval(t2,mles) < 1+10^(-mle_nonzt_tol)
    and eval(t3,mles_try)/eval(t3,mles) > 1-10^(-mle_nonzt_tol)
    and eval(t3,mles_try)/eval(t3,mles) < 1+10^(-mle_nonzt_tol)
    and eval(t4,mles_try)/eval(t4,mles) > 1-10^(-mle_nonzt_tol)
    and eval(t4,mles_try)/eval(t4,mles) < 1+10^(-mle_nonzt_tol)
    and eval(t5,mles_try)/eval(t5,mles) > 1-10^(-mle_nonzt_tol)
    and eval(t5,mles_try)/eval(t5,mles) < 1+10^(-mle_nonzt_tol)
    then break;
  fi;

  # if mles not converged, update parameter estimates
  mles:=mles_try;

  # track error if EM step reaches max # of iterations
  if i=mle_nonzt_iter+1
    then tracking[t]:=1;
    fi;

# end EM1 loop
od;


#### Impute a new value for s0
s0_new:=evalf(eval(p0,mles))*(s[t]+s0);
print(t,h,"s0_new",s0_new);


### S_0 CONVERGENCE TEST
# when new s0 has converged, break out of loop
    if  s0/s0_new > 1-10^(-mle_nonzt_tol)
    and s0/s0_new < 1+10^(-mle_nonzt_tol)
    then break;
  fi;

### keep track of number of consecutive em iterations
```

```
  ncem:=ncem+1;
  print("ncem",ncem,"so_old",s0_old,"s0",s0,"s0_new",s0_new);


  #### acceleration step


  print("max i",i);
  if h>10 and abs(s0-s0_old)>abs(s0_new-s0) and ncem>1
    then s0_step:=s0_new-s0;
    num_step:=(s0_new-s0)/((s0-s0_old)-(s0_new-s0));
    print("s0_step",s0_step,"num_step",num_step);
    s0_new:=s0_new+num_step*s0_step;
    # reset nem
    ncem:=0;
    fi;

  #### end of acceleration step

  ### save s0 as old s0
  s0_old:=s0;

  ## save new s0 if no convergence
  s0:=s0_new;

  # track error if s0 loop reaches max # of iterations
    if h=mle_nonzt_iter+1
      then tracking[t]:=2;
      fi;

## end outer loop
od;

  # save converged mles
  mles_final[t]:=mles;
  print("mles final","t1=",eval(t1,mles_final[t]),"t2=",\
      eval(t2,mles_final[t]),"t3=",eval(t3,mles_final[t]),\
      "t4=",eval(t4,mles_final[t]),"t5=",eval(t5,mles_final[t]));


  ### COMPUTE FITTED VALUES
  # we will export these in the fits file later
```

```
for j from 1 to t do;
  fitzt[j,t]:=evalf(s[t]*eval(p_zt,mles_final[t]));
od;
unassign('j');


### GOODNESS OF FIT BASED ON FINAL MLES

accum_obs:=0: accum_fit:=0:
# k is concatenated cell number
k:=0:

# add fits and observations for each frequency
for j from 1 to t do:
  accum_obs:=accum_obs+freqdata[j,2]:
  accum_fit:=accum_fit+fitzt[j,t]:
  # if fits >=5 then assign to cell[k]
  if accum_fit >= 5 then
    k:=k+1: cell_obs[k]:=accum_obs:
    cell_fit[k]:=accum_fit: j_stop:=j: k_stop:=k:
    accum_obs:=0: accum_fit:=0:

    # print concatenated cells with fits >= 5
    # print("k",k,"cell_fit[k]",cell_fit[k]):
  fi:
od:

# if fits in last cells do not sum to 5,
# then add them to the previous cell
if j_stop < t then
  cell_obs[k]:=cell_obs[k]+add(freqdata[l,2],l=j_stop+1..t):
  cell_fit[k]:=cell_fit[k]+add(fitzt[l,t],l=j_stop+1..t):
fi:
unassign('j'):
unassign('k'):

# print all concatenated cells with observed values
for k from 1 to k_stop do:
  print("k",k,"cell_fit[k]",cell_fit[k],"cell_obs[k]",\
     cell_obs[k]):
od:
```

```
  # print freq where last fits>=5,
  # and number of concatenated cells
  # print("j_stop",j_stop,"k_stop",k_stop):
  unassign('k'):

