INPUT MODELING FOR DISCRETE DISTRIBUTIONS

by

Hale Dündar

B.S., Industrial Engineering, Dokuz Eylül University, 2009

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Master of Science

Graduate Program in Systems and Control Engineering

Boğaziçi University

2015

ACKNOWLEDGEMENTS

First I would like to express my sincere gratitude to my thesis advisor, Assoc. Prof. Wolfgang Hörmann for his persistent support and patience for teaching and advising throughout my thesis.

I am thankful to my committee members Prof. Yağmur Denizhan and Assist. Prof. Gönenç Yücel for their concern and contribution with valuable comments.

A very special thank to my friend; Başak Pınar for her encourage to be involved in this project, as well as her support.

I am greatly indebted to my family for their encourage and believe throughout my life and for their concern throughout my dissertation. This thesis is lovingly dedicated to my parents.

ABSTRACT

INPUT MODELING FOR DISCRETE DISTRIBUTIONS

In this thesis we developed a new input modeling tool called `FitAllDiscrete(x)` for the statistical software R. We then tested the success of our modeling tool by comparing it with two popular commercial softwares: Arena input analyzer and EasyFit. Considering limitations of the software packages we mention, we developed a more powerful tool which is based on well-accepted and efficient statistical methods. Input modeling tools model the data to find the most proper distribution. Our input modeling function considers input data from discrete distributions. It tries all distributions automatically and recommends the most proper one for the data. Input modeling is executed by the three essential steps including distribution selection, parameter estimation and goodness of fit testing. In this study, for distribution selection we used Akaike's information criterion (AIC) which is a popular selection criterion. For parameter estimation we used maximum likelihood estimation (MLE). Akaike's information criterion does not guarantee that the selected distribution has a good fit for the data. Therefore, we applied the chi-square test to assess whether the fit is statistically meaningful. According to it, if the fit is good the distribution is accepted and recommended to the user. Otherwise, the empirical distribution is recommended. To decide how reliable our tool is, we first performed an automatic test. Then, we performed a manual test by comparing our input modeling tool with Arena input analyzer and EasyFit. During the tests, we used random discrete data sets generated from the distributions of our pool. The results of the automatic test showed that our input modeling tool is successful in finding the correct distributions for simulated data sets. The comparison test showed that the recommendations by `FitAllDiscrete(x)` are more successful than those of the commercial products. Hence we succeeded in developing a new high-quality input analyzer.

ÖZET

KESİKLİ DAĞILIMLAR İÇİN GİRDİ MODELLEME

Bu tezde `FitAllDiscrete(x)` adında yeni bir girdi modelleme aracını R istatistiksel yazılımı için geliştirdik. Daha sonra, modelleme aracımızın başarısını iki iyi bilinen ticari yazılım Arena girdi modelleyici ve EasyFit ile karşılaştırarak test ettik. Bahsi geçen yazılım paketlerinin kısıtlarını dikkate alarak, kabul görmüş ve etkin istatistiksel yöntemlere dayanan daha başarılı bir araç geliştirdik. Girdi modelleme araçları en uygun dağılımı bulmak için veriyi modeller. Bizim modelleme fonksiyonumuz `FitAllDiscrete(x)` kesikli dağılımlardan gelen verileri dikkate alır. Fonksiyon, tüm dağılımları otomatik olarak dener ve veri için en uygun olanı önerir. Girdi modelleme dağılım seçimi, parametre tahmini ve iyi uyum testinden oluşan üç temel aşama ile gerçekleştirilir. Bu çalışmada, dağılım seçimi için popüler bir seçim kriteri olan Akaike'nin bilgi kriterini kullandık. Parametre tahmini için, en yüksek olabilirlik yöntemini kullandık. Akaike'nin bilgi kriteri seçilen dağılımın veri için iyi bir uyuma sahip olduğunu garantilemez. Bu nedenle, uyumun istatistiksel olarak anlamlı olup olmadığını belirlemek için ki-kare testini uyguladık. Buna göre, eğer uyum iyi ise dağılım kabul edilir ve kullanıcıya önerilir. Aksi halde, deneysel dağılım önerilir. Modelleme aracımızın ne kadar güvenilir olduğuna karar vermek için öncelikle bir otomatik test gerçekleştirdik. Daha sonra girdi modelleme aracımızı Arena girdi modelleyici ve EasyFit ile karşılaştırarak manuel bir test gerçekleştirdik. Testler esnasında, havuzumuzdaki dağılımlardan üretilen rasgele kesikli veri setlerini kullandık. Otomatik test sonuçları girdi modelleme aracımızın simüle edilen veri setleri için doğru dağılımı bulmada başarılı olduğunu göstermiştir. Karşılaştırma testi, `FitAllDiscrete(x)`'in önerilerinin ticari yazılımlarınkinden daha başarılı olduğunu göstermiştir. Bunun sonucu olarak, yeni bir yüksek kaliteli girdi analizcisi geliştirme konusunda başarılı olduk.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS

$x_{(1)}$	Smallest order statistic
$x_{(n)}$	Largest order statistic
Λ	Log-likelihood function
χ^2	Chi-square test statistic
Γ	Gamma function
I	Indicator function
L	Likelihood function

LIST OF ACRONYMS/ABBREVIATIONS

AD	Anderson Darling
AIC	Akaike's Information Criterion
BIC	Bayesian Information Criterion
CDF	Cumulative Density Function
CLT	Central Limit Theorem
iid	Independent and Identically Distributed
KS	Kolmogorov-Smirnov
LSE	Least Square Error
MLE	Maximum Likelihood Estimate
MOM	Method of Moments
MOLM	Method of L-Moments
PDF	Probability Density Function
PMF	Probability Mass Function
SE	Square Error

1. INTRODUCTION

Almost all real-world systems include the influence of random or unpredictable variables in the environment or in its own components. Simulation models represent real systems and those which have random or stochastic components can imitate the probabilistic nature of the system. In simulation models, typical input models are the distributions to represent random inputs (e.g. number of people arriving to an underground station every hour or the demand per unit time). Input modeling is used to select the most proper probability distribution to represent the input data. Therefore, input modeling is a crucial task in stochastic simulation applications since improper models may lead to misleading simulation results. In our study, we consider input data from discrete distributions. For modeling data from continuous distributions, see [1].

There are many distribution fitting softwares that automatically or manually fit a large number of distributions to data and select the most proper distribution. In the fitting softwares, input modeling is executed by the three essential steps including parameter estimation, distribution selection and goodness of fit testing. Two of the popular input modeling softwares are Arena input analyzer and EasyFit. They are quite practical programs since they can be used without knowledge of statistics. However, it is important to note that they have some limitations that affect the tool performance on selecting the true distribution. This issue is an important motivation for us to develop a better modeling tool by considering the problems explained below.

The limitations of the two softwares arise from the methods they prefer. For distribution selection, Arena input analyzer uses square error (SE), EasyFit uses Kolmogorov Smirnov (KS) and Anderson Darling (AD) test statistics. The first disadvantage is that the result of the square error depends on the number of intervals of the histogram which is designated by the user. Another disadvantage is that none of these methods considers the effect of the number of parameters in distribution selection. Distributions with many parameters are more flexible and softwares have tendency

to choose them as the best fit. For parameter estimation, Arena input Analyzer and EasyFit use different methods based on the distribution including maximum likelihood estimation (MLE), method of moments (MOM), method of L-moments (MOLM), least square error (LSE). In this way, they focus on fast computation rather than the accuracy of estimates. It is also important to note that for the goodness of fit test EasyFit does not include the chi-square test although it is a well-known and successful method for modeling discrete data sets.

Our main objective in this thesis is to develop a more powerful input modeling tool which is based on well-accepted and efficient statistical methods. It should be able to automatically fit many discrete distributions to the input data and recommend the most proper distribution. We aim to develop that new tool called `FitAllDiscrete(x)` in the software R which is fast, easy to use and popular especially at universities. It is not required that the user has statistical knowledge as the tool itself can identify the correct distribution. To reach this aim we decided to use the following three well-accepted statistical procedures: maximum likelihood estimation (MLE) for parameter estimation, Akaike's information criterion (AIC) for distribution selection and the chi-square test for goodness of fit testing. It is natural to use Akaike's information criterion (AIC) and maximum likelihood estimation (MLE) together as both methods require the log-likelihood function of the candidate distribution.

After developing the new code, we plan to make an automatic performance test to evaluate how reliable our modeling function `FitAllDiscrete(x)` is in identifying the correct distribution. Then, we will carry out a manual comparison test to find out whether the new tool is more successful than the two input modeling softwares Arena input analyzer and EasyFit. During the tests, we will use discrete data sets generated from different distributions of our pool with many different parameter sets. In this way it is possible to compare the recommended and the true distributions easily.

The contents of this thesis is as follow. Chapter 2 lays out the main topic input modeling by explaining it with the main steps; it also introduces the mentioned input

analyzing softwares. Chapter 3 is concerned with the methodologies used for this study. Chapter 4 gives information on R. Chapter 5 presents the implementation of maximum likelihood estimation and Akaike's information criterion methods for discrete distributions including newly defined rounded continuous distributions. In Chapter 6, goodness of fit tests are discussed and the chi-square test is explained in detail. Chapter 7 clarifies the use of the empirical distribution which is suggested in the absence of a proper parametric distribution. Chapter 8 is designed as a user guide for our input modeling function `FitAllDiscrete(x)`. It gives information about its usage and how to interpret its results. Chapter 9 includes the results of performance tests of our input analyzer function. We take into account the performance of Arena input analyzer and EasyFit in Chapter 10. Finally, the conclusions shortly summarize the main results.

2. GENERAL PRINCIPLES AND SOME COMMENTS ON INPUT MODELING

In this chapter, we first introduce the subject input modeling. Secondly, we mention about two commercial input modeling softwares: Arena input analyzer and EasyFit. Finally, we explain the main steps of input modeling.

2.1. Input Modeling

Stochastic systems contain one or more sources of randomness and very few real world systems are free from the influence of random variables [2]. Therefore, simulation models that imitate real systems also should capture the randomness. To drive the simulation models, we need to model the data related to the random inputs. Input models to simulation applications are based upon probability distributions since each distribution is developed to represent certain physical processes [3]. Hence, input modeling, indeed, corresponds to selecting a probability distribution to data which represents the randomness and uncertainty. To get reliable outputs, we need good models for the inputs. As a result, input data modeling is a critical component of a successful simulation application [4]. It is carried out as the second phase of simulation modeling after the definition phase [5].

There are four steps specified in the development of a useful model of input data [6]:

(i) Collect data from the real system of interest. When data are not available, expert opinion and knowledge of the process must be used to make educated guesses.
(ii) Identify a probability distribution to represent the input process.
(iii) Estimate parameters from data.
(iv) Evaluate the chosen distribution and the associated parameters for goodness of fit. If the procedure fails to yield a fit between an assumed distributional form

and collected data, the empirical form of the distribution may be used.

In this thesis, we design an input modeling tool by following the steps specified above. For details of the main steps, see Section 2.3.1. Input modeling is easily performed when the system inputs can be represented as a sequence of iid random variables [7]. Therefore, our tool considers independent and identically distributed (iid) data that come from discrete distributions. The number of phone calls a company receives and the number of people who vote for a particular candidate are examples of random discrete inputs. For the dependent case, multivariate models or time-series methods are required [4].

It is important to recognize that there is no "true" input model for any stochastic input. The goal is to obtain an approximation that captures the key characteristics of the underlying input process [3]. Briefly, input modeling consists of selecting and fitting a probability distribution to represent the processes whose behaviour can not be predicted with certainty [8].

2.2. Input Modeling Softwares

There are several software packages developed to fit data to the most proper distribution among a list of various distributions. They are either part of a simulation software as the Arena input analyzer, or a software only implemented for input analyzing such as EasyFit, BestFit and ExpertFit. The data are imported as a text file or the user is requested to paste them to a column of a sheet that is formed to insert data. Then, the histogram of the data is constructed. The user is expected to choose the Fit menu item and the data set is automatically tried for all distributions according to the implemented goodness of fit methods. After that, the fit results are represented and the best fitting distribution is chosen based on the distribution selection criterion which the software uses. In this section, we briefly explain two widely used input modeling softwares: Arena input analyzer and EasyFit [9, 10].

2.2.1. Arena Input Analyzer

Arena input analyzer is an input analyzing tool that is part of the Arena simulation software. We have used version 13.5 (2010) throughout this study (See [11]). In this version, there are 12 distributions and among them only the Poisson distribution is a discrete distribution. It imports the data as a text file and first forms the histogram of the data set. There is an option that allows setting the number of subintervals to a value between five and 40 for the histogram. Then it finds the best fitting distribution using the FitAll choice. For goodness of fit testing, it performs the chi-square test for discrete data sets and the p-value is calculated. p-values above 0.05 indicate that the data set can be modeled by that distribution. In order to estimate parameters, it applies different methods depending on the distribution. Methods including method of moments (MOM), maximum likelihood estimation (MLE) and least square error (LSE) are implemented for parameter estimation. It uses square error (SE) as the distribution selection criterion [12]. It ranks all the distributions beginning from the smallest square error value shown by e^2 .This value is measured by the sum of squared differences between the hypothesized frequencies $(\hat{p_j})$ and the actual frequencies (p_j) of the intervals. Therefore, the smaller the value of e^2, the better the fit. SE formula is given by the Equation 2.1 below.

$$e^2 = \sum_{j=1}^{J} [\hat{p}_j - p_j]^2. \tag{2.1}$$

The disadvantage of the SE measure is that it is not sensitive to the model complexity as it does not consider the number of parameters of the distribution. The screen of Arena input analyzer is shown in Figure 2.1.

2.2.2. EasyFit

EasyFit is a data analyzing software that supports 56 distributions, eight of them are discrete. We used version 5.5 which was released in 2010 (See [13]). It is easy to import the data by pasting them into a work sheet. After selecting the FitDistri-

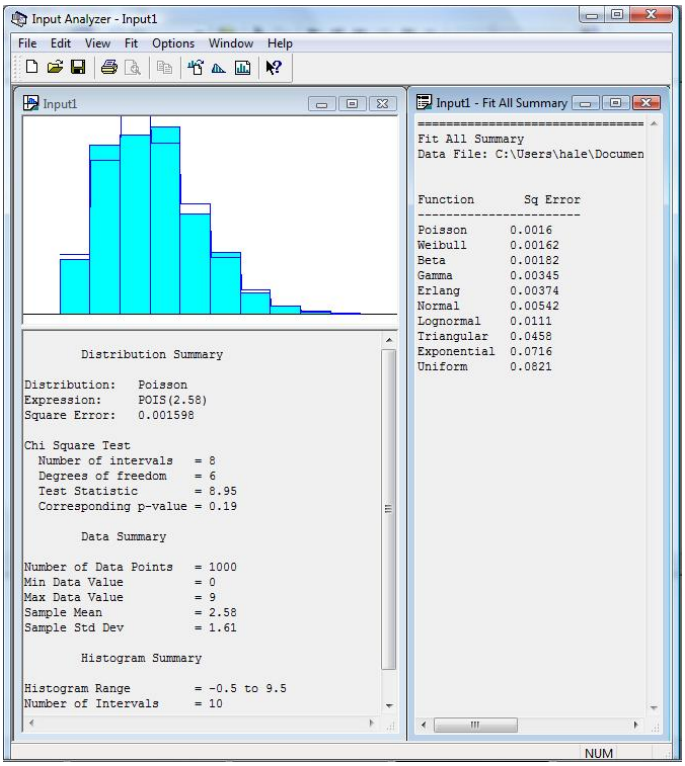
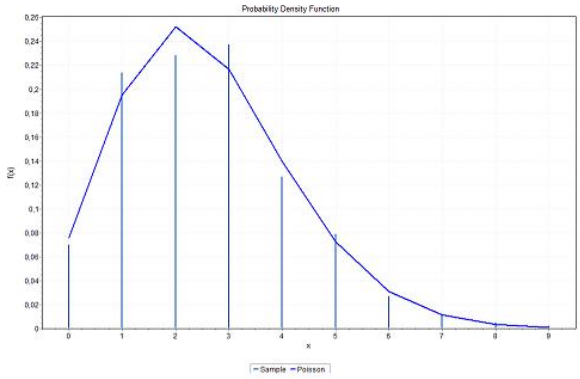


Figure 2.1. Arena Input Analyzer.

butions option, the probability mass function (PMF) of the hypothesized and actual distributions are calculated. As parameter estimation methods, the MOM, MLE or method of L-moments (MOLM) is implemented. The MOM is used for the distributions that allow an analytical solution otherwise MLE or MOLM are implemented [14]. For goodness of fit testing, it implies KS and AD tests for discrete data sets and ranks distributions with respect to these test statistics. Therefore the distribution selection criteria are considered as KS and AD test statistics. As two different goodness of fit methods are available, the user can decide on the goodness of fit test that should be used for distribution selection. The details can be seen on the screen of EasyFit shown in Figure 2.2.

2.3. General Consideration on Input Modeling

In this section, we explain the main steps of input data modeling and tell how the most proper distribution is specified for a given data set using these steps.



Fitting Results

#	Distribution	Parameters
1	D. Uniform	a=0 b=5
2	Geometric	p=0,27925
3	Neg. Binomial	n=405 p=0,99367
4	Poisson	λ=2,581
5	Bernoulli	No fit (data max > 1)
6	Binomial	No fit
7	Hypergeometric	No fit
8	Logarithmic	No fit (data min < 1)

Goodness of Fit - Summary

#	Distribution	Kolmogorov Smirnov		Anderson Darling	
		Statistic	Rank	Statistic	Rank
1	D. Uniform	0,26333	3	343,2	4
2	Geometric	0,41052	4	198,29	3
3	Neg. Binomial	0,24004	2	65,586	2
4	Poisson	0,23921	1	65,214	1
5	Bernoulli	No fit (data max > 1)			
6	Binomial	No fit			
7	Hypergeometric	No fit			
8	Logarithmic	No fit (data min < 1)			

Figure 2.2. EasyFit Software.

2.3.1. Main Steps of Input Modeling

Input modeling consists of three main steps including the parameter estimation, selection of the best fitting distribution and testing the statistical goodness of that fit. Parameter estimation is the phase of determining estimates for the parameters of the distribution. We begin with parameter estimation since it is the first procedure that has to be done. The second step consists of trying out all distributions and choosing the one that matches the data best. In the last step, it is questioned whether the best fitting distribution really corresponds to a good fit with respect to a statistical evaluation. It is obvious that choosing proper and successful methods for each step of input modeling is an important issue.

2.3.1.1. Selecting the Parameters. Selecting the parameter values is a crucial part of input modeling since distributions are specified with parameters and it is a prerequisite for the following steps. It corresponds to finding the estimated values of the

parameters for each probability distribution function. In general, there are two groups of parameter estimation methods consisting of MLE and LSE methods. LSE includes the methods like linear regression which are practical as they do not require assumptions on distributions [15]. However, this becomes a disadvantage as we require doing assumptions by hypothesis testing in the goodness of fit testing step. Therefore we prefer MLE. Due to its good theoretical properties it is the most widely used method of estimation in statistics [16]. Contrary to the input modeling softwares EasyFit and Arena input analyzer, we implement the MLE method for all probability distributions. The other softwares let the user choose more than one estimation methods considering the advantageous sides of each one. However, we focus on the accuracy of estimates rather than having fast computations.

2.3.1.2. Finding the Best Fitting Distribution. Finding the best fitting distribution of our distribution pool corresponds to trying out all distributions with respect to a selection criterion. In literature, there are many selection methods, among them Akaike's information criterion (AIC), Bayesian information criterion (BIC), the root mean squared deviation and cross validation criterion [17]. For each of them, the model which minimizes the chosen criterion is selected as the best fit. AIC and BIC are referred to as the two most commonly used penalized model selection criteria [18]. We prefer AIC which is a popular method for comparing the adequacy of models [19]. For its properties, see Section 3.2.1. The most significant advantage is that it is carried out easily if the MLE is available and gives numerical results which can be interpreted from a statistical point of view. Another advantage is that AIC can also compare distributions with different number of parameters (See Section 3.2). Therefore, we prefer to use this method for selecting the best fitting distribution of our list. Basically, this method calculates a quality value called AIC value for each candidate distribution and compares them. Using those AIC values, it finds the best fitting distribution of the list. It is determined by choosing the one with the smallest AIC value. The AIC formula consists of two parts: the number of parameters and the log-likelihood value. Therefore, the log-likelihood function is required for both parameter estimation and AIC calculation. In order to avoid repetitions, we calculate the AIC value together

with the MLE. On the other hand, AIC -like all selection criteria- does not guarantee that the selected distribution really provide a good fit for the data. We need another assessment of reasonability in order to specify whether the goodness is also statistically meaningful. This assessment is done by goodness of fit testing which is the next stage of input modeling.

Before explaining goodness of fit testing, we would like to discuss the model selection criteria for Arena input analyzer and EasyFit. For model selection, Arena Input Analyzer uses the square error (SE) which calculates the sum of squared differences of frequencies between the empirical histogram and the fitted distributions. Using this method can be disadvantageous as the calculation depends on the number of intervals of the histogram. When the data are arranged with a different number of intervals, different values are calculated each time for the SE value. Therefore, Arena input analyzer is not generally successful in finding the best fitting distribution. EasyFit uses as selection criterion, the test statistics of the Kolmogorov Smirnov (KS) and the Anderson Darling (AD) goodness of fit tests that are both related to the distance between the data and the fitted distribution. Therefore, the distribution with the smallest test statistic value is chosen as the best fit.

2.3.1.3. Verifying the Goodness of Best Fitting Distribution. As we mentioned above, goodness of fit testing is the next main step that is necessary to ensure that the hypothesized model corresponds to a sufficiently good fit for the data. Here, a statistical test is used. “Acception” or “Rejection” is used to decide between the use of a parametric distribution or an empirical distribution. If the test statistic of the best fitting distribution exceeds the level of significance, then it is confirmed that the best fitting distribution also provides an acceptable fit. Otherwise, the empirical distribution is recommended. There are many possible methods for goodness of fit testing based on probability density function (PDF) and cumulative density function (CDF) in general. KS, AD and chi-square tests are the most popular tests [20]. For the goodness of fit testing method, Arena input analyzer uses chi-square however EasyFit implements AD and KS tests for discrete data sets. Each method calculates different test statistics for

the same sample. Therefore, different suggestions are possible when we run the test with different softwares. Hence, we emphasize the most appropriate method for reliable suggestions. We implement the chi-square test because of its success for discrete data sets (See Section 6.2).