  # calculate chi-sq statistic and p-value for concatenated cells
  if k_stop>6 then
    chistat_cell:=add((cell_obs[k]-cell_fit[k])^2/\
       cell_fit[k],k=1..k_stop) + s[t]-add(fitzt[i,t],i=1..t);
    GOF_mles_cell[t]:=evalf(1-stats\
       [statevalf,cdf,chisquare[k_stop+1-5-1]](chistat_cell));

    # print GOF chi-sq statistic and p-value
    print(t,"cells chi-sq stat",chistat_cell,\
       "concatenated cells GOF",GOF_mles_cell[t]):
  fi:

  # calculate (naive) chi-sq statistic and p-value for all cells
  chistat_all:=add((freqdata[i,2]-fitzt[i,t])^2/\
     fitzt[i,t],i=1..t)+s[t]-add(fitzt[i,t],i=1..t);
  GOF_mles_all[t]:=evalf(1-\
     stats[statevalf,cdf,chisquare[t+1-5-1]](chistat_all)):

  # print GOF (naive) chi-sq stat and p-value
  print(t,"all cells chi-sq stat",chistat_all,\
     "all cells GOF",GOF_mles_all[t]):


### SET UP ORDINARY NON-ZERO-TRUNCATED INFORMATION MATRIX,
### ASSOCIATED VECTOR, & EMPIRICAL QUANTITIES
   p_mles_final[t]:=eval(p,mles_final[t]):
   p0_mles_final[t]:=evalf(eval(p0,mles_final[t])):
   s0_mles_final[t]:=evalf(s[t]*p0_mles_final[t]/\
      (1-p0_mles_final[t])):
   s_hat[t]:=s[t]+s0_mles_final[t]:
   p_t1t1_mles_final:=eval(p_t1t1,mles_final[t]):
   p_t1t2_mles_final:=eval(p_t1t2,mles_final[t]):
   p_t1t3_mles_final:=eval(p_t1t3,mles_final[t]):
   p_t1t4_mles_final:=eval(p_t1t4,mles_final[t]):
   p_t1t5_mles_final:=eval(p_t1t5,mles_final[t]):
   p_t2t2_mles_final:=eval(p_t2t2,mles_final[t]):
   p_t2t3_mles_final:=eval(p_t2t3,mles_final[t]):
```

```
   p_t2t4_mles_final:=eval(p_t2t4,mles_final[t]):
   p_t2t5_mles_final:=eval(p_t2t5,mles_final[t]):
   p_t3t3_mles_final:=eval(p_t3t3,mles_final[t]):
   p_t3t4_mles_final:=eval(p_t3t4,mles_final[t]):
   p_t3t5_mles_final:=eval(p_t3t5,mles_final[t]):
   p_t4t4_mles_final:=eval(p_t4t4,mles_final[t]):
   p_t4t5_mles_final:=eval(p_t4t5,mles_final[t]):
   p_t5t5_mles_final:=eval(p_t5t5,mles_final[t]):
   v_t1_mles_final:=eval(v_t1,mles_final[t]):
   v_t2_mles_final:=eval(v_t2,mles_final[t]):
   v_t3_mles_final:=eval(v_t3,mles_final[t]):
   v_t4_mles_final:=eval(v_t4,mles_final[t]):
   v_t5_mles_final:=eval(v_t5,mles_final[t]):

### EVALUATE STANDARD ERROR
   inf_t1t1_mles_final:=0:
   inf_t1t2_mles_final:=0:
   inf_t1t3_mles_final:=0:
   inf_t1t4_mles_final:=0:
   inf_t1t5_mles_final:=0:
   inf_t2t2_mles_final:=0:
   inf_t2t3_mles_final:=0:
   inf_t2t4_mles_final:=0:
   inf_t2t5_mles_final:=0:
   inf_t3t3_mles_final:=0:
   inf_t3t4_mles_final:=0:
   inf_t3t5_mles_final:=0:
   inf_t4t4_mles_final:=0:
   inf_t4t5_mles_final:=0:
   inf_t5t5_mles_final:=0:
   std_err_try:=0:
   std_err[t]:=1:
   for j from 0 to 1000 do;
      inf_t1t1_mles_final:=inf_t1t1_mles_final+\
         evalf(p_mles_final[t]*p_t1t1_mles_final):
      inf_t1t2_mles_final:=inf_t1t2_mles_final+\
         evalf(p_mles_final[t]*p_t1t2_mles_final):
      inf_t1t3_mles_final:=inf_t1t3_mles_final+\
         evalf(p_mles_final[t]*p_t1t3_mles_final):
      inf_t1t4_mles_final:=inf_t1t4_mles_final+\
         evalf(p_mles_final[t]*p_t1t4_mles_final):
      inf_t1t5_mles_final:=inf_t1t5_mles_final+\
```