2.3.2. How do We Carry Out Input Modeling?

By implementing the steps in Section 2.3.1, our input analyzer `FitAllDiscrete(x)` basically finds the best fitting distribution for the given data set. Firstly, it initiates the MLE for all distributions and finds the parameter estimates, then it calculates the AIC value. After that, it ranks all the distributions with respect to their AIC values in ascending order. The distribution with the smallest AIC is chosen as the best fitting one. The best fit does not guarantee a good fit. In order to verify the quality of the fit, the chi-square test is used and its p-value is calculated for all distributions. If the p-value is larger than the significance level, the fit can be considered to be acceptable. It means that the data set is presented well by this distribution. On the other hand, if the p-value does not exceed the significance level, then it is clear that the data are not presented well. For this case, it is recommended to use the empirical distribution rather than using one of the parametric distributions of our list. In brief, if the best fitting distribution of our list has a good fit, the use of this parametric distribution is recommended to the user. Otherwise, the empirical distribution is recommended. We give some information about the parametric approach and the empirical distribution in order to interpret the modeling results.

2.3.2.1. Parametric Approach. We basically aim to select a parametric distribution from our list therefore this is called the parametric approach. We choose the distribution with the smallest AIC value if the p-value is large enough. Besides that, it is also possible to have more than one proper distribution with desirable p-values. This case occurs when more than one distribution is similar to the given data set. For instance, the Binomial distribution can approximate the rounded Normal distribution when the parameter n_b is sufficiently large. If the user gets more than one proper fit, she should

take into account the difference between the two AIC values. If it is below 2, it can be assumed that the fits are very similar. Therefore, distributions close to that of the best fitting distribution can be accepted as alternative models.

2.3.2.2. The Empirical Distribution. It is sometimes difficult to find a suitable parametric distribution for the given data. For the case that none of the parametric distributions fits to the data, the empirical distribution is recommended to the user. The empirical distribution is a discrete probability distribution estimated directly from the given data (See Chapter 7). This is called the nonparametric approach that we suggest to use in the absence of a proper parametric distribution.

3. PRELIMINARIES

In this part, we explain the maximum likelihood estimation (MLE) method for parameter estimation and Akaike's information criterion (AIC) for distribution selection. We also give the general properties and the advantageous sides of each method.

3.1. Maximum Likelihood Estimation

The MLE method is used to find the parameter values which maximize the likelihood function of a probability distribution. The probability mass function (PMF), for a parameter θ, is denoted by f(x, θ). This function gives the probability values for an integer x when the parameter is θ. To obtain the function of the unknown parameter vector, θ, where x is the collection of sample data, we define the joint function. The joint PMF of n independent and identically distributed (iid) observations from this experiment is the product of the individual PMFs. Considered as a function of the parameter θ for a fixed sample $X = (X_1, X_2, ..., X_n)$, it is named "likelihood function" and is denoted by L$(\theta|x)$. The likelihood function is given by the Equation 3.1:

$$L\left(\theta|x\right) = f\left(x_1, x_2, \ldots, x_n|\theta\right) = \prod_{i=1}^{n} f\left(x_i|\theta\right). \tag{3.1}$$

The log of the likelihood is preferred in practice due to its simplicity. Therefore we use the logarithmic form.

$$\Lambda(\theta) = \log L\left(\theta|x\right) = \sum_{i=1}^{n} \log f(x_i|\theta).$$

We would like to get the discrete distribution that has the highest possible likelihood. Therefore we need the parameter values that maximizes the log-likelihood function. It is called the maximum likelihood estimate $\hat{\theta}$

$$\hat{\theta} = \theta \;\; with \;\; L(\theta) = \max_{\theta} L(\theta).$$

For the cases that we can find a closed form solution for the Equation $\Lambda^{'}(\theta) = 0$, we use the analytical method, otherwise we solve the problem numerically using optimization algorithms.

3.1.1. Properties of the MLE

Estimates obtained by the method of maximum likelihood have some desirable properties [21]:

(i) *Sufficiency*: If a single sufficient statistic exists for θ, a maximum likelihood estimator of θ must be a function of the minimal sufficient statistic.

(ii) *Consistency*: When the sample size is large enough, the maximum likelihood estimator generates estimates very close to the true value of the parameter with high probability.

(iii) *Efficiency/ Asymptotical Optimality*: As the sample size increases, the MLE is the minimum variance unbiased estimators of θ, in other words the estimator has the smallest possible variance.

(iv) *Asymptotic Normality*: The maximum likelihood estimator converges to the normal distribution.

(v) *Invariance*: The maximum likelihood estimate is invariant under transformations of the parameters.

3.1.2. Solution Methods of the MLE

3.1.2.1. Analytical Solution. Analytical solution is used when it is possible to find the MLEs as closed form equations. We first differentiate the log-likelihood function $\Lambda(\theta)$ with respect to each parameter and set it equal to zero. Then we solve the equation to find the MLE.

$$\Lambda^{'}(\theta) = \frac{d\Lambda}{d\theta} = \frac{d(logL(\theta))}{d\theta} = 0.$$

We use analytical solution for the Poisson, Geometric and Uniform distribution.

3.1.2.2. Numerical Solution. It is usually not possible to obtain an closed form solution for the MLE estimate especially, when the distribution has many parameters or it is highly non-linear [15]. In this case, a numerical solution is required using nonlinear optimization algorithms. These algorithms enable to estimate parameter values that maximize the log-likelihood function numerically. Optimization algorithms are iterative. They search the parameter space starting from an initial parameter value θ_0 and generate a sequence of improved estimates until they find an optimal θ value [22]. We try to choose reasonable initial parameter values for an efficient search to reach the global maximum faster. Numerical methods are grouped as gradient-based and direct search based methods according to their search processes. Gradient based methods use analytic derivatives of the objective function, direct search methods do not explicitly use them.

(i) *Gradient-based search methods*: Given the initial value θ_0 and the log-likelihood function, Gradient based optimization methods such as Newton's method, BFGS and Brent try an iterative search to find a minimum of the given log-likelihood function. A build-in function of R, `optimize`() allows to make one dimensional optimization. Gradient based methods obtain information from the derivatives of the optimization function [23]. Therefore they find the optimum point much faster than the direct search methods hence they are computationally more efficient. The use of `optimize`() is explained in Section 4.1. We use gradient based search algorithms for the Binomial, negative Binomial and Logarithmic distributions.

(ii) *Direct search methods*: They are derivative-free algorithms which evaluate the objective function by using a step size and various directions [24]. Direct search methods are proper to use especially for the noisy or discontinuous functions for which the derivative of the optimization function is not available. To search the parameter space for the optimum point, direct search methods use systematic approaches however they are inefficient by means of computational time. In R, the function `optim`() carries out multidimensional optimization using the direct

search method called Nelder-Mead. The method is less sensitive to initial conditions than the gradient search method. The use of `optim`() is explained in Section 4.2. We use the direct search algorithm for the rounded Pareto, rounded Gamma and rounded Normal distributions.

3.1.3. Profile Log-likelihood Function

The profile log-likelihood function is a concentrated form of the log-likelihood function. It is obtained by replacing some of the parameters by its closed form MLE in the log-likelihood function. It enables to have a simplified form of the log-likelihood function by reducing the number of parameters and hence performs optimization for a reduced number of variables. Supposing that the parameters are denoted by $\theta = (\alpha, \beta)$. Then the log-likelihood function is:

$$\Lambda(\alpha, \beta) = \log L(\theta) = \log L(\alpha, \beta).$$

Let us suppose that, for α fixed, the parameter β has a closed form MLE that can be represented by $\hat{\beta} = \hat{\beta}(\alpha)$. By replacing β by its MLE, we obtain the profile log-likelihood function that only depends on the variable α.

$$\Lambda_p\alpha = \log L_p(\alpha) = \log L(\alpha, \hat{\beta}(\alpha)).$$

This new form simplifies the optimization procedure since Λ_p only depends on α. As result of maximizing the profile log-likelihood function, we get $\hat{\alpha}$, the estimated value of α. It is easy to find $\hat{\beta}$ by substituting the $\hat{\alpha}$ into the closed form equation for $\hat{\beta}$. We utilize the profile log-likelihood function forms for the Binomial and negative Binomial distribution.

3.2. Akaike's Information Criterion

The Akaike's information criterion (AIC) is a measure of the relative goodness of fit of a statistical model. It is the maximal value of the log-likelihood function $\log L(\hat{\theta})$ for that model. In the general case, the AIC formula is given by Equation 3.2 below.

$$AIC = -2 \log L(\hat{\theta}) + 2k. \tag{3.2}$$

The variable k which denotes the number of parameters is multiplied with a constant and added as a penalty value. The reason for that penalty is that more complex models with more parameters tend to match the data better than simple models with fewer parameters. Therefore, there is a tendency to choose distributions with many parameters which is called "overfitting" [25]. Hence, it is possible to obtain improper distributions with a large number of parameters, although it is not a good fit for the data. The term $2k$ is thus a penalty for models with more parameters [26].

As the value of the log-likelihood function increases, the fits get better and we obtain smaller AIC values. Therefore, the best fitting distribution is chosen as the one with the minimum AIC value. AIC results are written as ΔAIC scores which represent the closeness to the best fitting distribution. It is obtained by calculating the difference of the AIC values of each hypothesized distribution to the best fitting one. As a result, ΔAIC is by definition zero for the best fitting distribution and the other ΔAIC values indicate the closeness to the best fitting distribution. ΔAIC values up to 2 provide substantial support for good fits [27]. Therefore, we consider the candidate distributions that has ΔAIC values less than or equal to 2. It is important to note that, minimum AIC indicates the best fitting distribution however it does not guarantee a good fit. To find out whether the best fitting distribution represents the data well, goodness of fit testing is required (See Chapter 6).

3.2.1. AIC Properties

The most important properties of AIC are:

- AIC is a consistent criterion. Thus as sample size increases, it is more successful in selecting the correct model.
- It can compare models using different probability distributions.
- There is no requirement that the accepted model must be one of the models in the candidate list. Therefore it allows to use the alternative "empirical" distribution.
- It can take into account distributions with different number of parameters.
- The results are interpretable and comparisons can be done easily using the ΔAIC score that represents the closeness to the best fitting distribution.

4. ABOUT R

In this chapter, we explain the way of implementing numerical solution methods in the statistical software R. It includes both one dimensional and multidimensional optimization implementations depending on the number of parameters.

4.1. One Dimensional Optimization with the Function `optimize`()

For one-dimensional optimization, we use the function `optimize`() in R.

```
optres<-optimize(f=loglikfunc,x=x,interval=c(lwb,upb),maximum=T)
```

Here, x denotes the data vector and f is the objective function for optimization. This algorithm searches the user-defined interval for a minimum or maximum of the given function f. We always assign `TRUE` for the logical variable maximum as we would like to maximize f and find the parameter values that maximize the function f. We use the default method "Brent" in this study. It is not necessary to specify the name of the method in the statement, as it is the default method. We use one dimensional optimization for the parameter estimations of the Binomial, negative Binomial and Logarithmic distributions.

4.2. Multi-Dimensional Optimization with the Function `optim`()

For multi-dimensional optimization, we use the general purpose optimization function `optim`() in R.

```
res<-optim(par=starting.func,fn=loglikfunc,method=‘‘Nelder-Mead”,x=x,addsign=T)
```

In the statement of `optim`(), x denotes the data set and fn is the objective function. *par* holds the user defined initial values for the parameters which enable

starting optimization over this points. As the default algorithm of `optim`() performs minimization, we assign `TRUE` to the logical variable *addsign* which adds a negative sign to the objective function fn. Therefore we carry out maximization by minimizing the negative of fn. We used the default method "Nelder-Mead". It is not necessary to specify the name of the default method in the statement. We use two dimensional optimization for the parameter estimations of rounded Pareto, rounded Gamma and rounded Normal distributions.

5. MLE AND AIC CALCULATION FOR DISCRETE PROBABILITY DISTRIBUTIONS

In this part, we explain maximum likelihood estimation (MLE) of parameters and calculation of Akaike's information criterion (AIC) for all classical discrete and rounded continuous distributions of our list. First, we derive the formula of the log-likelihood function. Then, we find the MLE. Finally, we add an implementation part which includes our R code of maximum likelihood estimation and explanations about it. In this section, we consider six classical distributions and three rounded continuous distributions.

5.1. Classical Discrete Distributions

We have chosen six well-known classical discrete distributions which are often included in distribution fitting tools. For the MLE, we use if possible the closed form solution otherwise numerical optimization is necessary. After obtaining the MLE, it is easy to calculate the AIC as well.

5.1.1. Poisson Distribution

The Poisson distribution is one of the most important distributions in statistics. It is often used to model the number of events that occur in a specified region or time [28]. The parameter (λ) is the mean value of the distribution. The Poisson distribution has widespread applications including analysing traffic flow, fault prediction on electric cables, prediction of randomly occuring accidents and in reliability engineering [29].

The Poisson distribution is essentially a derived limiting case of the Binomial distribution [29]. In cases where the number of trials n_b of the Binomial distribution is sufficiently large and the probability of success p is sufficiently small, the Poisson distribution with parameter $\lambda = n_b p$ can be used as an approximation to that Bi-

nomial distribution. Different to the Binomial distribution, the Poisson distribution determines the probabilities of successes considering the population size as unbounded. The Poisson distribution is defined for all nonnegative integers and the parameter λ is positive. The PMF of the Poisson distribution with parameter λ is given by the formula.

$$f\left(x|\lambda\right) = \frac{e^{-\lambda}\lambda^{x}}{x!} \text{ for } x \geq 0 \text{ and } \lambda > 0.$$

We assume that $X = (X_1, X_2, ..., X_n)$ is an iid sample from a Poisson distribution with parameter λ. Because of the independence of the sample, we can write the joint PMF as product of the marginal PMFs:

$$f\left(x_1, x_2, \ldots, x_n|\lambda\right) = \frac{e^{-\lambda}\lambda^{x_1}}{x_1!}\frac{e^{-\lambda}\lambda^{x_2}}{x_2!}\cdots\frac{e^{-\lambda}\lambda^{x_n}}{x_n!} = \frac{e^{-n\lambda}\lambda^{\sum_{i=1}^{n}x_i}}{x_1!x_2!\ldots x_n!}$$

The likelihood function is therefore:

$$L\left(\lambda|x\right) = \prod_{i=1}^{n}\left(\frac{e^{-\lambda}\lambda^{x_i}}{x_i!}\right) = \frac{e^{-n\lambda}\lambda^{\sum_{i=1}^{n}x_i}}{\prod_{i=1}^{n}\left(x_i\right)!}.$$

Due to its simpler form, we take the logarithm of likelihood function to get the log-likelihood.

$$\log L\left(\lambda|x\right) = \Lambda = -n\lambda + \sum_{i=1}^{n}x_i \log\lambda - \sum_{i=1}^{n}\log(x_i!). \tag{5.1}$$

We can replace the factorial expression by the Gamma (Γ) function as $x! = \Gamma(x+1)$ and get

$$\log L\left(\lambda|x\right) = \Lambda = -n\lambda + \sum_{i=1}^{n}x_i \log\lambda - \sum_{i=1}^{n}\log(\Gamma(x+1)).$$

In order to find the MLE for the parameter $\hat{\lambda}$, we differentiate the log-likelihood function with respect to λ and set the derivative equal to zero. We need to find the

stationary point which is the global maximum. In Equation 5.1, the $\sum_{i=1}^{n}\log(x_i!)$ term depends only on x and is therefore insignificant for the derivative. We get:

$$\frac{d\Lambda}{d\lambda} = -n + \frac{\sum_{i=1}^{n} x_i}{\lambda} = 0.$$

It is seen from Equation 5.2 that we have the solution $\hat{\lambda}$ which is the MLE.

$$\hat{\lambda} = \frac{\sum_{i=1}^{n} x_i}{n} = \bar{x}. \tag{5.2}$$

We also check the concavity in order to verify existence of a unique maximum value at λ. As the second derivative is negative, we infer that the log-likelihood function is concave and we get a unique global maximum.

$$\frac{d^2\Lambda}{d\lambda^2} = -\frac{\sum_{i=1}^{n} x_i}{\lambda^2} < 0.$$

5.1.1.1. R Implementation. In R, we define the function `MLEPois(x)` which calculates the value $\hat{\lambda}$ and the AIC. `MLEPois(x)` gets the data vector x as input and returns a vector holding AIC and $\hat{\lambda}$. First, $\hat{\lambda}$ is calculated using the closed form solution 5.2. After that, `lpois(`λ`, x)` is called and the log-likelihood function value is found for $\hat{\lambda}$. We need that log-likelihood value in order to calculate the AIC. The R code of the `MLEPois(x)` function and an example are given in the code below.

Table 5.1. R Code for MLE-AIC of Poisson Distribution.

```
MLEPois<-function(x) {
  if (min(x)<0){
    r<-c(NA,NA)
    warning("Warning Poisson: Negative Values")
  } else{
    lpois<-function (lambda, x) {
      n<-length(x)
      return((-n*lambda)+sum(x)*log(lambda)- sum(lgamma(x+1)))
    }
```

Table 5.2. R Code for MLE-AIC of Poisson Distribution (cont.).

```
lambda_hat<-mean(x)
    AIC<--2*lpois(lambda_hat,x)+2*1; r<- c(AIC,lambda_hat)
  }
names(r)<-c( "AIC "," lambda_hat ")
  return(r)
}
#Example
x<-rpois(1000,lambda=1.6)
MLEPois(x)
    AIC   lambda_hat
3102.636     1.557
```

5.1.2. Binomial Distribution

The Binomial distribution models the number of successes in n_b independent trials where each trial can result only in success or failure, live or dead, head or tail etc. The parameter n_b denotes the fixed number of trials, p the probability of success in each trial.

The Binomial distribution is commonly used for testing statistical probabilities and significance, and is a good way of visually detecting unexpected values. Common usage areas for Binomial distribution include quality control, public opinion surveys, medical research and insurance problems [30]. For example, the Binomial distribution can describe the number of defective computer chips in a lot of n chips, the number of people in a clinical study that died of heart disease or the number of animals in a population that carry a certain genetic trait.

The Binomial distribution is defined for nonnegative integers $x \leq n_b$. The parameter n_b is a positive integer and p is a real positive number within $(0, 1)$. The PMF of the Binomial distribution with parameters n_b and p is given by the formula.

$$f\left(x|n_b, p\right) = \binom{n_b}{x} p^x (1-p)^{n_b - x} \text{ for }, \;\; x \leq nb \text{ and } 0 < p < 1, \;\; p \in R.$$

This formula can also be written in factorial form. Then we can replace the factorial by the Gamma (Γ) function as $x! = \Gamma(x+1)$ and get

$$\begin{aligned} f\left(x|n_b,p\right) &= \frac{n!}{x!(n-x)!}p^x(1-p)^{n_b-x} \\ &= \frac{\Gamma(n+1)}{\Gamma(x+1)\Gamma(n-x+1)}p^x(1-p)^{n_b-x}. \end{aligned} \tag{5.3}$$

We assume that $X = (X_1, X_2, ..., X_n)$ is an iid sample from a Binomial distribution with parameters n_b and p. The likelihood function is therefore:

$$\begin{aligned} L\left(n_b,p|x\right) &= \prod_{i=1}^{n}\left(\binom{n_b}{x_i}p^{x_i}(1-p)^{n_b-x_i}\right) \\ &= \left(\prod_{i=1}^{n}\binom{n_b}{x_i}\right)p^{\sum_{i=1}^{n}x_i}(1-p)^{nn_b-\sum_{i=1}^{n}x_i}. \end{aligned}$$

The log-likelihood function is therefore equal to

$$\log L\left(n_b,p|x\right) = \Lambda = \sum_{i=1}^{n}\log\binom{n_b}{x_i} + \sum_{i=1}^{n}x_i\log p + \left(nn_b - \sum_{i=1}^{n}x_i\right)\log\left(1-p\right). \tag{5.4}$$

We take the derivate of the log-likelihood function with respect to p and by setting it to zero we obtain the maximum likelihood estimator of p. We get:

$$\frac{d\Lambda}{dp} = \frac{1}{p}\sum_{i=1}^{n}x_i - \left(nn_b - \sum_{i=1}^{n}x_i\right)\frac{1}{1-p} = 0.$$

We easily get the closed form solution

$$\hat{p} = \frac{\sum_{i=1}^{n}x_i}{n_b n} = \frac{\bar{x}}{n_b}. \tag{5.5}$$

It is a unique global maximum as the second derivative is negative.

$$\frac{d^2\Lambda}{dp^2} = -\frac{1}{p^2}\sum_{i=1}^{n}x_i - \left(nn_b - \sum_{i=1}^{n}x_i\right)\frac{1}{(1-p)^2} < 0.$$

The log-likelihood function is difficult to differentiate with respect to n_b. As it is seen from 5.4, the PMF includes a combinatorial expression which can also be written using the Γ (Gamma) function as in 5.3. Differentiation of Γ is complicated and not a practical method for estimation. Therefore, we find $\hat{n_b}$ by performing numerical optimization instead of searching for a closed form solution.

For optimization, we use the profile log-likelihood function which is the simplified form of log-likelihood function by reducing the number of parameters. In 5.4, we replace p by its MLE 5.5 and get the profile log-likelihood function which depends only on the parameter n_b.

$$\log L\left(n_b|x\right) = \Lambda_p = \sum_{i=1}^{n} \log \binom{n_b}{x} + \sum_{i=1}^{n} x_i \log\left(\frac{\bar{x}}{n_b}\right) + \left(n n_b - \sum_{i=1}^{n} x_i\right) \log\left(1 - \frac{\bar{x}}{n_b}\right).$$

5.1.2.1. R Implementation. In R, we define the function `MLEBinom(x)` which requires the data vector x as input and returns the estimated parameter values and AIC value as output. As $\hat{p}$ is obtained analytically by the closed form Equation 5.5, we replace the parameter p with that formula given by Equation 5.4 and obtain the profile log-likelihood function (`lbinomprofile(n_b, x)`). The parameter n_b is obtained numerically by optimization of that function using `optimize()` which is a build in function in R. As left border of the starting interval, we choose $max(x)$ since n_b must be larger or equal to the maximum value of x.

As right border of the starting interval, we use 10 times the left border. If $\hat{n_b}$ is found exactly equal to the upper bound of an interval, we multiply both interval borders by 10 and try again using a while loop. We stop the iterations if the right border is larger than 10^{10}.

As result, we obtain a non integer value for $\hat{n_b}$ as the optimization function is designed for continuous optimization. We obtain two discrete candidate values by rounding upward and downward. Then, we take the candidate of $\hat{n_b}$ with the larger

value of the profile log-likelihood function. Eventually, by calling `MLEBinom(x)`, the user gets the MLE for the parameters (n_b, p) and the AIC value. The R code of `MLEBinom(x)` and an example how to use it are given in the code below.

Table 5.3. R Code for MLE-AIC of Binomial Distribution.

```
MLEBinom<-function (x) {
  n<-length(x)
  if ((min(x)<0)==TRUE || is.integer(x[1]:x[n])==FALSE){
    warning("Negative or Noninteger Value")
    r<- c(NA,NA,NA)
  } else{
    sx<-sum(x); mn<-mean(x)
    lprofilebinom<-function(nb,x){
      return(sum(lchoose(nb,x))+sx*log(mn/nb)+(n*nb-sx)*log(1-(mn/nb)))
    }
    lwb<-max(x); upb<-lwb*10
    while(upb<1e+10){
      optres<-optimize(lprofilebinom,x=x,c(lwb,upb),maximum=T)
      nbopt<-optres$maximum
      if(nbopt<=(upb-(0.1))){
        break()
      }
      if(nbopt>=1e+10){
        warning("MLE found maximal nb")
        break()
      }
      lwb<-upb; upb<-upb*10
    }
    res1<-lprofilebinom(ceiling(nbopt),x)
    res2<-lprofilebinom(floor(nbopt),x)
    if(res1>=res2){
      res<-res1; nb<-ceiling(nbopt)
    }else{
      res<-res2; nb<-floor(nbopt)
    }
    p_hat<-mn/nb; AIC<--2*(res)+2*2
    r<- c(AIC, nb, p_hat)
```

Table 5.4. R Code for MLE-AIC of Binomial Distribution (cont.).

```
    if(upb>=1e+10){
      warning("max(x) is out of allowed interval")
      r<-c(Inf,NA,NA)
    }
  }
  names(r)<-c(" AIC ","nb", "p_hat")
  return(r)
}

#Example
x<-rbinom(1000,size=10,prob=0.5)
MLEBinom(x)
   AIC       nb      p_hat
3719.780  10.000  0.502
```

5.1.3. Negative Binomial Distribution

The negative Binomial distribution models the number of trials or failures until a predefined number of successes in a sequence of independent trials occur. The parameters of the negative Binomial distribution are p, the success probability which is constant for each trials, and r, the number of successes.

Success and failure terms for events can be different from our perception when applied to real-world problems [31]. For instance, to model the number of days a certain machine works before it breaks down, the first breakdown can be called "success", the number of days the machine works until that time is called "failure". However, to model the number of goal attempts a sportsman makes before scoring a goal, we name the unsuccessful attempt "failure", and scoring a goal "success" as used in daily life. For this reason, we are only interested in the number of trials.

The negative Binomial distribution is defined for nonnegative integers where the parameter r is a positive integer and p is a real number in $(0, 1)$. The PMF of the

negative Binomial distribution with parameters r and p is given by the formula.

$$f\left(x|r,p\right)=\binom{r+x-1}{x}p^{r}(1-p)^{x}\ for\ r\geq 1,\ x\geq 0\ and\ 0<p<1,\ \ p\in R.$$

This formula can also be written in factorial form. Then we can replace the factorial by the Gamma (Γ) function as $x!=\Gamma(x+1)$ and get

$$\begin{aligned} f\left(x|r,p\right) &= \frac{(r+x-1)!}{x!(r-1)!}p^{r}(1-p)^{x} \\ &= \frac{\Gamma(r+x)}{\Gamma(x+1)\Gamma(r)}p^{r}(1-p)^{x}. \end{aligned} \tag{5.6}$$

We assume that $X=(X_1,X_2,...,X_n)$ is an iid sample from a negative Binomial distribution with parameters r and p. The joint PMF of independent samples and thus also the likelihood function is

$$\begin{aligned} L\left(r,p|x\right) &= \prod_{i=1}^{n}\left(\binom{r+x_i-1}{x_i}p^{r}(1-p)^{x_i}\right) \\ &= \left(\prod_{i=1}^{n}\binom{r+x_i-1}{x_i}\right)p^{rn}(1-p)^{\sum_{i=1}^{n}x_i}. \end{aligned} \tag{5.7}$$

Taking the logarithm of Equation 5.7, we get the log-likelihood function.

$$\log L\left(r,p|x\right)=\Lambda=\sum_{i=1}^{n}\log\binom{r+x_i-1}{x_i}+rn\log p+\sum_{i=1}^{n}x_i\log\left(1-p\right). \tag{5.8}$$

In order to find the maximum likelihood estimator of p, we take the derivative of Equation 5.8 and set it to zero. We get:

$$\frac{d\Lambda}{dp}=\frac{rn}{p}-\frac{\sum_{i=1}^{n}x_i}{1-p}=0.$$

We easily get the closed form solution for $\hat{p}$, the maximum likelihood estimator of p:

$$\hat{p} = \frac{\hat{r}n}{\hat{r}n + \sum_{i=1}^{n} x_i} = \frac{\hat{r}}{\hat{r} + \bar{x}}. \tag{5.9}$$

We also check the second derivative. As it is negative we know that at $\hat{p}$ the function takes its unique global maximum.

$$\frac{d^2\Lambda}{dp^2} = -\frac{rn}{p^2} - \frac{\sum_{i=1}^{n} x_i}{(1-p)^2} < 0.$$

In order to estimate the integer parameter r, we can not obtain a closed form solution as the derivative of the Gamma function (See Equation 5.6) is analytically difficult. Therefore, we find $\hat{r}$ numerically by one dimensional optimization. We obtain the simplified form called profile log-likelihood function 5.10 by replacing all p by Equation $\hat{p}$ given in 5.8. As a result, we get the profile likelihood function that includes only the parameter r.

$$\log L\left(r|x\right) = \Lambda_p = \sum_{i=1}^{n} \log \binom{r + x_i - 1}{x_i} + rn \log\left(\frac{r}{r+\bar{x}}\right) + \sum_{i=1}^{n} x_i \log\left(\frac{\bar{x}}{r+\bar{x}}\right). \tag{5.10}$$

5.1.3.1. R Implementation. The function `MLENBinom(x)` includes maximum likelihood estimation and the computation of the AIC. As input, the vector holding the x data is required. First, $\hat{r}$ is computed by maximizing the profile log-likelihood function `lprofilenbinom(r, x)` given in 5.10. As the upperbound of the starting interval, we choose 1, the minimum value of r. Similar to the Binomial distribution, we use one dimensional optimization with `optimize()` inside a for loop that uses gradually expanding intervals to find $\hat{r}$. If $\hat{r}$ is found exactly equal to the upper bound of the given interval, the program continues to search for $\hat{r}$ by increasing both the upper and lower bounds by 10 times. The iteration is stopped, if the upper bound exceeds 10^{10}. As the result of optimization, we get a noninteger value for $\hat{r}$. We calculate two discrete values using the two neighboring integers. The aim is to choose the one which gives the larger value for the profile log-likelihood function. According to it, $\hat{r}$ is chosen and

the profile log-likelihood value is found. Then, $\hat{p}$ is found easily by the closed form Equation 5.9. As the last step, AIC is calculated using the profile log-likelihood value. As a result, `MLENBinom(x)` returns a vector holding the estimated parameters and the AIC value. The R code of the function and an example how to use it are given in the code below.

Table 5.5. R Code for MLE-AIC of Negative Binomial Distribution.

```
MLENBinom<-function (x) {
  n<-length(x)
  if ((min(x)<0)==TRUE || is.integer(x[1]:x[n])==FALSE){
    warning("Negative or Noninteger Value")
    res<-c(NA,NA,NA)
  } else{
    mn<-mean(x); sx<-sum(x)
    lprofilenbinom<-function(r,x){
      return(sum(lchoose((r+x-1),x))+(r*n)*log(r/(mn+r))
             +sx*(log(mn/(mn+r))))
    }
    lwb<-1; upb<-lwb*10
    while(upb<1e+10){
      resopt<-optimize(lprofilenbinom,x=x,c(lwb,upb),maximum=T)
      ropt<-resopt$maximum
      if(ropt<=(upb-(0.1))){
        break()
      }
      lwb<-upb; upb<-upb*10
    }
    res1<-lprofilenbinom(ceiling(ropt),x)
    res2<-lprofilenbinom(floor(ropt),x)
    if(res1>=res2){
      res<-res1; r<-ceiling(ropt)
    } else{
      res<-res2; r<-floor(ropt)
    }
    p_hat<-r/(mn+r); AIC<--2*(res)+2*2
    res<- c(AIC,r,p_hat)
```

Table 5.6. R Code for MLE-AIC of Negative Binomial Distribution (cont.).

```
if(upb>=1e+10){
      warning("r is out of allowed interval")
      r<-c(Inf,NA,NA)
    }
  }
  names(res)<-c(" AIC ","r", "p_hat")
  return(res)
}
#Example
x<-rnbinom(1000,size=15,prob=0.4)
MLENBinom(x)
   AIC        r       p_hat
6826.197   15.000    0.401
```

5.1.4. Geometric Distribution

The Geometric distribution is a special case of the negative Binomial distribution when $r = 1$. Therefore, X denotes the failures until the first success. The parameter r is fixed to 1 hence p is the only parameter of the Geometric distribution. It represents the constant success probability for the Bernoulli trials. $(p - 1)$ shows the constant probability for the failures.

The Geometric distribution is defined for nonnegative integers when the constant success probability parameter p is a real number within $(0, 1)$. We can directly find the log-likelihood function using the log-likelihood formula of the negative Binomial distribution. We set p equal to 1 in Equation 5.8 and we get:

$$\log L\left(p|x\right) = \Lambda = n\log p + \sum_{i=1}^{n} x_i \log\left(1 - p\right).$$

Similar to the log-likelihood function, we can also get the closed form solution for $\hat{p}$

easily by setting it equal to 1 in Equation 5.9 :

$$\hat{p} = \frac{n}{\sum_{i=1}^{n} x_i + n} = \frac{1}{\bar{x} + 1}. \tag{5.11}$$

5.1.4.1. R Implementation. In R, we define the function `MLEGeom(x)` to calculate $\hat{p}$ and the AIC. It gets the data as input and calculates $\hat{p}$ using the closed form Equation 5.11. Then, function `lgeom(p̂, x)` is called to calculate the log-likelihood value for $\hat{p}$ and the AIC. As a result, `MLEGeom(x)` returns a vector holding the values of AIC and the $\hat{p}$. The R code of the function `MLEGeom(x)` and an example how to use it are given in the code below.

Table 5.7. R Code for MLE-AIC of Geometric Distribution.

```
MLEGeom<-function(x) {
  if (min(x)<0){
    r<-c(NA, NA)
    warning("Negative Value")
  }else{
    lgeom<-function(p, x) {
      n<-length(x)
      return(n*log(p)+sum(x)*log(1-p))
    }
    x_hat<-mean(x)
    p_hat<- 1/(x_hat+1)
    AIC<--2*lgeom(p_hat,x)+2*1
    r<- c(AIC, p_hat)
  }
  names(r)<-c("AIC","p_hat")
  return(r)
}
#Example
x<-rgeom(1000, prob=0.85)
MLEGeom(x)
    AIC     p_hat
1050.752    0.840
```

5.1.5. Discrete Uniform Distribution

The Discrete Uniform distribution models integers that are equally likely to occur. It is parametrized by upper bound a and lower bound b and defined in the interval $[a, b]$. Each value has the probability p equal to $p = (1/(b-a+1))$. A common usage of discrete Uniform distribution is encountered in computer programming which focuses on equal-probability random selections between a number of choices [32].

The PMF of the discrete Uniform distribution with parameters a and b is given by the formula.

$$f\left(x|a,b\right) = \frac{1}{b-a+1}\ I_{(a,b)}x \ \text{ for } \ a \leq x \leq b.$$

We assume that $X = (X_1, X_2, ..., X_n)$ is an iid sample from a discrete Uniform distribution with parameters a and b. Considering that the trials are independent, we can write the joint PMF as in 5.12. In the formula, $I_{(a,b)}x$ denotes the indicator function. If x is outside of the given interval, the PMF equals to zero.

$$L\left(a,b|x\right) = \prod_{i=1}^{n} \frac{1}{b-a+1}\ I_{(a,b)}x_i = \frac{1}{(b-a+1)^n}\ I_{(a,b)}x_i \text{ for } i = 1, ..., n. \tag{5.12}$$

The log-likelihood function of the Discrete Uniform distribution is obtained by taking the logarithm of Equation 5.12.

$$\log L(a,b|x) = \Lambda = -n\log\left(\frac{1}{b-a+1}\right) + \log\left(I_{(a,b)}x_i\right). \tag{5.13}$$

To get MLE for b, we take the first derivative of 5.13 with respect to b for $b \geq \max(x_i)$.

$$\frac{d\Lambda}{db} = \frac{-n}{b-a+1} < 0. \tag{5.14}$$

As the equation 5.14 is negative, the log-likelihood function is decreasing when $(b -$

$a) > 0$. Therefore, the smallest possible value of b and the largest possible value of a maximizes the log-likelihood function. As, $a = min(x_1, ..., x_n)$ and $b = max(x_1, ..., x_n)$, the closed MLEs of a and b are:

$$\hat{a} = x_{(1)} = min(x)$$
$$\hat{b} = x_{(n)} = max(x). \tag{5.15}$$

5.1.5.1. R Implementation. In R, we define the function `MLEUnif(x)` to calculate the values $\hat{a}$, $\hat{b}$ and the AIC. As we have closed form Equations 5.15 for $\hat{a}$ and $\hat{b}$, we can easily calculate them. Then, the function `lunif(a,b,x)` is called and the log-likelihood function value is computed for $a = \hat{a}$ and $b = \hat{b}$. Then, AIC is calculated using the log-likelihood value. As a result of `MLEUnif(x)`, a vector which holds the values of AIC, $\hat{a}$ and $\hat{b}$ is returned. The R code of the function and an example are given below.

Table 5.8. R Code for MLE-AIC of Uniform Distribution.

```
MLEUnif <-function (x) {
  if (min(x)<0){
    r<-c(NA,NA,NA)
    warning("Negative Value")
  } else{
    lunif<-function(a,b,x) {
      n<-length(x)
      return(-n*(log(b-a+1)))
    }
    lb<-min(x); ub<-max(x)
    AIC<--2*lunif(lb,ub,x)+2*2
    r<- c(AIC,lb,ub)
  }
  names(r)<-c("AIC","lower bound","upper bound")
  return(r)
}
#Example
x<-round(runif(1000)*10+20)
MLEUnif(x)
   AIC   lower_bound  upper bound
4799.791   20.000     30.000
```

5.1.6. Logarithmic Distribution

The Logarithmic distribution also known as the Logarithmic series (log-series) distribution is based on the standard power series expansion of the natural logarithm function [33]. The log-series distribution is commonly used for modeling plant species, animal species, population growth and also for microbiology and economic applications [34].

It is defined for positive integers when the shape parameter p is a real number within $(0, 1)$. The PMF of the Logarithmic distribution with parameter p is given by the following equation.

$$f\left(x|p\right) = \frac{-1}{\log\left(1-p\right)}\frac{p^x}{x} \text{ for } x \geq 1, \;\; 0 < p < 1.$$

We assume that $X = (X_1, X_2, ..., X_n)$ is an iid sample from a Logarithmic distribution with parameter p. The likelihood function is:

$$L\left(p|x\right) = \prod_{i=1}^{n}\left(\frac{1}{-\log\left(1-p\right)}\frac{p^{x_i}}{x_i}\right) = \frac{p^{\sum_{i=1}^{n}x_i}}{\prod_{i=1}^{n}x_i(-\log\left(1-p\right))^n}. \tag{5.16}$$

To simplify the form, we take the logarithm of 5.16 and get the log-likelihood function.

$$\log L\left(p|x\right) = \Lambda = \sum_{i=1}^{n}x_i \log p - n\log\left(-\log\left(1-p\right)\right) - \sum_{i=1}^{n}\log x_i. \tag{5.17}$$

We take the derivative of 5.17 with respect to p and set it to zero. However, we can not get a closed form equation for the MLE of $\hat{p}$. Therefore, we find $\hat{p}$ numerically by using an optimization method.

$$\frac{d\Lambda}{dp} = \frac{\sum_{i=1}^{n}x_i}{p} + \frac{n}{\log(1-p)}\frac{d}{dp}(-\log(1-p)) = 0$$

$$\frac{d\Lambda}{dp} = \frac{n\bar{x}}{p} - \frac{n}{\log(1-p)}\frac{1}{(1-p)} = 0.$$

5.1.6.1. R Implementation. We define the function `MLELog(x)` to calculate the value of $\hat{p}$ and the AIC. Since we can not find a closed form solution for the MLE of p, we find it numerically and carry out one dimensional optimization. The interval of p is defined as $((10^{-8}), 1-(10^{-8}))$. Optimization is carried out using the function `optimize()` for `llog(p,x)`. The maximized value of the log-likelihood function and the maximizing value of p are found as result. Then, AIC is calculated according to them. Briefly, `MLELog(x)` returns a vector holding $\hat{p}$ and the AIC. The R code of the function and an example how to use it are given in the code below.

Table 5.9. R Code for MLE-AIC of Logarithmic Distribution.

```
MLELog<-function(x){
  if (min(x)<=0){
    r<-c(NA,NA)
    warning("Negative Value or Zero")
  } else{
    llog<-function(p, x) {
      n<-length(x)
      res<-sum(x)*log(p)-n*log(-log(1-p))-sum(log(x))
      if(is.infinite(-res)){
        res<--1e+10
      }
      return(res)
    }
    plw<-(1e-08); p.interval<-c(plw,(1-plw))
    res<-optimize(llog,x=x,p.interval,maximum=T)
    if(res$objective==-1e+10){
      res$objective<--Inf
    }
    AIC<--2*(res$objective)+2*1
    r<- c( AIC ,res$maximum)
  }
```

Table 5.10. R Code for MLE-AIC of Logarithmic Distribution (cont.).

```
names(r)<-c(" AIC "," p_hat")
return(r)
}
#Example
library(VGAM)
x<-rlog(1000, prob=0.54)
MLELog(x)
  AIC      p_hat
1993.719   0.553
```

5.2. Rounded Continuous Distributions

Considering the number of popular discrete distributions in statistics, we realize that they are limited compared to the number of popular continuous distributions. In addition, the discrete distributions we have considered above have geometric or lower tails. Therefore, problems may arise to find a proper distribution for data having different shapes. Therefore, we focus on the idea of defining a discrete distribution using a continuous distribution. By doing that, we increase the number of possible discrete distributions and include new shapes in our list, therefore there is more hope to find a good fit for the data. Our idea is based on the possibility of defining a new discrete distribution PMF using the CDF of a continuous distribution. It is important to note that this approach is not discussed in the literature, but for every continuous distribution, it is possible to define a corresponding discrete distribution. We call these newly defined discrete distributions "Rounded Continuous Distributions" and we include three of them in this thesis.

5.2.1. Constructing a Discrete Distribution Using a Continuous Distribution

For a continuous random variate Y we define the discrete random variate X as the value of Y rounded to the closest integer. We can easily calculate the PMF $f(x)$

using the CDF $G(y)$.

$$f(x) = P(X = x) = \int_{x-0.5}^{x+0.5} G(x)dx = G(x + 0.5) - G(x - 0.5).$$

It is not difficult to see that

$$\sum_{-\infty}^{\infty} f(x) = \int_{-\infty}^{\infty} G(y)dy = 1.$$

You can see the discrete rounded Normal PMF together with the Normal density in Figure 5.1. For discrete data sets, probabilities can be calculated from the CDF of a continuous distribution, hence the rounded form is possible to be used as a new discrete distribution.

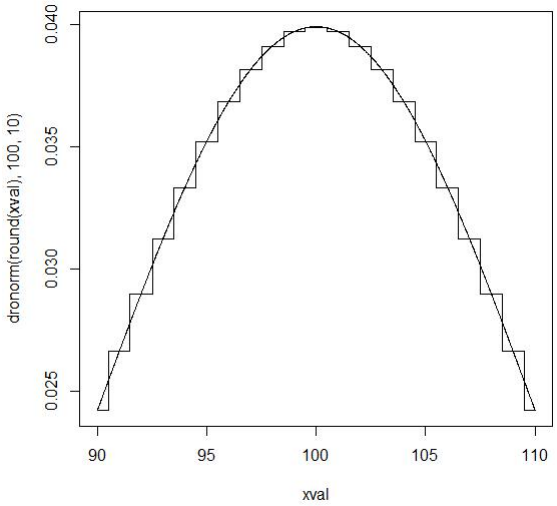


Figure 5.1. Rounded Normal Plot-Normal Curve Comparison.

5.2.2. Rounded Normal Distribution

The Normal distribution is a continuous distribution with parameters μ and σ which is the location and the scale parameters respectively. It is used throughout statistics, natural sciences and social sciences [35]. Due to the Central Limit Theorem (CLT), the Normal distribution can be used to approximate a large variety of distributions [36]. Therefore, it is useful to include the rounded form in our discrete

distribution pool. We obtain its PMF using the CDF of the Normal distribution. We call this new discrete distribution the rounded Normal distribution.

The CDF of the Normal distribution with parameter μ and σ is:

$$G(y|\mu,\sigma) = \phi\left(\frac{y-\mu}{\sigma}\right) \quad \text{for } y,\ \mu \in R \text{ and } \sigma > 0,$$

where ϕ denotes the CDF of the standard Normal distribution. The PMF of the rounded Normal distribution is found by using formula given by Equation 5.18:

$$\begin{aligned} f(x|\mu,\sigma) &= G((x+0.5)|\mu,\sigma) - G((x-0.5)|\mu,\sigma) \qquad (5.18) \\ &= \phi\left(\frac{x+0.5-\mu}{\sigma}\right) - \phi\left(\frac{x-0.5-\mu}{\sigma}\right). \end{aligned}$$

We assume that $X = (X_1, X_2, ..., X_n)$ is an iid sample from a rounded Normal distribution. The likelihood function is:

$$L(\mu,\sigma|x) = \prod_{i=1}^{n} \left\{G((x_i+0.5)|\mu,\sigma) - G((x_i-0.5)|\mu,\sigma)\right\}. \qquad (5.19)$$

Taking the logarithm of Equation 5.19, we get the log-likelihood function.

$$\log L(\mu,\sigma|x) = \sum_{i=1}^{n} \left\{\log(G((x_i+0.5)|\mu,\sigma) - G((x_i-0.5)|\mu,\sigma))\right\}.$$

5.2.2.1. R Implementation. We code the function `MLEroNorm(x)` to calculate $\hat{\mu}$, $\hat{\sigma}$ and the AIC. It returns a vector holding the AIC and the estimates. We define the PMF `dronorm(x, mean, sd)` and the log-likelihood function `lronorm()`. Then, μ and σ are estimated numerically by minimizing `lronorm()`. Two dimensional optimization is carried out for μ and σ by the build-in function `optim()` using the method "Nelder-Mead". To find starting values easily, we use the estimator function of the Normal

distribution `MLENorm(x)`. For details of two dimensional optimization in R, see Section 4.2. The R code of the functions `MLEroNorm(x)`, `MLENorm(x)` and an example how to use it are given in the code below. To use a suitable discrete data in our example, we first generate a data set from the Normal distribution and then round it to integers.