```
   evalf(p_mles_final[t]*p_t1t5_mles_final):
inf_t2t2_mles_final:=inf_t2t2_mles_final+\
   evalf(p_mles_final[t]*p_t2t2_mles_final):
inf_t2t3_mles_final:=inf_t2t3_mles_final+\
   evalf(p_mles_final[t]*p_t2t3_mles_final):
inf_t2t4_mles_final:=inf_t2t4_mles_final+\
   evalf(p_mles_final[t]*p_t2t4_mles_final):
inf_t2t5_mles_final:=inf_t2t5_mles_final+\
   evalf(p_mles_final[t]*p_t2t5_mles_final):
inf_t3t3_mles_final:=inf_t3t3_mles_final+\
   evalf(p_mles_final[t]*p_t3t3_mles_final):
inf_t3t4_mles_final:=inf_t3t4_mles_final+\
   evalf(p_mles_final[t]*p_t3t4_mles_final):
inf_t3t5_mles_final:=inf_t3t5_mles_final+\
   evalf(p_mles_final[t]*p_t3t5_mles_final):
inf_t4t4_mles_final:=inf_t4t4_mles_final+\
   evalf(p_mles_final[t]*p_t4t4_mles_final):
inf_t4t5_mles_final:=inf_t4t5_mles_final+\
   evalf(p_mles_final[t]*p_t4t5_mles_final):
inf_t5t5_mles_final:=inf_t5t5_mles_final+\
   evalf(p_mles_final[t]*p_t5t5_mles_final):
infomat_mles_final:=evalf(Matrix(5,5,[\
   [inf_t1t1_mles_final,inf_t1t2_mles_final,\
      inf_t1t3_mles_final,inf_t1t4_mles_final,\
         inf_t1t5_mles_final],\
   [inf_t1t2_mles_final,inf_t2t2_mles_final,\
      inf_t2t3_mles_final,inf_t2t4_mles_final,\
         inf_t2t5_mles_final],\
   [inf_t1t3_mles_final,inf_t2t3_mles_final,\
      inf_t3t3_mles_final,inf_t3t4_mles_final,\
         inf_t3t5_mles_final],\
   [inf_t1t4_mles_final,inf_t2t4_mles_final,\
      inf_t3t4_mles_final,inf_t4t4_mles_final,\
         inf_t4t5_mles_final],\
   [inf_t1t5_mles_final,inf_t2t5_mles_final,\
      inf_t3t5_mles_final,inf_t4t5_mles_final,\
         inf_t5t5_mles_final]])):
v_mles_final:=evalf(Matrix(5,1,[[v_t1_mles_final],\
   [v_t2_mles_final],[v_t3_mles_final],[v_t4_mles_final],\
      [v_t5_mles_final]])):
if LinearAlgebra[Rank](infomat_mles_final) = 5
   then  std_err_try:=evalf(sqrt(s_hat[t]*p0_mles_final[t])\
```

```
            *(1-p0_mles_final[t]-(1/p0_mles_final[t])*\
          LinearAlgebra[Multiply](LinearAlgebra[Multiply]\
            (LinearAlgebra[Transpose](v_mles_final),\
          LinearAlgebra[MatrixInverse](infomat_mles_final)),\
            v_mles_final)[1,1])^(-1/2))
          else next fi:
       if std_err_try>0
          and std_err_try/std_err[t] > 1-10^(-std_err_tol)
          and std_err_try/std_err[t] < 1+10^(-std_err_tol) then break
          else std_err[t]:=std_err_try: next: fi:
    od:

 # print(t,j):
  unassign('j'):


# end t loop
od;


### OUTPUT FITTED VALUES

# first, find max non-zero frequency in interval (fmin,fmax)
nonzero:=0:
for u from fmin to fmax do:
  if freqdata[u,2]=0 then next
  fi:
  # nonzero is the max non-zero frequency in interval (fmin,fmax)
  nonzero:=nonzero+1:
od:
unassign('u'):

fits:=Matrix(fmax,nonzero+2):

for w from 1 to fmax do:
  # first column lists frequencies
  fits[w,1]:=w:
  # second column lists counts
  fits[w,2]:=freqdata[w,2]:
od:

# rest of columns fill in fitted values for non-zero t values
```

```
u:=0:
for z from fmin to fmax do:
  if freqdata[z,2]=0 then next
  fi:
  u:=u+1:
  for v from 1 to z do:
    fits[v,u+2]:=fitzt[v,z]:
  od:
od:
unassign('w'):
unassign('u'):
unassign('v'):
unassign('z'):

# print fitted values
print("fits",fits):
ExportMatrix(output_fits_file,fits):


### OUTPUT ANALYSIS FILE
results:=Matrix(nonzero,11):
u:=0:
for t from fmin to fmax do:
  if freqdata[t,2]=0 then next
  fi:
  u:=u+1:
  # right truncation point
  results[u,1]:=t:

  # final mle estimates
  results[u,2]:=mles_final[t]:

  # non-coverage
  results[u,3]:=p0_mles_final[t]:

  # estimate of unobserved species
  results[u,4]:=s0_mles_final[t]:

  # estimate of total species based on subset
  results[u,5]:=s_hat[t]:

  # estimate of total species based on full data
```

```
  results[u,6]:=add(freqdata[i,2],i=1..fmaxdata)+s0_mles_final[t]:

  # standard error
  results[u,7]:=std_err[t]:

  # lower bound for standard error
  results[u,8]:=sqrt(s_hat[t]*p0_mles_final[t]/\
     (1-p0_mles_final[t])):

  # naive GOF
  results[u,9]:=GOF_mles_all[t]:

  # concatenated GOF
  results[u,10]:=GOF_mles_cell[t]:

  # error tracking
  results[u,11]:=tracking[t]:
od:

# print analysis file
print(results):

# export analysis file
ExportMatrix(output_analysis_file, results):
```

REFERENCES

[1] Berger, J. O., L. Brunero, and L. Wolpert (1999) Integrated Likelihood Methods for Eliminating Nuisance Parameters. *Statistical Science*, **14**(1): 1-28.

[2] Böhning, Dankmar and Ronny Kuhnert (2005) Equivalence of Truncated Count Mixture Distributions and Mixtures of Truncated Count Distributions. *Unpublished Manuscript*, (accessed 23 March 2006) http://www.personal.rdg.ac.uk/∼sns05dab/equivalencebio.pdf.

[3] Böhning, Dankmar and Dieter Schön (2005) Nonparametric Maximum Likelihood Estimation of Population Size Based on the Counting Distribution. *Journal of the Royal Statistical Society, Series C, Applied Statistics*, **54**: 721-737.

[4] Booth, Andrew D. (1967) A 'Law' of Occurrences for Words of Low Frequency. *Information and Control*, **10**: 386-393.

[5] Borgella, R. and T. A. Gavin (2005) Avian Community Dynamics in a Fragmented Tropical Landscape. *Ecological Application*, **15**(3): 1062-1073.

[6] Brookmeyer, R. and M. H. Gail (1988) A Method For Obtaining Short-Term Projections and Lower Bounds on the Size of the Aids Epidemic. *Journal Of The American Statistical Association*, **83**(402): 301-308.

[7] Bunge, John and M. Fitzpatrick (1993) Estimating the Number of Species: A Review. *Journal of the American Statistical Association*, **88**(421): 364-373.

[8] Burrell, Q. L. (1988) A Simple Empirical Method for Predicting Library Circulations. *Journal of Documentation*, **44**(4): 302-314.

[9] Cannon, C. H., D. R. Peart, and M. Leighton (1998) Tree Species Diversity in Commercially Logged Bornean Rainforest. *Science*, **281**(5381): 1366-1368.

[10] Chao, Anne and Shen-Ming Lee (1992) Estimating the Number of Classes Via Sample Coverage. *Journal of the American Statistical Associaiton*, **87**: 210-217.

[11] Cobabe, E. A. and W. D. Allmon (1994) Effects of Sampling on Paleoecological and Taphonomic Analyses in High-Diversity Fossil Accumulations-an Example from the Eocene Gosport Sand, Alabama. *Lethaia*, **27**(2): 167-178.

[12] Ding, Ye (1996) On the Asymptotic Normality of Multinomial Population Size Estimators with Application to Backcalculation of AIDS Epidemic. *Biometrika*, **83**(3): 695-699.

[13] Dunbar, J., S. M. Barns, L. O. Ticknor, and C. R. Kuske (2002) Empirical and Theoretical Bacterial Diversity in Four Arizona Soils. *Applied and Environmental Microbiology*, **68**(6): 3035-3045.