Table 5.11. R Code for MLE-AIC of Rounded Normal Distribution.

```
MLENorm <- function(x){
  n <- length(x)
  lnorm <- function(mu,sigma,x){
    return(-n*log(sqrt(2*pi))-n*log(sigma)
           -sum((((x)-mu)^2)/(2*(sigma)^2)))
  }
  mu_hat <- mean(x)
  sigma_hat <- sqrt((1/n)*sum((x-mu_hat)^2))
  AIC <- -2*lnorm(mu_hat,sigma_hat,x)+2*2
  r <- c(AIC,mu_hat,sigma_hat)
  names(r) <- c("AIC","location","scale")
  return (r)
}
MLEroNorm<-function(x){
  dronorm<-function(x,mean=0,sd=1){
    n<-length(x)
    x<-round(x)
    res<-(pnorm((x+0.5),mean,sd)-pnorm((x-0.5),mean,sd))
    if(is.nan(sum(res))){
      res<-rep(NA,n)}
    return(res)
  }
  lronorm<-function(theta,x,addsign=F){
    res<-sum(log(dronorm(x,theta[1],theta[2])))
    if(is.infinite(-res)){
      res<--1e+10
    }
    if(addsign) {res<--res}
    return(res)
  }
  stn<-MLENorm(x)
```

Table 5.12. R Code for MLE-AIC of Rounded Normal Distribution (cont.).

```
  stn1<-stn[2]; stn2<-stn[3]
  starting.norm<-c(stn1,stn2)

  res<-optim(starting.norm,lronorm,method="Nelder-Mead",x=x,addsign=T)
  if(res$val==1e+10){
    warning("Loglikelihood Value is Inf")
    res$val<-Inf
  }
  AIC<--2*(-res$value)+2*2; r <- c(AIC,res$par)
  names(r)<-c("AIC","location_rnd","scale_rnd")
  return (r)
}
#Example
x<-round(rnorm(1000)*10+100)
MLEroNorm(x)
  AIC     location_rnd  scale_rnd
7495.235  100.535       10.240
```

5.2.3. Rounded Pareto Distribution

The Pareto distribution is a skewed, heavy-tailed distribution that is sometimes used to model the distribution of incomes and other financial variables [37]. It has the parameters a and k which represent the shape and the location respectively. By using the rounded Pareto distribution we include a heavy tailed distribution in our distribution pool.

We assume that $X = (X_1, X_2, ..., X_n)$ is an iid sample from a rounded Pareto distribution. The likelihood function is:

$$L(a, k|x) = \prod_{i=1}^{n} \{G((x_i + 0.5)|a, k) - G((x_i - 0.5)|a, k)\}. \tag{5.20}$$

Taking the logarithm of 5.20, we get the log-likelihood function.

$$\log L(a, k|x) = \sum_{i=1}^{n} \{\log(G((x_i + 0.5)|a, k) - G((x_i - 0.5)|a, k))\}.$$

5.2.3.1. R Implementation. We define the function `MLEroPareto(x)` to calculate the value of $\hat{a}$ and $\hat{k}$ and the AIC. First of all, we code the PMF `dropareto(x, k, a)`. Then we define the log-likelihood function `lropareto()`. By minimizing the function `lropareto()`, we get the estimates for the parameters. For the numerical solution, two dimensional optimization is executed using the function `optim()` and the method "Nelder-Mead". Starting points are calculated easily from the estimator of the Pareto distribution `MLEPareto(x)`. Eventually, `MLEroPareto(x)` returns a vector which holds the values of AIC, $\hat{a}$ and $\hat{k}$. The R code of the functions `MLEroPareto(x)`, `MLEPareto(x)` and an example how to use it are given in the code below. In our example, we generate a data set from the Pareto distribution, then we round it to get discrete data. The random generation function of the Pareto distribution in included in the library VGAM. Therefore, we call it before generating the data.

Table 5.13. R Code for MLE-AIC of Rounded Pareto Distribution.

```
MLEPareto<-function(x){
  n<-length(x)
  lpareto<-function (k, a, x) {
    return(n*log(a)+a*n*log(k)-(a+1)*sum(log(x)))
  }
  k_hat<-min(x); a_hat<-n/(sum(log(x/k_hat)))
  res<-lpareto(k_hat, a_hat, x)
  if(is.infinite(-res)){res<--Inf}
  AIC<--2*(res)+2*2
  r<- c(AIC, k_hat, a_hat)
  names(r)<-c(" AIC "," location", "shape")
  return(r)
}
MLEroPareto<-function(x){
  if (min(x)<=0){
    r<-c(NA, NA, NA)
    warning("Negative Values or Zero")
  }
  else{
```

Table 5.14. R Code for MLE-AIC of Rounded Pareto Distribution (cont.).

```
dropareto<-function(x,k=1,a=1){
      x<-round(x)
      ppareto<-function(x,k,a){
        return((1-(k/x)^(a))*(x>=k)+0*(x<a))
      }
      res<- (ppareto((x+0.5),k,a)
             -ppareto((x-0.5),k,a))
      if(is.nan(sum(res))){
        n<-length(x)
        res<-rep(NA,n)}
      return(res)
    }
    lropareto<-function(theta,x,addsign=F){
      res<-sum(log(dropareto(x,theta[1],theta[2])))
      if(is.nan(res)){res<-NA}
      if(is.infinite(-res)){res<-(-1e+10)}
      if(addsign) {res <- -res}
      return(res)
    }
    st<- MLEPareto(x)
    st1<-st[2]; st2<-st[3]
    starting.pareto<-c(st1,st2)
    res<-optim(starting.pareto,lropareto,x=x,method="Nelder-Mead",addsign=T)

    if(res$val==1e+10){
      warning("Loglikelihood Value is Inf")
      res$val<-Inf
    }
    AIC<--2*(-res$val)+2*2; r <- c(AIC,res$par)
  }
  names(r)<-c("AIC","location_rnd","shape_rnd")
  return(r)
}
#Example
library(VGAM)
x<-round(rpareto(1000,3,1))
MLEroPareto(x)
    AIC    location_rnd  shape_rnd
 6331.594    3.006       1.022
```

5.2.4. Rounded Gamma Distribution

The Gamma distribution is one of the most important continuous distributions. It has shape parameter α and scale parameter β. The Gamma distribution has been applied to many fields, including waiting time in queueing systems, the lifetime of devices as in reliability theory, the load on web services etc [38]. This is the third continuous distribution that we select to define a PMF using the CDF of it. The name of our new discrete distribution is the rounded Gamma distribution.

We assume that $X = (X_1, X_2, ..., X_n)$ is an iid sample from a rounded Gamma distribution. The likelihood function therefore is:

$$L(\alpha, \beta|x) = \prod_{i=1}^{n} \{G((x_i + 0.5)|\alpha, \beta) - G((x_i - 0.5)|\alpha, \beta)\}.$$

As taking the logarithm of that equation, we obtain the log-likelihood function.

$$\log L(\alpha, \beta|x) = \sum_{i=1}^{n} \{\log(G((x_i + 0.5)|\alpha, \beta) - G((x_i - 0.5)|\alpha, \beta))\}. \tag{5.21}$$

5.2.4.1. R Implementation. In R, we define the function `MLEroGamma(x)` to calculate $\hat{\alpha}$ and $\hat{\beta}$ and the AIC. As internal functions, we code the PMF `drogamma(x, shape, scale)` and the log-likelihood function `lrogamma()` in Equation 5.21. $\hat{\alpha}$ and $\hat{\beta}$ are found numerically therefore the `lrogamma()` is optimized using the two dimensional optimization function `optim()` and the method "Nelder-Mead" in R. For the starting points, estimates calculated from the MLE function of Gamma distribution `MLEGamma(x)` are used. As a result, `MLEroGamma(x)` returns a vector holding the AIC and the parameter estimates. The R implementation of `MLEroGamma(x)`, `MLEGamma(x)` and also an example how to use it are given in the code below. In our example, we obtain a data set generated from the Gamma distribution and then we round it to get a discrete set.

Table 5.15. R Code for MLE-AIC of Rounded Gamma Distribution.

```
MLEGamma<-function(x){
  if(min(x)<0){
    warning("Negative Values or Zero")
    r<-c(NA,NA,NA)
  }
  else{
    n <- length(x)
    lprofilegamma <- function(alpha,x){
      res <- -n*(alpha)*log(sum(x))+n*(alpha)*log(n*alpha)+
        (alpha-1)*sum(log(x))-(n*alpha)-n*lgamma(alpha)
      if(is.infinite(-res)){res<--1e+11}
      return(res)
    }
    lwb<-0.1; upb<-lwb*100
    while(upb<1e+10){
      res<-optimize(lprofilegamma,x=x,c(lwb,upb),maximum=T)
      alpha<-res$maximum
      if(ceiling(res$objective)==-1e+11){
        res$objective<--Inf
        warning("Loglikelihood Value is Inf")
        break()
      }
      if(res$maximum<=(upb-0.1)){
        break()
      }
      if(ceiling(res$maximum)>=1e+10){
        warning("MLE found maximal r ")
        break()
      }
      lwb<-upb; upb<-upb*10
    }
    beta <- sum(x)/(n*alpha)
    AIC<--2*(res$objective)+2*2; r <- c(AIC,c(alpha,beta))
  }
  names(r) <- c("AIC","shape","scale")
  return (r)
}
MLEroGamma<-function(x){
  if(min(x)<0){
    warning("Negative Values or Zero")
    r<-c(NA, NA, NA)
  }
  else{
```

Table 5.16. R Code for MLE-AIC of Rounded Gamma Distribution (cont.).

```
    drogamma<-function(x,shape,rate=1,scale){
      x<-round(x)
      n<-length(x)
      res<- (pgamma((x+0.5),shape,rate,scale)
             -pgamma((x-0.5),shape,rate,scale))
      if(is.nan(sum(res))){
        res<-rep(NA,n)}
      return(res)
    }
    lrogamma<-function(theta,x,addsign=F){
      res<-sum(log(drogamma(x,shape=theta[1],rate,scale=theta[2])))
      if(is.infinite(-res)){
        res<--1e+11}
      if (addsign) res <- -res
      return (res)
    }
    st<-MLEGamma(x)
    alpha<-st[2]; beta<-st[3]
    starting.gamma<-c(alpha,beta)
    res <- optim(starting.gamma,lrogamma,x=x,method="Nelder-Mead",addsign=T)

    if(res$value==1e+11){
      res$value<-Inf
    }
    AIC<--2*(-res$value)+2*2; r <- c(AIC,c(res$par))
  }
  names(r) <- c("AIC","shape_rnd","scale_rnd")
  return (r)
}
#Example
x<-round(rgamma(1000,2)*4)
MLEroGamma(x)
   AIC   shape_rnd  scale_rnd
5916.7   2.038      3.887
```

6. GOODNESS OF FIT TESTS FOR DISCRETE DISTRIBUTIONS

In this section, we aim to make an assessment on the suitability of the best fitting distribution of our list. We ask whether the best fitting distribution really has a good fit for our data. Therefore, we apply the goodness of fit test for all discrete distributions in our list. The best fitting distribution is chosen as the proper model if the quality of its fit is good. Otherwise, the empirical distribution is chosen.

Chi-square, Anderson Darling (AD) and Kolmogorov-Smirnov (KS) tests are the most popular goodness of fit tests [20]. In our study, we apply the Chi-square test due to its good performance.

6.1. Basics on Goodness of Fit Tests

Goodness of fit describes the correspondence of a set of observations with the fitted statistical model. The null hypothesis H_0 assumes that our data come from a certain hypothesized distribution f_0 and therefore the model fits the data. The hypothesis H_1 assumes that our model does not fit the data.

We assume $X_1, ..., X_n$ is an iid random sample of size n from an unknown distribution f. The hypotheses are represented as:

$$H_0 : f = f_0 \quad vs. \quad H_1 : f \neq f_0.$$

There exist goodness of fit tests based upon either the PDF or the CDF of the distribution. Most of them consider continuous distributions as the hypothesized distributions. AD and KS tests are the most common methods that utilize the CDF and belong to the class of distance tests [39]. The chi-square test relies on the PDF. It is the best known and most widely used of the tests that may be used with discrete data [40]. Its

convenience of using with discrete data sets is one of our substantial reason to prefer the chi-square test.

The CDF approaches avoid data binning and are convenient for small samples. In fact, goodness of fit tests for smalls are harder to design and less versatile than for large sample sizes [41]. CDF approaches are also convenient for large samples. In their basic form, they are intended for continuous distributions, but they have also been adapted to discrete distributions [20].

There are many goodness of fit test methods which are suitable to use for discrete data including the multinomial test, the chi-square test, the two stage chi-square test and the discrete KS test [40]. We use the chi-square test as it is more convenient.

6.2. Chi-Square Test

The most distinct advantage of the chi-square test is that it can be used easily for any univariate discrete distribution [42]. Another significant feature is that the chi-square test is convenient for large samples. Using the chi-square test has also a disadvantage. The approximation it uses to calculate the p-value is sensitive to the number of expected frequencies. Therefore we develop a procedure to collect integers with small expected frequencies in a group, thus constructing a group with a higher expected frequency.

6.2.1. The Procedure of Chi-Square Testing for Discrete Distributions

When we fit a distribution to our observed data, we wonder how well the distribution actually reflects the data. The chi-square test compares the observed values and the expected values under the fitted model and statistically computes how close they are. If the test statistic is small, the difference is also small therefore the fit is good. Now we will explain the steps of the chi-square test procedure below.

(i) χ^2, the chi-square test statistic is found by summing the difference between the observed and expected frequencies for each class divided by the expected frequencies for the class. It measures how the pattern of observed frequencies differs from the pattern of expected frequencies. The χ^2 formula is given by Equation 6.1 below.

$$\chi^2 = \sum_{i=1}^{k} \frac{(O_i - E_i)^2}{E_i} \approx {\chi_0}^2{}_{k-m-1}, \tag{6.1}$$

where O_i denotes the observed frequency and E_i the expected frequency for class i, m is the number of parameters of the null hypothesis and k is the number of classes. χ^2 follows approximately the Chi-square distribution with $(k - m - 1)$ degrees of freedom. Probabilities for the test statistic are obtained from the Chi-square probability distribution. Hence, we can test hypotheses. If the values of observed and expected frequencies are clearly different, the hypothesis H_0 is rejected. To find that out, the p-value is calculated using the CDF of the Chi-square distribution with $(k - m - 1)$ degrees of freedom and compared to the significance level α.

(ii) The null hypothesis is accepted when the p-value is larger than the significance level α that is often chosen to be 0.01, 0.05, or 0.10. It is also possible to choose any value between 0 and 0.1. For the explanations and the formula of the significance level α we used, see Section 8.4.4.

(iii) The chi-square test can be applied to binned data which are arranged by classes. Therefore, before the test, a histogram or frequency table should be constructed by dividing the data into bins or classes. As we consider discrete data sets, often each different observation can be a class. See Section 6.3.1 for the implementation of the frequency table.

(iv) For the chi-square approximation to be valid, the expected frequency should be at least five since the calculated distribution of χ^2 is not very closely realized for very small classes [43]. Therefore, if some of the counts are less than five, it is required to combine some bins in the tails [42]. We have to decide on how to

define the intervals before the chi-square test. In general we have to think about the groups. See Section 6.3.2 for grouping.

(v) Another important consideration is MLE for the chi-square test, as fixed parameter values are used to calculate the PMF values and finally the expected frequencies. We explain the calculation of observed and expected frequencies below.

(vi) Observed frequencies are calculated by finding the number of observations in each class.

(vii) According to the distribution indicated by the null hypothesis, the PMF values and then the expected frequencies are calculated for the data set. It is computed by:

$$E_i = np_i,$$

where n is the sample size and p_i is the PMF value of the null hypothesis at MLEs for class i. As p_i is calculated using the MLEs, we know that the null hypothesis includes a fully specified distribution.

6.3. R Implementation

We code three functions including the main chi-square test function `CSTestDiscrete`() and the subsidiary data rearrangement functions `ObsFreq`(`x`) and `Grouping`(). We first explain the preparing functions as the main chi-square test function calls them. The flow of computation in the functions and examples are given below.

6.3.1. Function `ObsFreq`(`x`)

We require binned data to perform the chi-square test. Therefore, to organize and summarize the data, `ObsFreq`(`x`) constructs the frequency table. It gets the data vector as input and returns a table with two rows holding the different observed values

and their observed frequencies. The values in the first row determine the intervals or classes. The number of classes is specified by the number of different observations. We use the second row for the observed frequencies which are determined by counting the number of times a different observation occurs. As the data set we use is a particular sample, some values can be not observed. Therefore, we designate the missing data values between the maximum and the minimum observed values and add them to the first column. Then, we assign zero for the observed frequencies of these values. Observed frequencies are the source to compute the expected frequencies. Therefore `ObsFreq(x)` will be called in the main function.

Table 6.1. R Code for Observed Frequency Table.

```
ObsFreq<-function(x){
  num.indice<-length(table(x))
  Tobs<-array(NA,num.indice)
  Tobs.freq<-array(NA,num.indice)
  Tobs<-as.numeric(names(table(x)))
  Tobs.freq<-table(x)
  upper<-max(Tobs)
  Tmain<-matrix(NA,nrow=2, ncol=(upper+1), byrow=TRUE)
  rownames(Tmain)<-c("Observed","ObsFrequency")
  Tmain[1,]<-(0:upper); Tmain[2,]<-0
  for(j in 1:(upper+1)){
    for(i in 1:num.indice){
      if(Tmain[1,j]==Tobs[i]){
        Tmain[2,j]<-Tobs.freq[i]}
    }
  }
  return(Tmain)
}
#Example
x<-rbinom(1000,10,0.5)
ObsFreq(x)

             [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11]
Observed        0    1    2    3    4    5    6    7    8     9    10
ObsFrequency    1   10   43  104  219  258  190  122   41     7     5
```

6.3.2. Function `Grouping(T.group, minexpect, x)`

The chi-square test is sensitive to small expected frequencies. When the expected frequencies are smaller than five, the chi-square distribution gives misleading probabilities because of bad approximation. Therefore, we created the function `Grouping()` to have expected frequencies with acceptable values. Although there is not a specific rule for the minimum acceptable expected value for a class, we should make an assumption that no class has an expected frequency less than 5. Different common rules are: 80 percent of the expected frequencies should exceed five, they should be at least 10 and should be at least 20 [44]. We use the variable `minexpect` to let the user choose that value. `Grouping()` requires the following inputs respectively: `T.group`, the observed and expected frequency table to be grouped, `minexpect`, the minimum allowed expected frequency value and `x`, the data vector. It collects the small frequency values in the tails and forms new classes. Thus, it returns a grouped table and the expected frequencies in the table become acceptable for each class. We explain the procedure of grouping in detail below.

Grouping basically combines the classes in the tails which have small expected frequencies. Therefore, the algorithm first designates the points where grouping will start. From the rightmost and leftmost to the center of the distribution, the very first expected frequency values larger than `minexpect` are chosen as the starting points indicated by "`ie`". After finding them, right tail grouping is done from that point to the rightmost and then left tail grouping is done to the leftmost. Here we only explain the right tail grouping as the procedure is similar and we apply the same idea also for the left tail. For the new groups, we used a dynamic indicator "`mi`" which shows the index number. For the first group the index number is "`ie + 1`". As new groups are formed, the index number is increased by 1.

For the first group, expected frequency values starting from "`ie + 1`" are summed until the summation reaches at least `minexpect`. Then it is assigned to the class corresponding to the index "`mi`". Other groups are constructed in the same way.

Observed frequencies are also arranged according to the expected frequencies. The algorithm assigns "NA"s for the emptied cells after summation. Due to the fact that grouping starts from center to the tails, sometimes one single expected frequency value may exist in the last class. In that case, the algorithm checks whether it is smaller than `minexpect`. If so, the value is added to the last group. As the last step, all "NA"s appearing in the tails are removed from the table. Eventually, we have a table with reasonable observed and expected frequencies which is useful for chi-square testing.

Table 6.2. R Code for Grouping.

```
Grouping<-function(T.group,minexpect,x){
  # start of grouping from the right tail
  m<-ncol(T.group)
  for(j in m:1){
    if(T.group[2,j]>=minexpect){
      ie<-j
      break()}
  }
  if(T.group[2,m]<minexpect){
    sum1<-0; sum2<-0; mi<-ie+1
    for(j in (ie+1):m){
      sum1<-sum1+T.group[1,j]; sum2<-sum2+T.group[2,j]
      T.group[1,j]<-NA; T.group[2,j]<-NA
      if (sum2 >=minexpect){
        T.group[2,mi]<-sum2; T.group[1,mi]<-sum1
        mi<-mi+1; sum1<-0; sum2<-0 }
    }
    T.group[2,mi-1]<-sum2+T.group[2,mi-1]
    T.group[1,mi-1]<-sum1+T.group[1,mi-1]
  }
  # start of grouping from the left tail
  m<-ncol(T.group)
  for(j in 1:m){
    if(T.group[2,j]>=minexpect){
      ie<-j
      break()}
  }
  if(T.group[2,1]<minexpect){
    sum1<-0; sum2<-0; mi<-ie-1
    for(j in (ie-1):1){
```

Table 6.3. R Code for Grouping (cont.).

```
      sum1<-sum1+T.group[1,j]
      sum2<-sum2+T.group[2,j]
      T.group[1,j]<-NA; T.group[2,j]<-NA
      if (sum2 >=minexpect) {
        T.group[2,mi]<-sum2; T.group[1,mi]<-sum1
        mi<-mi-1; sum1<-0; sum2<-0}
    }
    T.group[2,mi+1]<-sum2+T.group[2,mi+1]
    T.group[1,mi+1]<-sum1+T.group[1,mi+1]
  }
  print("Grouped Table with NA s")
  print(T.group)
  print("Grouped Table omiting NA s")
  T.group.omit<-T.group[,apply(T.group, 2, function(x) !any(is.na(x)))]
  print(T.group.omit)
  #Omitting the NAs
  return(T.group.omit)
}
#Example
x<-rbinom(1000,15,0.5)
Grouping(T.group,minexpect=10,x)

 [1] "RoundedNormal Distribution"
[1] "Ungrouped Table"
         [,1]  [,2]  [,3]   [,4]   [,5]  [,6]    [,7]    [,8]    [,9]   [,10]
ObsFreq 0.000 0.000 3.000 11.000 36.000 89.00 164.000 183.000 206.000 169.000
ExpFreq 0.079 0.555 2.972 12.045 36.963 85.92 151.301 201.862 204.058 156.294
         [,11]  [,12]  [,13] [,14]
ObsFreq 78.000 38.000 20.000 3.000
ExpFreq 90.698 39.873 13.278 3.348
[1] "Grouped Table with NA s"
        [,1] [,2] [,3]   [,4]   [,5]  [,6]    [,7]    [,8]    [,9]   [,10]
ObsFreq   NA   NA   NA 14.000 36.000 89.00 164.000 183.000 206.000 169.000
ExpFreq   NA   NA   NA 15.651 36.963 85.92 151.301 201.862 204.058 156.294
         [,11]  [,12]  [,13] [,14]
ObsFreq 78.000 38.000 23.000    NA
ExpFreq 90.698 39.873 16.626    NA
[1] "Grouped Table omitting NA s"
          [,1]   [,2]  [,3]    [,4]    [,5]    [,6]    [,7]   [,8]   [,9]
ObsFreq 14.000 36.000 89.00 164.000 183.000 206.000 169.000 78.000 38.000
ExpFreq 15.651 36.963 85.92 151.301 201.862 204.058 156.294 90.698 39.873
         [,10]
ObsFreq 23.000
ExpFreq 16.626
```

6.3.3. Function `CSTestDiscrete(x, minexpect, pmf, distname...)`

`CSTestDiscrete()` is created to have a general chi-square testing function for the discrete distributions of our pool. As the input variables, `x` represents the data vector, `minexpect` indicates the minimum allowed value for the expected frequencies (we used 10 as default in our implementation). The third variable `pmf` is the probability mass function defined in R such as `dunif()`, `dpois()`. `distname` holds the name of the hypothesized distribution for the warnings. The other variable after `distname` is represented by "(...)" and it captures the list of parameter estimates of the hypothesized distribution. `CSTestDiscrete()` returns a vector holding the chi-square test statistic, p-value, degrees of freedom (dof) and a case number that we will explain below. `CSTestDiscrete()` is called from our main input analyzing function `FitAllDiscrete(x)`. It is coded externally hence the usage is possible if the test for a single discrete distribution is required.

Before explaining the procedure, we give the four cases for which the chi-square test can be not available. The first case occurs when the parameter estimates are not available. Then the chi-square test statistics can not be calculated. As second case, we consider whether the data are discrete. If there are many different observations with very small frequencies, the data set resembles a continuous set. We can not perform the chi-square test with very small expected frequencies. Therefore, if the maximum of the expected or the observed frequencies is smaller than `minexpect` we terminate the procedure without a result. The third case occurs when there is only one group hence one expected frequency value. In this case, the matrix form of the table turns into an array in R therefore it is not proper to use it for the chi-square test. The fourth case occurs when the grouping function results in so few groups that the dof is found to be zero or negative. In the code below, we enumerate the problems to specify them and give the case numbers in the returned vector. We assign zero for the case that the chi-square test is executed successfully.

When the chi-square test is possible, first `ObsFreq(x)` is called and the observed

frequencies are obtained. Then, the probabilities for each different observation are found using PMF and then the expected frequencies are calculated. As a result, we obtain a table of observed and expected frequencies. Then the function `Grouping`() is called and the table is grouped based on the expected frequencies. After that, using the built in functions `chisq.test` and $1 - $ `pchisq` of R, `CSTestDiscrete`() calculates and holds the chi-square test statistic, the complementary p-value, dof and the case number in the returned vector. It is used in the main input analyzing function to decide about acceptance or rejection of the hypothesized distribution.

Table 6.4. R Code for Chi-Square Test.

```
CSTestDiscrete<-function(x,minexpect, pmf,distname,...){
  print(paste(distname,"Distribution"))
  T.main<-ObsFreq(x)
  n<-length(x)
  ncolT<-ncol(T.main)
  probmass<-pmf(T.main[1,],...)

  if(is.na(any(probmass))){
    probmass<-pmf(c(NA,(T.main[1,2:ncolT])),...)
    probmass[1]<-0
  }
  #(logarithmic and pareto distributions find Inf for zero (x>0))
  if(is.infinite(probmass[1]) || is.nan(probmass[1])){
    probmass[1]<-0}

  ExpFreq<-probmass*n
  Tbl<-matrix(NA,nrow=2,ncol=ncolT,byrow=TRUE)
  #Observed Frequency and Expected Frequency Table
  rownames(Tbl)<-c("ObsFreq","ExpFreq")
  Tbl[1,]<-T.main[2,]; Tbl[2,]<-ExpFreq
  print("Ungrouped Table")
  print(Tbl)

  #Decision for Starting the Chi-Square Test
  #Case 1.Check the parameter estimates
  if(all(is.na(Tbl[2,]))){
    warning("can not find parameter")
    warning("Chi-Square Testing is not available due to NAs")
    res<-c(NA,NA,NA,1)}
```

Table 6.5. R Code for Chi-Square Test (cont.).

```
else{
    #Case 2.Check the number of integers
    maxexp<-max(Tbl[2,])
    if((max(table(x)/n)<(minexpect/n)) || (maxexp<minexpect)){
      warning("not discrete there is continuity")
      warning("Grouping and Chi-Square Testing is not available for ",distname,
              " Distribution because max(exp)is too small")
      res<-c(NA,NA,NA,2)}
    else{
      #grouping is necessary
      T.grouped<-Grouping(Tbl,minexpect,x)
      Tbl.csq<-T.grouped

      #Case 3. only one group (matrix turns into an array)
      if(Tbl.csq[1]==n){
        warning("Chi Square is not available because only one
                                exp freq is available")
        res<-c(NA,NA,NA,3)}
      else{
        npar<-length(c(...))
        dof<-(ncol(Tbl.csq)-npar-1)
        #Case 4. dof problem
        if(dof<=0){
        warning("dof problem")
          res<-c(NA,NA,NA,4)
        }
        else{
          #Case 0. chi-square is possible
          ChiSq<-chisq.test(x=Tbl.csq[1,],p=(Tbl.csq[2,]/n),rescale.p=T)$statistic
          PValue<-1-pchisq(ChiSq,dof)
          res<-c(ChiSq,PValue,dof,0)
        }}}}
  names(res)<-c("ChiSq","PValue","dof","case")
  return(res)
}
#Example
x<-rpois(1000,1.6)
CSTestDiscrete(x,minexpect=10,pmf=dpois,distname="Poisson",1.6)
     ChiSq     PValue        dof       case
5.4151e+00 2.4729e-01 4.0000e+00 0.0000e+00
```

7. EMPIRICAL DISTRIBUTION

If none of the parametric distributions has a good fit for the data, the empirical distribution can be easily used to estimate the underlying PMF. Let $X = (X_1, X_2, ..., X_n)$ be an iid sample with PMF $\hat{f}(x)$. Then the empirical PMF is defined as:

$$P(X = x) = \hat{f}(x) = \frac{\#\ of\ x}{n},$$

where "$\#\ of\ x$" denotes the frequency for the observed integer x and n corresponds to the sample size of the data set. The empirical PMF $\hat{f}(x)$ converges to the true PMF $f(x)$ as the sample size goes to infinity according to the strong law of large numbers. If the sample size increases, the error coming from using the empirical PMF gets smaller hence the estimator gets closer to the correct distribution with larger probability [45].

$$\hat{f}_n(x) \to f(x) \ \ as \ \ n \to \infty. \tag{7.1}$$

7.1. Advantages and Disadvantages of Using the Empirical Distribution

For estimating the PMF, using the empirical distribution seems very sensible since it only uses the data. However, it should be preferred only if we can not describe our data well using a parametric distribution. In this section, we discuss the advantages and the disadvantages of using the empirical PMF or parametric distributions. Then, we explain the conditions that require using the empirical distribution for a better representation of the data.

7.1.1. Advantages of Empirical PMF

The parametric approach requires knowledge in order to make hypothesis testing for the distributions. However, the empirical distribution is constructed easily without assumptions about the underlying distribution. Therefore it is easier to understand than the parametric approach. It is also practical that the estimation is done directly from the data. The empirical distribution approximates the true PMF well as the sample size gets larger as indicated in formula given by Equation 7.1. Therefore, for large samples, the empirical distribution may allow a better representation of the data. Another important property is the extreme flexibility to represent the data. That is, the empirical PMF works for all kind of discrete data. Besides, when the data set contains only a small number of different integers, the empirical distribution can estimate the PMF very well.

7.1.2. Disadvantages of Empirical PMF

The empirical PMF is not a smooth distribution, especially in the tails [8]. This problem arises from using a particular sample of data to represent the parent population. When the sample size is small, often some integers are not observed in the sample. Hence the PMF we get has gaps. As a result, extreme conditions with small probabilities can not be represented well using the empirical distribution. Besides, if the data have dependency or non-stationary behaviour, then it is not possible to get samples naively, therefore the empirical distribution can not be used. As a consequence, it is suitable to use the empirical distribution when:

- the sample size is very large
- the sample is not very small and contains only a small number of different integers

7.2. R Implementation of Empirical PMF

The decision for choosing either a parametric or the empirical distribution is done in our main input analyzer `FitAllDiscrete(x)`. If none of the distributions of our list has an acceptable fit for our data, the nonparametric approach is chosen. Hence, the algorithm calculates the relative frequencies according to the data. After that, it is suggested to use the empirical distribution. The summary table for the parametric distributions and estimated parameter values are also displayed to show that none of the distributions of our list has an acceptable fit. The code part and an example are presented below. In this example, we generate a data set with sample size 1000 having only a small number of different integers. We can suppose that it presents the number of people who get on a taxi. It is seen that our tool recommends the empirical distribution, as no distribution of our list had a good fit.

Table 7.1. R Code for Empirical Distribution.

```
#Non-Parametric Model
    rf<-table(x)/n
    print("Recommend")
    print("Nonparametric Model")
    recommend<-("It is recommended to use the Empirical Distribution.
                  Relative Frequencies are calculated for all observed values")
    res<-list("model"=recommend, "summary"=ORD.nopar, "RelFreq"=rf)
    return(res)
#Example
u<-runif(1000)
y<-(u>0.5)+(u>0.6)+(u>0.7)
options(scipen=-20,digits=5)
FitAllDiscrete(y)
[1] "Nonparametric Model"
$model
[1] "It is recommended to use the Empirical Distribution.
Relative Frequencies are calculated for all observed values"
```

Table 7.2. R Code for Empirical Distribution (cont.).

```
$summary
                   AICvalue parameter1 parameter2  PVal   Decision  Delta AIC
Uniform          2.7766e+03 0.0000e+00 3.0000e+00 0e+00     reject 0.0000e+00
Geometric        2.9932e+03 4.6147e-01          NA 0e+00     reject 2.1663e+02
NegativeBinomial 2.9952e+03 1.0000e+00 4.6147e-01 0e+00     reject 2.1863e+02
RoundedGamma     3.0688e+03 5.2579e-01 2.3222e+00 0e+00     reject 2.9222e+02
Poisson          3.1423e+03 1.1670e+00          NA 0e+00     reject 3.6572e+02
Binomial         3.1443e+03 2.9998e+05 3.8903e-06 0e+00     reject 3.6772e+02
RoundedNormal    3.3891e+03 1.1671e+00 1.2819e+00 0e+00     reject 6.1250e+02
Logarithmic             NA          NA          NA    NA not avail.         NA
RoundedPareto           NA          NA          NA    NA not avail.         NA

$RelFreq
x
       0        1        2        3
5.05e-01 1.13e-01 9.20e-02 2.90e-01
```

8. THE NEW INPUT ANALYZER FITALLDISCRETE

In this part, we deal with distribution selection which is the procedure of selecting the most convenient probability distribution for a given data set. It aims to find the distribution that best fits to our data. The procedure of finding the most proper distribution will be explained and the input modeling function `FitAllDiscrete(x)` will be introduced which is implemented in R. We start this chapter with explaining the requirements for the input data.

8.1. Requirements for the Input Data

In the context of discrete input modeling, we only consider discrete data as input. The user is expected to insert independent and identically distributed data coming from discrete distributions over the non-negative integers. Sample size is another important consideration for the input data. As the sample size changes, the power of the test for the hypothesized PMFs also changes [46]. Although there is not a specific rule for the minimum sample size a sample with a sufficiently large size is required [47]. Otherwise, if the sample size is very small, the chi-square test is unlikely to reject any candidate distribution [6].

8.2. The Procedure of Distribution Fitting and Selection of the Best Fitting Distribution

As the distribution selection criterion, we consider Akaike's information criterion (AIC) and the algorithm chooses the distribution which has the smallest AIC value as the best fitting one. After deciding on the best fit of our distribution list, it is checked whether the selected distribution is really appropriate for our data. The reason for the second confirmation is that we search in a pre-defined pool of distributions which is insufficient to represent some data sets. Therefore, the best fitting distribution of our list may be not a good fit. In order to check the goodness of fit, the chi-square

test is performed and the p-value of each hypothesized distribution is calculated. Then it is checked whether the p-value of the best fitting distribution is larger than the significance value α. If that is true, it is assumed that the data are presented quite well by this distribution. For this case, the name of the best fitting distribution, its estimated parameters and the AIC value are displayed. Otherwise, we know that the data are not presented well by any of our parametric distributions. Then the empirical distribution is recommended to the user. In this case, the recommendation for using the empirical distribution and the relative frequencies of the observed integers are displayed. Besides these two cases, there is the case that the chi-square test can not be performed for the data set. The chi-square test is not possible for our implementation when the maximum relative frequency is not large enough due to the large number of different integers observed in the data set. In this case, the user is warned and it is recommended to use a continuous distribution to model the data. Besides that recommendation, a summary table containing all fitted distributions is displayed.

8.3. Probability Distributions

For input modeling, we consider 9 discrete distributions, among them 6 standard and 3 rounded continuous distributions. Using the rounded continuous distributions enlarges our discrete distributions pool. In this way, we introduce new shapes to our pool that are not available in the standard distribution list. The list of our discrete distributions and the name of their parameters are given in Table 8.1.

Table 8.1. Discrete Distributions Included by `FitAllDiscrete(x)`.

Standard Discrete Distributions	**Rounded Continuous Distributions**
Poisson (lambda)	Rounded Normal (mean, sd)
Binomial (size, prob)	Rounded Gamma (shape, scale)
Neg. Binomial (size, prob)	Rounded Pareto (location, shape)
Geometric (prob)	
Disc. Uniform (min, max)	
Logarithmic (prob)	

8.4. Interpretation of the Summary Table

As the result of input analyzing, the user gets a recommendation and also a summary table for all distributions. In that table, all fitted distributions are sorted in ascending order of their AIC values. Therefore, the best fitting distribution with the smallest AIC value of the list is displayed in the first row. The summary table has six columns consisting of: AIC value, parameter1, parameter2, p-value, Decision and Delta AIC, the difference between the minimal AIC value and that of the considered distribution.

8.4.1. Akaikes Information Criterion Value

AIC is the main criterion we use for selecting the best fitting distribution (see Section 3.2 for details). It includes the negative value of the log-likelihood function at the MLE and adds a penalty to this value for the number of parameters in the model. Better fits correspond to smaller AIC values. As AIC is our selection criterion, the summary table is constructed with respect to it. The fitted distributions are sorted in ascending order of the AIC values automatically. The best fitting distribution, that with the smallest AIC value, is thus placed in the first row of the summary table.

8.4.2. Estimated Parameter Values

As we mentioned in 3.1, we use maximum likelihood estimation (MLE). By using this method, we find the parameter values which maximize the log-likelihood function of the fitted discrete probability distributions. If the distribution has a closed form solution for the MLEs (e.g. Poisson), we use it directly. Otherwise a numerical solution is required using optimization (see Chapter 4 for details). For one-dimensional optimization, we use `optimize`() with the method "Brent" and specify the borders of the parameter interval to be searched. For multi-dimensional optimization, we use `optim`() with the method "Nelder-Mead". As `optim`() requires feasible starting points for the parameters, we specify convenient initial values for them. MLEs are displayed

in the second row and the third row of the summary table named “parameter1” and “parameter2” respectively.

8.4.3. p-value of the Chi-Square Test and Its Interpretation

The p-value of the chi-square test is calculated and used to assess the quality of the fit of the chosen parametric distribution for the data set. In other words, the p-value is a criterion to verify the adequacy of the fit statistically. First, the chi-square test statistic is calculated using the observed and the expected frequencies. Then, the corresponding p-value is found using the test statistic. For good fits, we expect to get large p-values.

8.4.4. Chi-Square Test Decision

Test decisions are made for each hypothesized distribution according to our statistical assessment. The decisions are defined as “accept” for acceptable fits, “reject” for bad fits and “not available” for distributions where the chi-square test is not available. To make a decision, the algorithm compares the p-value of each distribution with the significance level. The significance level α is a threshold probability value to assess the p-value. If the p-value exceeds the significance level, then the null hypothesis (see Section 6.1) is accepted. Otherwise, it is rejected. Hence, choosing a reasonable significance level is important as it directly affects the decision results. The power of the chi-square test must be considered because it affects the decision. The power of the chi -square test is the ability to reject the null hypothesis and the power increases, as the sample size gets larger [46]. Together with the sample size, the number of different observed integers in the sample is also of importance as it influences the quality of the empirical distribution approach. Considering these two criteria, we use the formula given by Equation 8.1. It makes easier the acceptance of the null hypothesis for large sample sizes and large numbers of different integers.

$$a = \frac{5}{n} + \frac{0.1}{max(x) - min(x) + 1}. \tag{8.1}$$

8.4.5. ΔAIC Value

The ΔAIC value is the difference between the AIC value of the hypothesized distribution and the minimal AIC observed. ΔAIC values are displayed in ascending order starting from zero in the summary list. ΔAIC assessment only makes sense for the parametric case. Distributions with ΔAIC values smaller than 2 are also reasonable to choose as alternative. Values between 3 and 7 indicate that the model has considerably less support, whereas ΔAIC> 10 indicates that the model is very unlikely [27].

8.5. R Implementation of the New Input Analyzer

This part presents our main input analyzing function `FitAllDiscrete(x)` and its subsidiary functions `MLETableDiscrete(x)` and `parnames()`. First, we will explain the flow of computation in the functions and we will give examples to show how to use them.

8.5.1. Function `MLETableDiscrete(x)`

We first need the parameter estimates in order to perform the chi-square test and to calculate the AICs. As both MLE and AIC use the log-likelihood function, we calculate the AIC and MLEs together. Hence, function `MLETableDiscrete(x)` constructs a table containing the calculated MLEs-AICs and the number of parameter values for all fitted distributions. By using this function, we calculate the values once and store them in a table. Therefore code optimization is also realized by avoiding repetitions of those time consuming calculations.

8.5.2. Function `parnames()`

This function returns the parameter names for each discrete distribution. It is required when the algorithm forms the recommendation expressions for the distributions. It prepares a recommendation by specifying the name of the distribution and

Table 8.2. R Code for AIC-MLE Table.

```
MLETableDiscrete<-function(x){
  #  AIC-MLE Calculations
  d<-matrix(NA,nrow=9, ncol=4, byrow=TRUE)
  rownames(d)<-c("Uniform","Poisson","Geometric","Binomial",
                 "NegativeBinomial", "Logarithmic","RoundedNormal",
                  "RoundedGamma", "RoundedPareto")
  colnames(d)<-c("AICvalue","parameter1","parameter2","npar")

  d[1,1:3]<-MLEUnif(x);  d[2,1:2]<-MLEPois(x);  d[3,1:2]<-MLEGeom(x)
  d[4,1:3]<-MLEBinom(x); d[5,1:3]<-MLENBinom(x); d[6,1:2]<-MLELog(x)
  d[7,1:3]<-MLEroNorm(x); d[8,1:3]<-MLEroGamma(x);
  d[9,1:3]<-MLEroPareto(x)

  #Calculation for Number of Parameters
  for(i in 1:9){
    if(is.na(d[i,1]) || is.infinite(d[i,1])){d[i,4]<-NA}
    else{
      d[i,4]<-sum(!is.na(d[i,2:3]))
    }
  }
  return(d)
}
#Example
x<-rpois(1000,3.4)
MLETableDiscrete(x)
                   AICvalue parameter1 parameter2  npar
Uniform          4.7998e+03 0.0000e+00 1.0000e+01 2e+00
Poisson          4.0654e+03 3.3000e+00          NA 1e+00
Geometric        4.6662e+03 2.3256e-01          NA 1e+00
Binomial         4.0674e+03 1.0000e+05 3.3000e-05 2e+00
NegativeBinomial 4.0619e+03 3.0000e+01 9.0090e-01 2e+00
Logarithmic              NA          NA          NA    NA
RoundedNormal    4.1377e+03 3.3002e+00 1.8895e+00 2e+00
RoundedGamma     4.1197e+03 2.5016e+00 1.3210e+00 2e+00
RoundedPareto            NA          NA          NA    NA

Warning Messages:
1: In MLELog(x) : Negative Value or Zero
2: In MLEroPareto(x) : Negative Value or Zer0
```

the name of its parameters by calling `parnames()`.

Table 8.3. R Code for the Table of Parameter Names.

```
parnames<-function(){
  result<-matrix(NA,nrow=9,ncol=2)
  rownames(result)<-c("Disc. Unif","Poisson","Geometric","Binomial",
  "Neg.Binomial","Logarithmic","Rounded Normal","Rounded Gamma",
  "Rounded Pareto")
  colnames(result)<-c("pname1","pname2")
  result[1,1:2]<-c("lowerb","upperb") ; result[2,1]<-"lambda"
        result[3,1]<-"prob"; result[4,1:2]<-c("size","prob")
        result[5,1:2]<-c("size","prob"); result[6,1]<-"prob"
        result[7,1:2]<-c("mean","sd"); result[8,1:2]<-c("shape","scale")
        result[9,1:2]<-c("location","shape")
  return(result)
}
#Example
parnames()

               pname1     pname2
Disc. Unif     "lowerb"   "upperb"
Poisson        "lambda"   NA
Geometric      "prob"     NA
Binomial       "size"     "prob"
Neg.Binomial   "size"     "prob"
Logarithmic    "prob"     NA
Rounded Normal "mean"     "sd"
Rounded Gamma  "shape"    "scale"
Rounded Pareto "location" "shape"
}
```

8.5.3. Function `FitAllDiscrete(x)`

`FitAllDiscrete(x)` is our main input modeling function; it can be used easily. The user first must import the data set in R and then call the function. It returns a recommendation for the best fitting distribution and the summary table. When the user calls `FitAllDiscrete(x)`, it first checks whether all the values are integer in the data set. If the data contain non-integer values, the user gets a warning with the suggestion to use an input modeling tool for continuous distributions. Otherwise, input modeling is started. Non-negativity control is also important for the discrete data sets

and we control it within each MLE function. Only the rounded Normal distribution allows negative integer values, therefore we do not make the non-negativity control in the main function. Now, we explain each step of the algorithm, when the data are all integer and the input modeling procedure can be started.