[14] Efron, B. and R. Thisted (1976) Estimating the Number of Unseen Species: How Many Words Did Shakespeare know? *Biometrika*, **63**: 435-447.

[15] Esty, W. W. (1986) Estimation of the Size of a Coinage: A Survey and Comparison of Methods. *Numismatic Chronicle*, **146**: 185-215.

[16] Feldmann, Anja and Ward Whitt (1998) Fitting Mixtures of Exponentials to Long-tail Distributions to Analyze Network Performance Models. *Performance Evaluation*, **31**(3-4): 245-279.

[17] Fisher, Ronald Aylmer, A. Steven Corbet, and C. B. Williams (1943) The Relation Between the Number of Species and the Number of Individuals in a Random Sample of an Animal Population. *Journal of Animal Ecology*, **12**: 42-58.

[18] Folgarait, P. J., O. Bruzzone, S. D. Porter, M. A. Pesquero, and L. E. Gilbert (2005) Biogeography and Macroecology of Phorid Flies that Attack Fire Ants in South-Eastern Brazil and Argentina. *Journal of Biogeography*, **32**(2): 353-367.

[19] Hong, Sun-Hee, John Bunge, Sun-Ok Jeon, and Slava S. Epstein (2006) Predicting Microbial Species Richness. *Proceedings of the National Academy of Sciences*, **103**(1): 117-122.

[20] Hser, Yih-Ing (1993) Population Estimation of Illicit Drug Users in Los Angeles County. *The Journal of Drug Issues*, **23**(2): 323-334.

[21] Lewins, W. A. and D. N. Joanes (1984) Bayesian Estimation of the Number of Species. *Biometrics*, **40**: 323-328.

[22] Lloyd, C. J., P. S. F. Yip, and K. S. Chan (1999) Estimating the Number of Faults: Efficiency of Removal, Recapture, and Seeding. *IEEE Transactions on Reliability*, **48**(4): 369-376.

[23] McLachlan, Geoffrey and David Peel (2000) *Finite Mixture Models*. New York: John Wiley and Sons, Inc..

[24] Olken, Frank and Doron Rotem (1995) Random Sampling from Databases: A Survey. *Statistics and Computing*, **5**(1): 25-42.

[25] Rodrigues, Josemar, Luis A. Milan, and José G. Leite (2001) Hierarchical Bayesian Estimation for the Number of Species. *Biometrical Journal*, **43**(6): 737-746.

[26] Sampaio, E. M., E. K. V. Kalko, E. Bernard, B. Rodriguez-Herrera and C. O. Handley (2003) A Biodiversity Assessment of Bats (Chiroptera) in a Tropical Lowland Rainforest of Central Amazonia Including Methodological and Conservation Considerations. *Studies on Neotropical Fauna and Environment*, **38**(1): 17-31.

[27] Sanathanan, Lalitha (1972) Estimating the Size of a Multinomial Population. *The Annals of Mathematical Statistics*, **43**(1): 142-152.

[28] Scharff, N., J. A. Coddington, C. E. Griswold, G. Hormiga, and P. D. Bjorn (2003) When to Quit? Estimating Spider Species Richness in a Northern European Deciduous Forest. *Journal of Arachnology*, **31**(2): 246-273.

[29] Smith, K. L. and M. L. Jones (2005) Watershed-Level Sampling Effort Requirements for Determining Riverine Fish Species Composition. *Canadian Journal of Fisheries and Aquatic Sciences*, **62**(7): 1580-1588.

[30] Solow, Andrew R. (1994) On the Bayesian Estimation of the Number of Species in a Community. *Ecology*, **75**(7): 2139-2142.

[31] Stam, A. J. (1987) Statistical Problem in Ancient Numismatics. *Statistica Neerlandica*, **41**(3): 151-173.

[32] Tollenaar, Nikolaj and Ab Mooijaart (2003) Type I Errors and Power of the Parametric Bootstrap Goodness-of-Fit Test: Full and Limited Information. *British Journal of Mathematical and Statistical Psychology*, **56**: 271-288.

[33] Wang, Ji-Ping Z. and Bruce G. Lindsay (2005) A Penalized Nonparametric Maximum Likelihood Approach to Species Richness Estimation. *Journal of the American Statistical Association*, **100**(471): 942-959.