If the data set is proper for discrete input modeling, `FitAllDiscrete(x)` first calls the function `MLETableDiscrete(x)` and gets the table of MLE and AIC values. Using the MLEs as input for the function `CSTestDiscrete()`, the chi-square test is performed for all nine discrete hypothesized distributions and all p-values are calculated. If the chi-square test is not possible, the p-value can not be calculated and the decision is set to "not available" for that distribution. If it is possible, p-values are compared to the significance level and the decision '"accept" or '"reject" is made for that distribution. If the chi-square test is not executed due to continuous nature of the data, the user is warned and recommended to use continuous distributions for modeling. Then the summary table is constructed with the following columns: AIC, parameter values, p-value, Decision and ΔAIC. Then, the table is sorted to get ascending AIC values. The distribution in the first row thus has the smallest AIC value and is considered the best fitting distribution of our list. If the best fitting distribution also passes the chi-square test, then it is chosen as the most proper distribution for the given data and recommended to the user. Otherwise, if it does not pass the chi-square test, the use of the empirical distribution is recommended. If the chi-square test can not be performed, it is recommended to use a continuous distribution to model the data. Thus, the algorithm may give three different recommendations. The R code and an example is given below. For more examples see Chapter 10.

Table 8.4. R Code for `FitAllDiscrete(x)`.

```
FitAllDiscrete<-function(x){

  #Integer Control for Data
  n<-length(x)
  if(is.integer(x[1]:x[n])==FALSE){
    res<-print("Given data set includes noninteger values.
Discrete data fitting is not possible. Try to use a continuous
input modeling tool.")
    return(res)
  }
  else{
    T<-MLETableDiscrete(x)
    nrowT<-nrow(T)
    n<-length(x)
    ncolT<-ncol(ObsFreq(x))

    #CS-PVal Calculation for All Distributions
    CS.PVal<-matrix(NA,nrow=nrowT,ncol=4)
    rownames(CS.PVal)<-rownames(T)
    colnames(CS.PVal)<-c("ChiSq","PValue","dof","case")
    FN<-c(dunif,dpois,dgeom,dbinom,dnbinom,dlog,dronorm,drogamma,
          dropareto)

    for(i in 1:nrowT){
      distname<-rownames(T[])[i]
      print(paste(distname, "Distribution"))
      #NA and Inf Control for AIC
      if(is.na(T[i,1]) || is.infinite(T[i,1])){
        warning("AIC is inf or NA")
        #Chi-Square testing is not called
        CS.PVal[i,]<-c(NA,NA,NA,5)
      }
      else{
        if(T[i,4]==1){ #one parameter
          param1=T[i,2]
          CS.PVal[i,]<-CSTestDiscrete(x=x,minexpect=10,pmf=FN[[i]],
                                      distname,param1)
        }
        else{#two parameters
          param1=T[i,2]; param2=T[i,3]
          CS.PVal[i,]<-CSTestDiscrete(x=x,minexpect=10,pmf=FN[[i]],
                                      distname,param1,param2)
        }}}
```

Table 8.5. R Code for `FitAllDiscrete(x)` (cont.).

```
PVal <-CS.PVal[,2];case<-CS.PVal[,4];AIC.par<-T[,1:3]
DeltaAIC<-array(NA,nrowT)
Decision<-array(NA,nrowT)
DF<- data.frame(AIC.par,PVal,Decision,DeltaAIC,parnames(),case)
# Ordering by AIC
ORD<-DF[order(DF[,1]),]

# Delta AIC
mnx<-ORD[1,1]; ORD[,6]<-(ORD[,1]-mnx)

#The Level of Significance
SignLevel<-(5/n)+(0.1)/(max(x)-min(x)+1)
#Decision
for(i in 1:nrowT){
  if(is.na(ORD[i,4])){
    Decision[i] <-"not avail."}
  else{
    if(ORD[i,4]>SignLevel){
      Decision[i] <-"accept"}
    else{Decision[i] <-"reject"}
  }
}
ORD[,5]<-Decision; ORD.nopar<-ORD[,1:6]

distribution.name <- rownames(ORD[])[1]
# Display AIC value:
AIC<-(ORD[1,1])
#parameters:
par<-ORD[1,2:3]
# Display parameters  by omitting "NA"s:
parameters<-(par[!is.na(par)])
names<-ORD[1,7:8]
par.names<-names[!is.na(names)]
full.definition.parameters<-paste(par.names,parameters,sep="=",
                                  collapse=",")
print("Recommend")
if(ORD[1,5]=="accept"){
  recommend<-paste("It is recommended to use the", distribution.name,
                           "distribution with the parameter(s):"
                   ,full.definition.parameters)
```

Table 8.6. R Code for `FitAllDiscrete(x)` (cont.).

```
res<-list("model"=recommend,"AIC"=AIC,"parameters"=parameters,
                "summary"=ORD.nopar)}
else{
      if(ORD[1,9]==2){
        recommend<-paste("The data set can be approximated by a continuous
          distribution. Due to the very large number of different observed integers,
          it is not possible to make chi-square test for discrete distributions.
          Try to use a continuous distribution to model this data. Best fitting
          discrete distribution is", distribution.name,"distribution with the
          parameter(s):",full.definition.parameters)
        res<-list("model"=recommend,"AIC"=AIC,"parameters"=parameters,
                  "summary"=ORD.nopar)}

      else{
        #Non-Parametric Model
        n<-length(x)
        rf<-table(x)/n

        print("Recommend")
        print("Nonparametric Model")
        recommend<-("It is recommended to use the Emprical Distribution.
        Relative Frequencies are calculated for all observed values")
        res<-list("model"=recommend, "summary"=ORD.nopar, "RelFreq"=rf)}
    }
    return(res)
  }
}
#Example
x<-rpois(1000,3.4)
options(scipen=-20,digits=5)
FitAllDiscrete(x)
[1] "Recommend"
$model
[1] "It is recommended to use the Poisson distribution with the parameter(s):
lambda=3.408e+00"
$AIC
[1] 4.0531e+03
$parameters
[1] 3.408e+00
```

Table 8.7. R Code for `FitAllDiscrete(x)` (cont.).

```
$summary
                   AICvalue parameter1 parameter2       PVal   Decision  Delta AIC
Poisson          4.0531e+03 3.4080e+00          NA 8.9734e-01     accept 0.0000e+00
NegativeBinomial 4.0533e+03 5.6000e+01 9.4263e-01 9.8892e-01     accept 2.3680e-01
Binomial         4.0551e+03 1.1000e+05 3.0982e-05 8.2556e-01     accept 2.0019e+00
RoundedGamma     4.1155e+03 2.7396e+00 1.2455e+00 2.9228e-12     reject 6.2395e+01
RoundedNormal    4.1271e+03 3.4080e+00 1.8797e+00 2.2306e-05     reject 7.4061e+01
Geometric        4.7226e+03 2.2686e-01          NA 0.0000e+00     reject 6.6949e+02
Uniform          4.9738e+03 0.0000e+00 1.1000e+01 0.0000e+00     reject 9.2073e+02
Logarithmic              NA         NA          NA         NA not avail.         NA
RoundedPareto            NA         NA          NA         NA not avail.         NA

Warning messages:
1: In MLEroPareto(x) : Negative Values or Zero
```

9. TESTING THE PERFORMANCE OF FITALLDISCRETE

In this chapter, we aim to evaluate the performance of our input analyzer on selecting the true distribution. To do this, we design an automatic test function called `TestFit`() which performs simulation studies based on our pool of discrete distributions. It first generates a number of random data sets from the distribution specified by the user. Then, it performs distribution fitting for each data set consecutively and forms a comprehensive list of results. The data sets are generated from the specified distributions with known parameter values. Thus we know the true underlying distribution and parameter values of the data sets. Hence, we can decide on the success of our input analyzer easily by comparing the suggested models with the true distributions.

One advantage of our test function is that it can easily generate data sets. It is not required for the use to code the random number generation procedure for the distributions as they are available in R. The user only gives the name of the distribution to be tested. Another advantage is that it is a practical method. Automatic testing allows to do many tests at a time which is impossible for manual tests. It is important to do many tests as the evaluation becomes more reliable. Our function also enables a simple usage. The user easily hands over the variables such as the number of different parameters to be used and the number of experiments for each parameter. The resulting tables are displayed together in a text file for each distribution. The name of the file is given automatically. The results are easily interpreted since the necessary information for the related data set is displayed before the tables.

Our function must not be used when the generated data set is not discrete. In order to perform the chi-square test for discrete distributions with our code, the expected frequencies must be large enough as explained in Section 6.2.1. Therefore choosing the intervals for the parameters is an important step for testing input modeling for the discrete distributions.

We first explain our automatic testing algorithm and how we use it. Then, we make a test for the Poisson distribution to demonstrate the use of our test function. After that, to evaluate the success of our input analyzer, we make the test for all discrete distributions. We summarize the results in Table 9.7. We mainly consider three criteria: ΔAIC, p-value and the number of accepted tests to evaluate the success in finding the true distribution.

9.1. R Implementation for the Function `TestFit()`

Our automatic test function `TestFit()` has six input variables: `n`, `rgf`, `numexp`, `numrandpar`, `parlist` and `distname`. First, the user decides on the sample size `n` for the data to be generated. She should choose a reasonable sample size (see Section 8.1) for a reliable test. `rgf` corresponds to the random generation function of R. For instance, the user must enter *rpois* for the Poisson distribution test and *rbinom* for the Binomial distribution test. The name of the hypothesized distribution is defined using the variable `distname`. For the possible values of `rgf` and `distname`, see Table 9.1. For random sampling, the algorithm also requires the parameter values. The variable `parlist` is used to hold information about the parameter values. It holds a list consisting of the user defined intervals of all parameter values and the data types (real or integer) of them. This list is defined as explained in Section 9.2 below. The fourth variable `numrandpar` determines the number of data sets generated for each parameter set. The fifth variable `numexp` specifies the number of repetitions for each parameter to generate data sets. Hence, using the information in `parlist`, the algorithm generates `numrandpar` many different parameter sets and performs input modeling `numexp` times for each parameter set. The algorithm holds the randomly generated data sets in a three dimensional array of dimension (`numexp`$\times$`n`$\times$`numrandpar`). As a result, by calling the function `TestFit()`, `FitAllDiscrete(x)` is called (`numrandpar` $\times$ `numexp`) times. The resulting tables are saved in a text file. The name of the text file is determined automatically using the name of the distribution and its sample size. Hence, at the end of the test, the user has different text files with unique names. The R code of the function and an example how to use it are given in the code below. Before presenting

the code, we explain how to prepare the `parlist`.

Table 9.1. Random Generation Functions and Number of Parameters for Each Distribution.

Random Generation Functions and Number of Parameters of Each Distribution		
Name of Distribution	**Random Generation Function**	**Number of Parameters**
Poisson	`rpois`	1
Binomial	`rbinom`	2
NegativeBinomial	`rnbinom`	2
Geometric	`rgeom`	1
Uniform	`runif`	2
Logarithmic	`rlog`	1
RoundedNormal	`rnorm`	2
RoundedGamma	`rgamma`	2
RoundedPareto	`rpareto`	2

9.2. Preparing `parlist`

Five of the variables of our automatic testing function `FitTest`() require single value as input. However, `parlist` holds a list of information for the parameter values and the data types. Therefore, the user must first prepare a list. For one parameter distributions, `parlist1` and for two parameter distributions `parlist2` must be used. After deciding on the list format, the user hands over the minimum and maximum values of the parameters.

Then the data types must be taken into account for each parameter. If the parameter is integer, the value of variable "by" must be equal to 1 to generate integer numbers, otherwise it can be set to 0.02. `reint` holds the information on the data type (T for real and F for integer) and in the algorithm we use the length of it to find the number of parameters. The mentioned listing formats and the codes for `FitTest`() are given as below.

(i)
```
parlist1<-list(par1=seq(from=val,to=val,by=1/0.02,reint=T/F))
```

(ii)
```
parlist2<-list(par1=seq(from=val,to=val,by=1/0.02),
                par2=seq(from=val,to=val,by=1/0.02),reint=c(T/F,T/F)))
```

Table 9.2. R Code for Automatic Test.

```
TestFit<-function(n,rgf,numexp,numrandpar,parlist,distname){
  res<-array(dim=c(nrow=numexp,ncol=n,numrandpar))
  npar<-length(parlist$reint)
  var_filename<-as.name(distname)
  sink(paste(var_filename,n,".txt"),append=TRUE)
  p1<-sample(parlist$par1,numrandpar,replace=TRUE)
  print("==================================================")
  print(paste(distname,"Distribution Experiments"))
  print("--------------------------------------------------")
  print("Chosen randomly generated parameters (par1)")
  print(p1)
  if(npar==2){
    p2<-sample(parlist$par2,numrandpar,replace=TRUE)
    print("Chosen randomly generated parameters (par2)")
    print("==================================================")
    print(p2)}
  for(k in 1:numrandpar){
    for(i in 1:numexp){
      if(npar==2){
        if(distname=="RoundedNormal"){
          res[i,,k]<-round(rgf(n)*p1[k]+p2[k])}
        else{
          if(distname=="RoundedGamma"){
            res[i,,k]<-round(rgf(n,p1[k])*p2[k])}
          else{
            if(distname=="Uniform"){
              res[i,,k]<-floor(rgf(n)*(p2[k]-p1[k]+1)+p1[k])}
            else{
              if(distname=="Binomial" || distname=="NegativeBinomial"){
                res[i,,k]<-rgf(n,p1[k],p2[k])}
              else{
                if(distname=="RoundedPareto"){
                  res[i,,k]<-round(rgf(n,p1[k],p2[k]))}
              }}}}}
      else{# for geom, log, poisson
        res[i,,k]<-rgf(n,p1[k])}
    }}
  for(k in 1:numrandpar){
    print("==================================================================")
    if(npar==1){
      print(paste("Num of RandomParameter:",k,"Parameter:",p1[k],
                                      "sample size:",n))}
```

Table 9.3. R Code for Automatic Test (cont.).

```
else{
     print(paste("Num of RandomParameter:",k,"Parameter1:",p1[k],
                                    "Parameter2:",p2[k], "sample size:",n))}
     print("================================================================")
    for(i in 1:numexp){
      print(paste("Number of experiment:",i))
      print("==================================")
      x<-res[i,,k]
      print(FitAllDiscrete(x))
    }}}
```

9.3. Example: A Single Automatic Test for the Poisson Distribution

In order to perform a test for the function `FitAllDiscrete(x)` using samples of the Poisson distribution, we first make a list using the format *parlist*1 as the Poisson distribution has one parameter. The parameter list we prepare for the test is given below. Here, real parameter values between 0.1 and 50 will be assigned to the parpois list as parameter λ gets real values.

```
parpois<-list(par1=seq(from=0.1,to=50,by=0.02),reint=T)
```

After defining the parlist, we easily assign the other input values and can use for example the following commands.

```
parpois<-list(par1=seq(from=0.1,to=50,by=0.02),reint=T)
TestFit(n=250,rgf=rpois,numexp=3,numrandpar=5,parlist=parpois,distname="Poisson")
```

According to this example, `TestFit()` performs 15 tests and writes all summary tables in a text file called "$Poisson2.5e+02.txt$" which contains 15 summary tables. The results and the explanations are given in the following section.

9.3.1. Results of the Automatic Test for the Poisson Distribution

As the result of the test, first the file is formed called "$Poisson2.5e+0.2.txt$". Then 5 different parameter values are obtained and printed in the text file as seen below. After that, for each parameter value, three different data sets are generated. Then input modeling is initialized for these data sets and the tables are displayed with the label of the experiment number. Hence, `FitAllDiscrete(x)` is called 15 times automatically and in this way, we have 15 summary tables representing the experiments. The file is listed below.

Table 9.4. Results of the Automatic Test for the Poisson Distribution.

```
"Poisson2.5e+0.2.txt"
"================================================="
[1] "Poisson Distribution Experiments"
[1] "-------------------------------------------------"
[1] "Chosen randomly generated parameters (par1)"
[1] 5.10e+00 1.21e+01 1.20e+01 2.69e+01 1.10e+01
[1] "================================================================"
[1] "Num of RandomParameter: 1 Parameter: 5.1e+00 sample size: 2.5e+02"
[1] "================================================================"7
[1] "Number of experiment: 1"
[1] "================================="
[1] "Recommend"
$model
[1] "It is recommended to use the Poisson distribution with the parameter(s):
 lambda=5.092e+00"
$AIC
[1] 1.136e+03
$parameters
[1] 5.092e+00
$summary
                  AICvalue parameter1 parameter2       PVal Decision  Delta AIC
Poisson          1.1360e+03 5.0920e+00         NA 6.3223e-01   accept 0.0000e+00
NegativeBinomial 1.1374e+03 6.5000e+01 9.2735e-01 6.0351e-01   accept 1.3417e+00
RoundedNormal    1.1374e+03 5.0916e+00 2.3169e+00 8.8078e-01   accept 1.4070e+00
Binomial         1.1380e+03 1.1000e+05 4.6291e-05 5.0291e-01   accept 2.0008e+00
RoundedGamma     1.1625e+03 3.7547e+00 1.3569e+00 1.1274e-01   accept 2.6436e+01
```

Table 9.5. Results of the Automatic Test for the Poisson Distribution (cont.).

```
Uniform          1.2465e+03 0.0000e+00 1.1000e+01 0.0000e+00   reject 1.1042e+02
Geometric        1.3620e+03 1.6415e-01          NA 0.0000e+00   reject 2.2597e+02
Logarithmic              NA          NA          NA          NA   not avail      NA
RoundedPareto            NA          NA          NA          NA   not avail      NA
Warning messages:
1: In MLELog(x) : Negative Value or Zero
3: In MLEroPareto(x) : Negative Values or Zero
[1] "Number of experiment: 2"
[1] "==================================="
...
[1] "Number of experiment: 3"
[1] "==================================="
...
[1] "============================================================="
[1] "Num of RandomParameter: 2 Parameter: 1.21e+01 sample size: 2.5e+02"
[1] "============================================================="
[1] "Number of experiment: 1"
[1] "==================================="
 ...
[1] "Number of experiment: 2"
[1] "==================================="
...
```

9.4. Example: Automatic Test for All Discrete Distributions

After performing the single automatic test for the Poisson distribution, we enlarge it to all discrete distributions using two different sample size values. We carry out simulation for the distributions of our list using 50 different parameter sets (`numrandpar` = `50`) with no repetition (`numexp` = `1`) using the sample size values 500 and 1000 (n=500, n=1000). For the parameter lists, we use the interval values given in Table 9.6.

Table 9.6. INTERVAL TABLE.

INTERVAL TABLE			
Distribution	**Par1 Interval**	**Par2 Interval**	**Real=T, Int=F**
Poisson	lambda=0.1-50	-	T
Binomial	nb=5-100	p=0.1-0.9	F-T
Negative Binomial	r=5-30	p=0.3-0.9	F-T
Geometric	p=0.1-0.9	-	T
Discrete Uniform	min=5-40	max=50-60	F-F
Logarithmic	p=0.3-0.9	-	T
Rounded Normal	sigma=2-10	mu=5-20	T-T
Rounded Gamma	a=1-10	b=1-5	T-T
Rounded Pareto	a=5-10	k=5-10	T-T

As first criterion, we consider the case with zero ΔAIC. Therefore, we count the number of cases that the true distribution is recommended. As the second criterion, we consider the case with ΔAIC value smaller than two and as the third one, the experiments with ΔAIC value smaller than five. As we mentioned before, distributions with small ΔAIC values can be accepted as alternative proper models, if it also passes the chi-square test. Therefore, we also count the total number of experiments with acceptable ΔAIC values. The fourth criterion gives the total number of experiments with an acceptable p-value for the true distribution. It only considers the p-value and not the ΔAIC value.

The fifth criterion is an overall measure for the performance of `FitAllDiscrete(x)`. Combining the second and fourth criterion, it counts the number of experiments with ΔAIC$\leq$2 and pval$\geq$SignV.

For instance, for the Binomial distribution with n=500, 46 experiments had ΔAIC$\leq$2 and 49 experiments had acceptable p-values. In 45 cases both criteria were met.

9.4.1. Results of the Automatic Test for All Discrete Distributions

As the result of the automatic tests for nine discrete distributions considering the sample sizes 500 and 1000, we get 18 text files including 50 tables in each file as we have 50 repetition. We summarize the results using the five criteria explained above. The results are given in Table 9.7.

The results of the first criterion shows that the true distribution is recommended as the best fitting distribution for most of the repetitions in each experiment. It is also possible to get the approximation instead of the true distribution. Therefore, we consider the ΔAIC values and count the number of acceptable cases. According to the second criterion, the ΔAIC values of the true distribution in experiments is mostly smaller than two. It means that our function could find the true distribution frequently and has a substantial support with that ΔAIC value. For a deeper examination, we control the number of repetitions that the true distribution has ΔAIC smaller than five which means that the results still have support for selecting the true distribution. The results in the third column show that the true distribution could be found successfully using our function as we count almost all of them 50 for the experiments. Not only the ΔAIC value, but also the chi-square test results must be successful for the convenience of the good fits. Therefore, we first count the number of cases that passes the chi-square test which is shown in the fourth column. The results show that only a little of them could not passed the chi-square test because of insufficient p-values.

Consequently, we consider the fifth criterion which takes into account the ΔAIC value with substantial support (ΔAIC$\leq$2) and successful chi-square test results together. The overall measure which we named "number of acceptance" gives the number of proper distributions found by `FitAllDiscrete(x)`. It is clear that our input modeling tool is successful in finding the correct distributions for simulated data sets.

Table 9.7. SUMMARY TABLE.

SUMMARY TABLE					
Distribution-Sample size	**1. Number of** $\Delta AIC = 0$	**2. Number of** $\Delta AIC \leq 2$	**3. Number of** $\Delta AIC \leq 5$	**4. Number of p-val $\geq$ SignV**	**5. Number of acceptance**
Poisson-500	43	49	50	50	49
Poisson -1000	34	42	48	49	41
Binom-500	33	46	49	49	45
Binom-1000	43	50	50	49	49
NBinom-500	41	48	50	48	46
NBinom-1000	42	49	50	49	48
Geometric-500	48	50	50	47	47
Geometric-1000	48	50	50	49	49
Discrete Uniform-500	50	50	50	46	46
Discrete Uniform-1000	50	50	50	50	50
Logarithmic-500	48	50	50	50	50
Logarithmic-1000	50	50	50	49	49
Rounded Normal -500	44	49	50	50	49
Rounded Normal-1000	48	50	50	50	50
Rounded Gamma-500	22	45	50	50	45
Rounded Gamma-1000	45	50	50	49	49
Rounded Pareto-500	50	50	50	50	50
Rounded Pareto-1000	50	50	50	50	50

10. COMPARISON WITH COMMON COMMERCIAL INPUT MODELING SOFTWARES

After we made sure that our function `FitAllDiscrete(x)` is successful in finding the correct distribution, we consider comparing it with commonly used input modeling softwares: the Arena input analyzer and EasyFit. For EasyFit, we consider input analyzing using both the Kolmogorov-Smirnov (KS) and Anderson Darling (AD) tests. For each distribution in our pool we generated and saved several data sets in R using different parameters. First, we did fitting experiments manually for these data sets using our input analyzer and reported the results in a table. Then, we carried out fitting experiments for the same data sets using Arena input analyzer and EasyFit. We prepared a comparison table which shows the best fitting distribution and the parameter estimates for each software. As we know the correct distribution and parameters, we can easily interpret and compare the best fit results of our tool and the softwares we mentioned. In the following section, we explain the experiments and present their results.

10.1. Manual Experiments with Data Sets Generated in R

For the comparison study, we first need to decide on the distribution and the parameter values to generate random data sets. By using a predefined parameter set, we generate a data set and then save it in a text file. For each distribution we generated 5 random data sets in R with different parameters for the sample size 1000. Therefore, we performed 45 experiments in R for the 9 discrete distributions in our pool. We give an example below for the Poisson distribution, to show how to save a generated data set for the general usage and how to call it later.

```
y<-rpois(1000,0.5); write(y,file = "pois1-1000.txt",ncol=1)
x<-read.table(file="pois1-1000.txt")
FitAllDiscrete(x[,1])
```

According to the example, for the random data generation, 0.5 is chosen as the parameter value. By using the random generation function *rpois*, the data set is obtained. Then, it is saved in a text file as column vector due to the fact that the comparing softwares use the data in this manner. However, for our function `FitAllDiscrete(x)`, we require to specify the data column, as the data set appears with the column numbers, when it is called from R. We use this procedure for all our distributions.

We reported the results of each distribution in different sections using two summary tables. The first table shows the predefined parameter values for the data set, distribution that the data were generated from, the recommended distribution for the data set and the parameter estimates. Besides the recommended distribution, it also includes the next two acceptable distributions. The ΔAIC value of each fitting model is also given to describe the closeness to the best fitting distribution. According to this measure, the user can decide on the alternative distributions to be chosen as the result of our input analyzing study.

After completing the experiments for our `FitAllDiscrete(x)` function in R, we performed distribution fitting with the other softwares using the same data sets. Arena input analyzer allows data import from text files and EasyFit requires column vector data to paste it to the first column of the sheet. Thus, we carried out 45 experiments practically for each software using our text files of the data sets. In order to compare the results of the input analyzers, we formed a second table which gives the distribution, the parameters used to generate the sample, the recommended distributions with its parameter estimates and p-values.

In general, the results can be classified in four different groups.

(i) Parametric case with one good fitting distribution
(ii) Parametric case with more than one good fitting distribution (discrete approximation case)
(iii) Non-parametric case with no good fitting distribution

(iv) Continuous approximation case

10.1.1. Experiments for the Poisson Distribution

For the Poisson distribution, the predefined λ values 0.5, 1.2, 2.5, 7.5 and 35.5 were chosen and data sets were generated according to them. As it is shown in Table 10.1, our function finds the Poisson distribution in four out of five experiments and the accuracy is acceptable for the estimated parameter values. In the fourth experiment, the negative Binomial distribution is recommended however we see that Poisson has 0.071 ΔAIC value which is very small. The Poisson distribution can be approximated by the Binomial and the negative Binomial distribution. Therefore the results in the second and the third ranks are not surprising. They can also be accepted as alternative models as the ΔAIC values are small. We give an example below for the case of more than one good fitting distribution.

Table 10.1. Best fitting and proper distributions found by `FitAllDiscrete(x)`.

BEST FITTING AND PROPER DISTRIBUTIONS FOR THE POISSON DISTRIBUTION			
Sample	**1st rank**	**2nd rank**	**3rd rank**
Poisson(0.5)	Pois(0.484) ΔAIC:0	Binom(949,5.1e-04) ΔAIC:1.99	NBinom(1.0e+05,1) ΔAIC:2.00
Poisson(1.2)	Pois(1.182) ΔAIC:0	Binom(25,0.047) ΔAIC:0.897	NBinom(9.9e+06,1) ΔAIC:2.00
Poisson(2.5)	Pois(2.581) ΔAIC:0	NBinom(410,0.993) ΔAIC:1.980	Binom(8.9e+04,2.86e-05) ΔAIC:2.00
Poisson(7.5)	NBinom(1.13,0.937) ΔAIC:0	Pois(7.547) ΔAIC:0.071	Binom(1.7e+05,4.4e-05) ΔAIC:2.074
Poisson(35.5)	Pois(35.537) ΔAIC:0	NBinom(893,0.962) ΔAIC:1.235	Binom(5.7e+05,6.2e-05) ΔAIC:2.00

10.1.1.1. Example 1: Parametric case with more than one good fitting distribution. The following experiment for the Poisson distribution corresponds to the case with more than one good fitting distribution, in other words a discrete approximation case. Here, besides the recommended Poisson distribution, the Binomial and the negative Binomial distributions are also found as alternative good fitting models. They are also proper models as the ΔAIC values are small.

Table 10.2. Example for parametric case with more than one good fitting distribution.

```
x<-rpois(1000,0.5)
FitAllDiscrete(x)

[1] "Recommend"
$model
[1] "It is recommended to use the Poisson distribution with the parameter(s):
 lambda=4.84e-01"
$AIC
[1] 1.8236e+03
$parameters
[1] 4.84e-01
$summary
                   AICvalue parameter1 parameter2       PVal   Decision  Delta AIC
Poisson          1.8236e+03 4.8400e-01          NA 9.4502e-01     accept 0.0000e+00
Binomial         1.8256e+03 9.4900e+02 5.1001e-04 7.4408e-01     accept 1.9999e+00
NegativeBinomial 1.8256e+03 1.0000e+05 1.0000e+00 7.3658e-01     accept 2.0000e+00
Geometric        1.8760e+03 6.7385e-01          NA 7.5187e-11     reject 5.2459e+01
RoundedNormal    2.1210e+03 4.8982e-01 6.3924e-01 0.0000e+00     reject 2.9740e+02
Uniform          3.2229e+03 0.0000e+00 4.0000e+00 0.0000e+00     reject 1.3993e+03
RoundedGamma     4.3154e+03 1.1490e+01 7.1105e-02 0.0000e+00     reject 2.4918e+03
Logarithmic              NA         NA          NA         NA not avail.         NA
RoundedPareto            NA         NA          NA         NA not avail.         NA
 In MLELog(x) : Negative Value or Zero
 In RoPareto(x): Negative Value or Zero
```

10.1.1.2. Comparison for the Poisson Distribution. Using the generated data sets, we performed the experiments with the Arena input analyzer and EasyFit this time. Table 10.3 shows the best fitting distributions and the parameter estimates for all softwares. `FitAllDiscrete(x)` recommends the Poisson distribution four times and the ΔAIC and p-values are acceptable for the remaining case. Arena input analyzer can not find the Poisson distribution two times and recommends a continuous distribution instead. For EasyFit, in two cases we get recommendations to use the Uniform distribution that is not reasonable at all. Therefore, `FitAllDiscrete(x)` is more successful in finding the best fitting distributions for the Poisson experiments.

Table 10.3. Best fitting and proper distributions found by All Softwares.

BEST FITTING AND PROPER DISTRIBUTIONS FOUND BY ALL SOFTWARES				
Sample	**1 FitAllDiscrete**	**2 Arena InputAnalyzer**	**3 EasyFitKS**	**4 EasyFit AD**
Poisson(0.5)	Pois(0.484) pval:0.945	POIS(0.484) pval:0.707	DUnif(0,1) -	Binom(915, 5.2e-04) -
Poisson(1.2)	Pois(1.182) pval:0.645	-0.5 + WEIB(1.85, 1.68) pval:0.158	DUnif(0, 3) -	Pois(1.182) -
Poisson(2.5)	Pois(2.581) pval:0.177	POIS(2.58) pval:0.19	Pois(2.581) -	Pois(2.581) -
Poisson(7.5)	NBinom(1.13,0.937) pval:0.961	POIS(7.55) pval>0.75	Pois(7.547) -	NBinom(116, 0.939) -
Poisson(35.5)	Pois(35.537) pval:0.671	NORM(35.5, 6.08) pval:0.685	NBinom(901, 0.962) -	NBinom(901, 0.962) -

10.1.2. Experiments for the Binomial Distribution

For the Binomial distribution, we used the parameter pairs $(3, 0.38)$, $(21, 0.52)$, $(50, 0.52)$, $(100, 0.74)$ and $(1200, 0.74)$ for n_b and p respectively. As seen in Table 10.4, our input analyzing function recommends the Binomial distribution twice for the five experiments. For the remaining cases the ΔAIC values for the Binomial distribution are smaller than one. Therefore all results can be called successful. The alternative good fitting distributions are reasonable as the Binomial distribution can be approximated by the rounded Normal and the rounded Gamma distributions. We have also satisfactory results for the estimated parameter values.

10.1.2.1. Comparison for the Binomial Distribution. For the Binomial distribution comparison experiments, we formed Table 10.5. Our function recommends the Binomial

Table 10.4. Best fitting and proper distributions found by `FitAllDiscrete(x)`.

BEST FITTING AND PROPER DISTRIBUTIONS FOR THE BINOMIAL DISTRIBUTION			
Sample	**1st rank**	**2nd rank**	**3rd rank**
Binom(3,0.38)	Binom(3, 0.368) ΔAIC:0	- -	- -
Binom(21,0.52)	RoNorm(10.9,2.35) ΔAIC:0	Binom(22,0.496) ΔAIC:0.403	- -
Binom(50,0.52)	RoNorm(26.23, 3.65) ΔAIC:0	Binom(54, 0.486) ΔAIC:0.534	RoGamma(50.64,0.518) ΔAIC:10.53
Binom(100,0.74)	RoNorm(74, 4.3) ΔAIC:0	Binom(99,0.747) ΔAIC:0.511	RoGamma(293.72,0.252) ΔAIC:5.579
Binom(1200,0.74)	Binom(1201, 0.739) ΔAIC:0	RoNorm(887.74,15.23) ΔAIC:0.88	RoGamma(3392.4,0.262) ΔAIC:2.2

distribution two times but also finds acceptable ΔAIC and p-values for it in the remaining cases. It is important to note that Arena input analyzer does not include the Binomial distribution in its pool and it does not offer a choice for doing a test only with discrete distributions. Therefore, it considers all continuous distributions and chooses the Normal and Weibull distributions for the experiments. According to the table, `FitAllDiscrete(x)` and the Arena input analyzer find reasonable results as the Normal distribution approximation is possible for the Binomial distribution when the parameter n_b gets larger. EasyFit software offers a choice for selecting the discrete distributions. However, in three cases, it recommends the Uniform distribution although it has the Binomial distribution in its list. According to the results, `FitAllDiscrete(x)` is a more suitable input analyzing tool for data coming from the Binomial distribution.

Table 10.5. Best fitting and proper distributions found by All Softwares.

BEST FITTING AND PROPER DISTRIBUTIONS FOUND BY ALL SOFTWARES				
Sample	**1 FitAllDiscrete**	**2 Arena InputAnalyzer**	**3 EasyFitKS**	**4 EasyFit AD**
Binom(3,0.38)	Binom(3, 0.38) pval:0.09	-0.5 + WEIB* BETA(1.84, 2.19) pval:0.219	DUnif(0, 2) -	Pois(1.104) -
Binom(21,0.52)	RoNorm(10.9, 2.35) pval:0.613	NORM(10.9, 2.37) pval:0.579	DUnif(7, 15) -	Pois(10, 907) -
Binom(50,0.52)	RoNorm(26.23, 3.65) pval:0.243	NORM(26.2, 3.66) pval:0.244	Binom(53, 0.491) -	Binom(53, 0.491) -
Binom(100,0.74)	RoNorm(74, 4.3) pval:0.651	NORM(74, 4.31) pval:0.644	DUnif(67, 81) -	Binom(98, 0.749) -
Binom(1200,0.74)	Binom(1201, 0.739) pval:0.219	NORM(888, 15.2) pval:0.18	Binom(1201, 0.739) -	Binom(1201, 0.739) -

10.1.3. Experiments for the Negative Binomial

Here, the negative Binomial data sets were generated using the parameter values $(3, 0.38)$, $(21, 0.52)$, $(50, 0.52)$, $(100, 0.74)$ and $(1200, 0.74)$ for the parameters r and p. Similar to the Poisson and the Binomial distribution, these experiments give more than one good fitting distributions for the data sets as seen in Table 10.6. The negative Binomial distribution is often found in the first rank however it is also possible to get the rounded Gamma distribution as the best fitting one as seen in the last experiment. However, for this case, the negative Binomial is found in the second rank with ΔAIC value equal to 0.544 which is quite small and therefore reasonable.

Table 10.6. Best fitting and proper distributions found by `FitAllDiscrete(x)`.

BEST FITTING AND PROPER DISTRIBUTIONS FOR THE NEGATIVE BINOMIAL DISTRIBUTION			
Sample	**1st rank**	**2nd rank**	**3rd rank**
NBinom(2,0.38)	NBinom(2,0.384) ΔAIC:0	RoGamma(1.149,2.808) ΔAIC:11.99	- -
NBinom(21,0.52)	NBinom(20, 0.509) ΔAIC:0	RoGamma(9.534,2.019) ΔAIC:4.8	- -
NBinom(50,0.52)	NBinom(48, 0.508) ΔAIC:0	RoGamma(23.33,1.996) ΔAIC:0.988	RoNorm(46.55,9.579) ΔAIC:17.20
NBinom(100,0.74)	NBinom(107, 0.754) ΔAIC:0	RoGamma(25.958, 1.346) ΔAIC:1.457	RoNorm(34.926,6.81) ΔAIC:12.54
NBinom(1200,0.74)	RoGamma(300.79,1.4) ΔAIC:0	NBinom(1045,0.712) ΔAIC:0.544	RoNorm(422.34,24.359) ΔAIC:2.858

10.1.3.1. Comparison for the Negative Binomial Distribution. Table 10.7 shows the recommended distributions for data sets coming from the negative Binomial distribution. The results show that, `FitAllDiscrete(x)` can find the negative Binomial distribution in four experiments successfully. For the fifth experiment, rounded Gamma distribution is recommended but ΔAIC and p-values are reasonable for the negative Binomial distribution. For Arena input analyzer, the closest continuous distributions with acceptable p-values are recommended including the Gamma and the Normal distributions. As the negative Binomial distribution can possibly be approximated by these continuous distributions, the results are not surprising. EasyFit could find the negative Binomial distribution in four times out of five. For the remaining cases, it finds the Poisson distribution which can be approximated by the negative Binomial distribution. When we consider both the recommendations and the quality of the es-

timates, it is clear that `FitAllDiscrete(x)` performs better than the others for data coming from the negative Binomial distribution.

Table 10.7. Best fitting and proper distributions found by All Softwares.

BEST FITTING AND PROPER DISTRIBUTIONS FOUND BY ALL SOFTWARES				
Sample	**1 FitAllDiscrete**	**2 Arena InputAnalyzer**	**3 EasyFitKS**	**4 EasyFit AD**
NBinom(2,0.38)	NBinom(2, 0.384) pval:0.325	-0.5 + GAMM(2.19, 1.69) pval:0.426	Pois(3.214) -	NBinom(2, 0.403) -
NBinom(21,0.52)	NBinom(20, 0.509) pval:0.792	4.5 + ERLA(2.95, 5) pval:0.0439	NBinom(20, 0.511) -	NBinom(20, 0.511) -
NBinom(50,0.52)	NBinom(48, 0.508) pval:0.323	20.5 + 63 * BETA(3.92, 5.55) pval:0.309	NBinom(47, 0.507) -	NBinom(47, 0.507) -
NBinom(100,0.74)	NBinom(107, 0.754) pval:0.483	15.5 + 42 * BETA(4.6, 5.32), pval<0.005	NBinom(106, 0.753) -	NBinom(106, 0.753) -
NBinom(1200,0.74)	RoGamma(300.79,1.4) pval:0.338	NORM(422, 24.4), pval:0.0362	Pois(422.34) -	Pois(422.34) -

10.1.4. Experiments for the Uniform Distribution

In these experiments, we generated five data sets coming from the Uniform distribution using the parameter sets $(2, 20)$, $(2, 50)$, $(20, 150)$, $(220, 1200)$ and $(1210, 1440)$ for the parameters a and b. The results show that the Uniform disribution is recommended by `FitAllDiscrete(x)` three times in the five experiments as shown in Table 10.8. The first three experiments correspond to the parametric case with one good fitting distribution as we do not have any other proper alternatives for the best fitting distribution. Example 1 is related to this case. In the fourth and the fifth experiment, the p-value of the chi-square test can not be calculated by our code because of the large number of different integers with small probabilities. Our analyzer informs that the p-value can not be calculated and recommends using continuous distributions for modeling. In Section 6.3.3, we explain how to decide in such a situation. Also, see example 2 below, for an example of the continuous approximation case.

Table 10.8. Best fitting and proper distributions found by `FitAllDiscrete(x)`.

BEST FITTING AND PROPER DISTRIBUTIONS FOR THE UNIFORM DISTRIBUTION					
Sample	Unif(2, 20)	Unif(2,50)	Unif(20,150)	Unif(220, 1200)	Unif(1210, 1440)
Best Fitting Distribution	Unif(2,20)	Unif(2,50)	Unif(20,150)	Unif(220, 1200)	Unif(1210, 1440)

10.1.4.1. Example 2: Parametric case with one good fitting distribution. In this example, our input analyzing function finds and recommends the Uniform distribution successfully for the data. All other distributions are rejected since shapes are quite different from the Uniform distribution. Large ΔAIC values also confirm that the distributions except the Uniform do not have a good fit to the data. Therefore, this is the case of only one good fitting distribution.

Table 10.9. Example for parametric case with one good fitting distribution.

```
x<-floor(runif(1000)*(20-2+1)+2)
FitAllDiscrete(x)

[1] "Recommend"
$model
[1] "It is recommended to use the Uniform distribution with the parameter(s):
 lowerb=2e+00,upperb=2e+01"
$AIC
[1] 5.8929e+03
$parameters
[1] 2e+00 2e+01
$summary
                   AICvalue parameter1 parameter2       PVal Decision  Delta AIC
Uniform          5.8929e+03 2.0000e+00 2.0000e+01 7.0275e-01   accept 0.0000e+00
NegativeBinomial 6.2326e+03 5.0000e+00 3.1610e-01 0.0000e+00   reject 3.3967e+02
RoundedNormal    6.2383e+03 1.0817e+01 5.4563e+00 0.0000e+00   reject 3.4541e+02
RoundedGamma     6.2442e+03 3.1016e+00 3.4876e+00 0.0000e+00   reject 3.5129e+02
Geometric        6.8541e+03 8.4617e-02         NA 0.0000e+00   reject 9.6126e+02
Poisson          7.0714e+03 1.0818e+01         NA 0.0000e+00   reject 1.1785e+03
Binomial         7.0734e+03 1.9999e+06 5.4092e-06 0.0000e+00   reject 1.1805e+03
RoundedPareto    7.2576e+03 2.3230e+00 7.2771e-01 0.0000e+00   reject 1.3647e+03
Logarithmic      7.5813e+03 9.7574e-01         NA 0.0000e+00   reject 1.6885e+03
```

10.1.4.2. Example 3: Continuous Approximation Case. In this example, the data set has a large number of different observed values with small probabilities. In this case, our implementation of the chi-square test can not calculate a result. Therefore, the user is recommended to use another distribution fitting tool to select an approximate continuous distribution.

Table 10.10. Example for Continuous Approximation Case.

```
x<-floor(runif(1000)*(150-20+1)+20)
FitAllDiscrete(x)

[1] "Recommend"
$model
[1] "The data set can be approximated by a continuous distribution. Due to the very
large number of different observed integers, it is not possible to make chi-square
test for discrete distributions. Try to use a continuous distribution to model this
data. Best fitting discrete distribution is Uniform distribution with the
parameter(s): lowerb=2e+01,upperb=1.5e+02"
$AIC
[1] 9.7544e+03
$parameters
[1] 2.0e+01 1.5e+02
$summary
                   AICvalue parameter1 parameter2  PVal   Decision  Delta AIC
Uniform          9.7544e+03 2.0000e+01 1.5000e+02    NA not avail. 0.0000e+00
RoundedNormal    1.0098e+04 8.4623e+01 3.7631e+01 0e+00     reject 3.4314e+02
RoundedGamma     1.0118e+04 4.1841e+00 2.0225e+01 0e+00     reject 3.6351e+02
NegativeBinomial 1.0122e+04 4.0000e+00 4.5135e-02 0e+00     reject 3.6745e+02
Geometric        1.0890e+04 1.1679e-02         NA 0e+00     reject 1.1358e+03
RoundedPareto    1.1173e+04 2.0393e+01 7.7006e-01 0e+00     reject 1.4185e+03
Logarithmic      1.2622e+04 9.9812e-01         NA 0e+00     reject 2.8677e+03
Poisson          2.4104e+04 8.4623e+01         NA 0e+00     reject 1.4349e+04
Binomial         2.4107e+04 1.5000e+06 5.6415e-05 0e+00     reject 1.4352e+04
```

10.1.4.3. Comparison for the Uniform Distribution. Using the random samples generated from the Uniform distribution, we performed the experiments for the three softwares. Then, we formed the comparison table as seen in Table 10.11. According to the results, our analyzer finds the correct distribution for the three discrete data sets and for the other two sets, it recommends using continuous distributions for modeling. Arena input analyzer finds the Beta distribution as the best fitting distribution with large p-values in all experiments. It is not surprising as Arena software does not include the Uniform distribution and finds the best fitting continuous distribution. EasyFit includes the Discrete Uniform distribution in its pool and it could find the Uniform distribution in many cases. However, `FitAllDiscrete(x)` is more successful in finding the correct distribution also with better estimates.

Table 10.11. Best fitting and proper distributions found by All Softwares.

BEST FITTING AND PROPER DISTRIBUTIONS FOUND BY ALL SOFTWARES				
Sample	**1 FitAllDiscrete**	**2 Arena InputAnalyzer**	**3 EasyFitKS**	**4 EasyFit AD**
Unif(2,20)	Unif(2,20) pval:0.703	1.5 + 19 * BETA(0.99, 1.03), pval:0.735	DUnif(2, 20) -	NBinom(6, 0.362) -
Unif(2,50)	Unif(2,50) pval:0.259	1.5 + 49 * BETA(0.953, 0.946), pval:0.366	DUnif(2, 51) -	DUnif(2. 51) -
Unif(20,150)	Unif(20,150) pval:NA	20 + 130 * BETA(0.894, 0.952) pval:0.211	DUnif(17, 149) -	DUnif(17, 149) -
Unif(220,1200)	Unif(220,1200) pval:NA	220 + 980 * BETA(1.07, 1.06) pval:134	DUnif(233, 1191) -	DUnif(233, 1191) -
Unif(1210,1440)	Unif(1210,1440) pval:NA	.21e+003 + 230 * BETA(0.991, 1.07) pval:0.646	DUnif(1207, 1434) -	DUnif(1207, 1434) -

10.1.5. Experiments for the Geometric Distribution

For the Geometric distribution, the predefined parameter values 0.09, 0.23, 0.47, 0.69 and 0.90 were chosen for parameter p and the data sets were generated with respect to these parameter values. According to the results, when using our tool the Geometric distribution is recommended for most of the experiments with good parameter estimates as seen in Table 10.12. For the remaining case, the negative Binomial distribution is recommended. This is not surprising, as the Geometric distribution is a special case of the negative Binomial distribution. However, we also control the ΔAIC and p-values of the Geometric distribution for this experiment. The results show that Geometric distribution also has a good fit with a small ΔAIC and an acceptable

p-value.

Table 10.12. Best fitting and proper distributions found by `FitAllDiscrete(x)`.

BEST FITTING AND PROPER DISTRIBUTIONS FOR THE GEOMETRIC DISTRIBUTION		
Sample	**1st rank**	**2nd rank**
Geom(0.09)	Geom(0.085) ΔAIC:0	- -
Geom(0.23)	Geom(0.236) ΔAIC:0	NBinom(1,0.236) ΔAIC:2
Geom(0.47)	Geom(0.458) ΔAIC:0	NBinom(1,0.458) ΔAIC:2
Geom(0.69)	NBinom(2, 0.818) ΔAIC:0	Geom(0.692) ΔAIC:0.084
Geom(0.90)	Geom(0.893) ΔAIC:0	- -

10.1.5.1. Comparison for the Geometric Distribution. In Table 10.13, we can see that our function finds the Geometric distribution for four experiments. For the remaining case, the ΔAIC value and the p-value are acceptable. Arena input analyzer finds some continuous distributions however as the value of the parameter p gets larger, it has difficulty to fit a distribution to our discrete data set as the p-values get smaller. The performance of EasyFit is not good since the selection criteria is not successful. For the three larger parameter values, it is never able to find the correct distribution and in one case of KS or AD, suggests the Poisson or the Uniform distribution. Therefore, input analyzing with `FitAllDiscrete(x)` is better for the data sets coming from the Geometric distribution.

Table 10.13. Best fitting and proper distributions found by All Softwares.

BEST FITTING AND PROPER DISTRIBUTIONS FOUND BY ALL SOFTWARES				
Sample	**1 FitAllDiscrete**	**2 Arena InputAnalyzer**	**3 EasyFitKS**	**4 EasyFit AD**
Geom(0.09)	Geom(0.085) pval:0.01	-0.5 + EXPO(11.2), pval:0.01	Geom(0.085) -	Geom(0.085) -
Geom(0.23)	Geom(0.236) pval:0.281	-0.5 + EXPO(3.75), pval:0.237	Geom(0.236) -	Geom(0.236) -
Geom(0.47)	Geom(0.458) pval:0.821	-0.5 + 14 * BETA(0.738, 5.4) pval:0.135	Pois(1.183) -	Pois(1.183) -
Geom(0.69)	NBinom(2, 0.818) pval:0.220	-0.5 + EXPO(0.945) pval:0.0097	DUnif(0, 1) -	Pois(0.445) -
Geom(0.90)	Geom(0.893) pval:0.862	POIS(0.12) pval<0.005	Pois(0.12) -	Pois(0.12) -

10.1.6. Experiments for the Logarithmic Distribution

As the predefined values of the parameter p, we used 0.18, 0.23, 0.47, 0.69 and 0.98 for our experiments considering that the sample size was 1000. Using `FitAllDiscrete(x)`, the Logarithmic distribution is recommended each time successfully as seen in Table 10.14. In the third experiment, we see that the rounded Pareto distribution also passes the chi-square test. However, we do not have to consider the Pareto distribution as an alternative distribution, as the ΔAIC value 6.187 is not small enough.

Table 10.14. Best fitting and proper distributions found by `FitAllDiscrete(x)`.

BEST FITTING AND PROPER DISTRIBUTIONS FOR THE LOGARITHMIC DISTRIBUTION		
Sample	**1st rank**	**2nd rank**
Log(0.18)	Log(0.179)	-
	ΔAIC:0	-
Log(0.23)	Log(0.2)	-
	ΔAIC:0	-
Log(0.47)	Log(0.45)	RoPareto(0.876,2.598)
	ΔAIC:0	ΔAIC:6.187
Log(0.69)	Log(0.694)	-
	ΔAIC:0	-
Log(0.98)	Log(0.98)	-
	ΔAIC:0	-

10.1.6.1. Comparison for the Logarithmic Distribution. Table 10.15 shows the results of the recommended distributions for the data sets generated from the Logarithmic distribution considering the three softwares we used. According to the results, our input analyzer is successful in finding the correct distribution with good parameter estimates. Arena input analyzer recommends different continuous distributions however none of them has a good p-value. Therefore, Arena software is not capable to represent the data sets coming from the Logarithmic distribution as no resembling distribution is found. EasyFit includes the Logarithmic distribution in its discrete distributions pool. However, it is not successful in finding the correct distribution as only for one experiment of the five, the Logarithmic distribution is recommended. Hence, `FitAllDiscrete(x)` is a more convenient tool for the input analyzing of data sets coming from the Logarithmic distribution.

Table 10.15. Best fitting and proper distributions found by All Softwares.

BEST FITTING AND PROPER DISTRIBUTIONS FOUND BY ALL SOFTWARES				
Sample	**1 FitAllDiscrete**	**2 Arena InputAnalyzer**	**3 EasyFitKS**	**4 EasyFit AD**
Log(0.18)	Log(0.179) pval:0.423	0.5 + 4 * BETA(6.01, 32.1) pval<0.005	Pois(1.106) -	Pois(1.106) -
Log(0.23)	Log(0.2) pval:0.153	0.5 + GAMM(0.108, 5.72) pval<0.005	Pois(1.12) -	Pois(1.12) -
Log(0.47)	Log(0.45) pval:0.665	0.5 + LOGN(0.821, 0.529) pval<0.005	Pois(1.368) -	Pois(1.368) -
Log(0.69)	Log(0.694) pval:0.692	0.5 + 18 * BETA(0.609, 7.14)), pval:0.0079	DUnif(0, 4) -	Pois(1.914) -
Log(0.98)	Log(0.98) pval:0.316	0.999 + 163 * BETA(0.201, 2.63) pval<0.005	Geom(0.07) -	Log(0.98) -

10.1.7. Experiments for the Rounded Normal Distribution

We generated five data sets coming from the rounded Normal distribution using the predefined parameter values $(5, 0.8)$, $(52.2, 0.65)$, $(63, 4.5)$, $(126.5, 20)$ and $(250, 2.4)$ for the parameters μ and σ respectively with the sample size 1000. According to the experiments using `FitAllDiscrete(x)`, the rounded Normal distribution is recommended four times with closely estimated parameters as seen in Table 10.16. For the remaining case, the Binomial distribution is recommended however the rounded Normal distribution still corresponds to a good fit as the ΔAIC value is small and the p-value is large.

Table 10.16. Best fitting and proper distributions found by `FitAllDiscrete(x)`.

BEST FITTING AND PROPER DISTRIBUTIONS FOR THE ROUNDED NORMAL DISTRIBUTION			
Sample	**1st rank**	**2nd rank**	**3rd rank**
RoNorm(5,0.8)	RoNorm(4.98, 0.82) ΔAIC:0	- -	- -
RoNorm(52.2,0.65)	RoNorm(52.17,0.65) ΔAIC:0	- -	- -
RoNorm(63,4.5)	Binom(94,0.67) ΔAIC:0	RoNorm(62.84,4.55) ΔAIC:2.487	RoGamma(188.042,0.334) ΔAIC:12.27
RoNorm(126.5,20)	RoNorm(126,24,19.94) ΔAIC:0	- -	- -
RoNorm(250,2.4)	RoNorm(249.92,2.42) ΔAIC:0	RoGamma(1048.65,0.0238) ΔAIC:0.394	- -

10.1.7.1. Comparison for the Rounded Normal Distribution. Table 10.17 shows the results of the recommended distributions considering three softwares we use. According to it, `FitAllDiscrete(x)` is successful in recommending the correct distribution with

good parameter estimates. Arena input analyzer finds the Normal distribution with acceptable p-values for three experiments. However, for the first two experiments, the p-values are too low and we can not consider them. EasyFit recommends the Binomial, Poisson and the negative Binomial distributions which are not bad as all of these can possibly approximate the rounded Normal distribution. However, it also recommends the Uniform distribution two times which is not suitable for the data. Therefore, for data sets coming from the rounded Normal distribution, it is best to use our function.

Table 10.17. Best fitting and proper distributions found by All Softwares.

BEST FITTING AND PROPER DISTRIBUTIONS FOUND BY ALL SOFTWARES				
Sample	**1 FitAllDiscrete**	**2 Arena InputAnalyzer**	**3 EasyFitKS**	**4 EasyFit AD**
RoNorm(5,0.8)	RoNorm(4.98, 0.82) pval:0.981	1.5 + 7*BETA(10, 10.2) pval<0.005	DUnif(4, 6) -	Poisson(4,981) -
RoNorm(52.2,0.65)	RoNorm(52.17, 0.65) pval:0.605	49.5 + 5 * BETA(8.53, 7.32) pval<0.005	Pois(52.175) -	Pois(52.175) -
RoNorm(63,4.5)	Binom(94, 0.67) pval:0.614	NORM(62.8, 4.56) pval:0.568	Binom(93, 0.669) -	Binom(93, 0.669) -
RoNorm(126.5,20)	RoNorm(126,24, 19.94) pval:0.104	NORM(126, 19.9) pval:0.076	DUnif(92, 160) -	NBinom(58, 0.318) -
RoNorm(250,2.4)	RoNorm(249.92, 2.42) pval:0.256	NORM(250, 2.44) pval:0.14	Binom(256, 0.976) -	Binom(256, 0.976) -

10.1.8. Experiments for the Rounded Gamma Distribution

We used the parameter values $(1.2, 2.4)$, $(3.5, 3.5)$, $(9, 4.8)$, $(12.6, 3)$ and $(20, 0.4)$ with sample size 1000. According to the Table 10.18, rounded Gamma distribution is recommended successfully in all experiments. In the second rank, the negative Binomial approximation to the Gamma distribution is seen from the table. For this case, we consider the small ΔAIC and large p-values. According to these experiments with more than one good fitting distribution, the negative Binomial can also be accepted as the alternative good fit to represent the data sets.

10.1.8.1. Comparison for the Rounded Gamma Distribution. Using the random samples generated from the rounded Gamma distribution, we carried out the experiments with all three softwares. Then we formed the Table 10.19 for comparison. According to the results, our input analyzer recommends the correct distribution each time with successfully estimated parameters. Arena input analyzer tries to fit continuous

Table 10.18. Best fitting and proper distributions found by `FitAllDiscrete(x)`.

BEST FITTING AND PROPER DISTRIBUTIONS FOR THE ROUNDED GAMMA DISTRIBUTION		
Sample	**1st rank**	**2nd rank**
RoGamma(1.2,2.4)	RoGamma(1.23,2.43) ΔAIC:0	- -
RoGamma(3.5,3.5)	RoGamma(3.54,3.47) ΔAIC:0	NBinom(5,0.289) ΔAIC:3.273
RoGamma(9,4.8)	RoGamma(8.51,5.16) ΔAIC:0	NBinom(11, 0.2) ΔAIC:2.909
RoGamma(12.6,3)	RoGamma(12.87,2.96) ΔAIC:0	NBinom(20, 0.344) ΔAIC:0.115
RoGamma(20,0.4)	RoGamma(20,0.4) ΔAIC:0	- -

distribution to the data sets. However, it can not always find a good matching one considering the p-values. Hence, it is not capable to represent the data sets coming from the rounded Gamma distribution. EasyFit tries to find the best matching discrete distribution from its list. Similar to some previous experiments, it finds the Uniform distribution which in fact does not have a similar shape to the rounded Gamma distribution. Therefore, our function is more suitable for the input analyzing of data sets coming from the rounded Gamma distribution.

Table 10.19. Best fitting and proper distributions found by All Softwares.

BEST FITTING AND PROPER DISTRIBUTIONS FOUND BY ALL SOFTWARES				
Sample	**1 FitAllDiscrete**	**2 Arena InputAnalyzer**	**3 EasyFitKS**	**4 EasyFit AD**
RoGamma(1.2,2.4)	RoGamma(1.23, 2.43) pval:0.129	-0.5 + LOGN(3.61, 3.57) pval:0.006	Pois(2.973) -	Pois(2.973) -
RoGamma(3.5,3.5)	RoGamma(3.54, 3.47) pval:0.797	0.5 + WEIB(13.4, 1.94) pval:0.465	DUnif(2, 23) -	NBinom(5, 0.302) -
RoGamma(9,4.8)	RoGamma(8.51, 5.16) pval:0.979	11 + WEIB(37.7, 2.34) pval<0.005	NBinom(10, 0.191) -	NBinom(10, 0.191) -
RoGamma(2.6,3)	RoGamma(12.87, 2.96) pval:0.226	15.5 + WEIB(25.6, 2.28) pval:0.288	DUnif(20, 56) -	NBinom(19, 0.344) -
RoGamma(20,0.4)	RoGamma(20, 0.4) pval:0.802	2.5 + ERLA(0.616, 9) pval:0.66	DUnif(5, 11) -	Pois(8.04) -

10.1.9. Experiments for the Rounded Pareto Distribution

We generated five data sets using the predefined parameter values $(1, 1)$, $(2.5, 3.8)$, $(20, 3)$, $(25.3, 2.4)$ and $(30, 21.9)$ for the parameters a and k with sample size 1000. According to the Table 10.20, the rounded Pareto is found each time using our function. We also get good quality parameter estimates. As any other alternative proper distri-

bution is not found for the experiments, the experiments corresponds to the one good fitting distribution case.

Table 10.20. Best fitting and proper distributions found by `FitAllDiscrete(x)`.

BEST FITTING AND PROPER DISTRIBUTIONS FOR THE ROUNDED PARETO DISTRIBUTION					
Sample	RoPareto(1,1)	RoPareto(2.5,3.8)	RoPareto(20,3)	RoPareto(25.3,2.4)	RoPareto(30,21.9)
Best Fitting Distribution	RoPareto(0.94,0.95)	RoPareto(2.5,3.5)	RoPareto(19.96,2.9)	RoPareto(25.32,2.27)	RoPareto(30,22.3)

10.1.9.1. Comparison for the Rounded Pareto Distribution. Using the generated data sets from the rounded Pareto distribution, we performed the experiments for Arena input analyzer and EasyFit this time. Table 10.21 represents the best fitting distributions with parameter estimates. Our input analyzing function is successful in finding the correct distribution with good parameter estimates. However, the Arena input analyzer can not find good fitting distributions each time as the p-values are very small. EasyFit recommends different distributions again including the Uniform distribution. The recommended distributions do not represent the rounded Pareto distribution well. For the data sets coming from the rounded Pareto distribution, we again get better results for `FitAllDiscrete(x)`.

Table 10.21. Best fitting and proper distributions found by All Softwares.

BEST FITTING AND PROPER DISTRIBUTIONS FOUND BY ALL SOFTWARES				
Sample	**1 FitAllDiscrete**	**2 Arena InputAnalyzer**	**3 EasyFitKS**	**4 EasyFit AD**
RoPareto(1,1)	RoPareto(0.94, 0.95) pval:0.313	0.999 + WEIB(1.11, 0.328) pval<0.005	Log(0.997) -	Log(0.997) -
RoPareto(2.5,3.8)	RoPareto(2.5, 3.5) pval:0.925	2.5 + 11 * BETA(0.559, 5.08) pval<0.005	DUnif(2, 5) -	Pois(3.591) -
RoPareto(20,3)	RoPareto(19.96, 2.9) pval:0.167	20 + 145 * BETA(0.571, 6.95) pval:0.19	DUnif(4, 57) -	NBinom(4, 0.127) -
RoPareto(25.3,2.4)	RoPareto(25.32, 2.27) pval:0.401	25 + WEIB(16.6, 0.741) pval<0.005	DUnif(-28, 120) -	Geom(0.0214) -
RoPareto(30,21.9)	RoPareto(30, 22.3) pval:0.997	29.5 + GAMM(1.02, 1.85) pval:0.042	DUnif(29, 34) -	Pois(31.382) -

10.2. Summarizing the Results

According to our experiments, the existing well known input modeling softwares do not allow a complete comparison study. First of all, because the contents of the

distribution pools are different.

As we mentioned in the experiments, the Arena input analyzer considers only one discrete distribution; the Poisson distribution. Also, it does not allow the user to make distribution fitting only for the discrete distributions. Therefore, it fits the Poisson and the continuous distributions together and finds the best fitting of them. It is significant that it uses the same goodness of fit method we use: chi-square test. However, the distribution selection criterion it uses is not sufficient, as even for the data sets coming from the Poisson distribution, it can not always find the Poisson distribution. Another problem is that Arena input analyzer should not recommend a continuous distribution for the discrete data sets at least for those with a small number of observed values. For example, if a data set includes only three different observed values, the software should not recommend a continuous distribution as the best fitting one.

EasyFit has more discrete distributions in its pool and allows to make fitting for discrete distributions. However, it does not support the chi-square test for discrete distributions. It uses the Kolmogorov Smirnov or the Anderson Darling tests instead. It is important to note that, some results of EasyFit examples are significantly wrong especially for the KS test. AD test gives a little better results than KS as seen in many of the experiments. But in general, the selection criteria of the AD or KS test statistics do not give sufficiently good results.

`FitAllDiscrete(x)` gives successful results in distribution fitting for discrete data sets using the methods: AIC for distribution selection, MLE for parameter estimation and the chi-square test for goodness of fitting. It has a quite large pool with nine discrete distributions and in most of the cases, it can recommend the correct distribution. For the remaining cases, an approximation of the correct distribution is recommended which fits a little better than it. However, `FitAllDiscrete(x)` finds small ΔAIC and acceptable p-values for the correct distribution which confirms the success of our tool. It is obvious from the experiments that our tool gives better results than the other softwares with regards to the success of finding the correct distribution. Besides, we

can see that the estimates are not bad. The results also clearly show that AIC is the best criterion to select discrete distributions.

11. CONCLUSION

In this study, we implemented a new input modeling tool called `FitAllDiscrete(x)`. It was developed as an R-function and is designed for the input modeling of discrete data sets. We aimed to build a good input analyzer which is able to recommend the most proper distribution for the data. The idea basically arose from our desire to have a better tool than the existing commercial softwares Arena input analyzer and EasyFit by removing the limitations of them.

`FitAllDiscrete(x)` uses a large pool of discrete distributions. We included six well-known standard discrete distributions (e.g: Poisson, Binomial) and also added three rounded discrete distributions (rounded Normal, rounded Gamma and rounded Pareto) which were defined from the corresponding continuous distributions. We thus added new discrete distributions to our pool and increased the number of possible shapes. Our tool only deals with independent and identically distributed (iid) input data coming from discrete distributions.

To execute input modeling, we carried out three steps including parameter estimation, distribution selection and goodness of fit test. For parameter estimation, we used the maximum likelihood estimation (MLE) and obtained acceptable estimates with close values to the original parameters. The Arena input analzer and EasyFit use different parameter estimation methods together such as maximum likelihood estimation (MLE), least square error (LSE) and method of moments (MOM) based on the distribution. Our method considers the accuracy for having good quality estimates rather than having them faster. Therefore, we could prefer a successful method to have closely estimated parameters.

Distribution selection is the second and very crucial part of input modeling. We used Akaike's information criterion (AIC) to select the best fitting distribution for the data. Both Akaike's information criterion (AIC) and maximum likelihood estimation

(MLE) requires the log-likelihood function of the fitted distribution. Therefore, using these two methods together is natural. Akaike's information criterion (AIC) has also the advantage that it considers the number of parameters of the distributions. As the number of parameters increases, it calculates a larger penalty value for that distribution. Thus, AIC decreases the tendency for choosing distributions with large parameters by penalizing. The square error (SE) that Arena input analyzer uses and Kolmogorov Smirnov (KS), Anderson Darling (AD) test statistics that EasyFit uses do not consider the effect of number of parameters. Besides, square error relies on the number of intervals for the histograms. We have seen in our experiments that Arena input analyzer is not frequently successful in selecting the correct distribution. Compared to the results of our tool EasyFit is also not very successful for recommending the true distribution. Both softwares recommend improper distributions in a considerable number of experiments. We can therefore conclude that AIC is clearly more successful in finding the most proper distribution than the square error (SE), Kolmogorov Smirnov (KS) and Anderson Darling (AD) methods.

As the third step of input modeling, we used the chi-square test to assess the statistical adequacy of the fit. Arena input analyzer also uses the chi-square test however EasyFit executes Kolmogorov Smirnov (KS) and Anderson Darling (AD) tests for the goodness of fit step. EasyFit does not include the chi-square test although it is a basic and successful method. A likely reason is the difficulty of the implementation of an automatic interval selection procedure.

To decide how reliable our tool is, we evaluated its success by an automatic performance test. Besides that, we also compared the results manually with the mentioned commercial input analyzers. We considered the recommended best fitting distribution as the first criterion. In our tests, we used independent and identically distributed (iid) random data generated in R from the distributions of our pool. The results of our experiments show that `FitAllDiscrete(x)` is successful in representing the discrete data sets by recommending the most proper distribution for each experiment. It can recommend the correct distribution in most cases. For the remaining cases, it recom-

mends an approximation of the correct distribution. In this case, `FitAllDiscrete(x)` finds acceptable values for the selection criteria of the correct distribution (p-value and ΔAIC). Therefore, the correct distribution can still be considered as a good fit and it is not a problem that it is not recommended. Comparing with Arena input analyzer and EasyFit, we observe that the recommendations by `FitAllDiscrete(x)` are more successful.

If none of the distributions has a good fit for the data, `FitAllDiscrete(x)` recommends the empirical distribution which is estimated directly from the data. Both Arena input analyzer and EasyFit do not include a recommendation for the empirical distribution and always display the best fitting distribution even the fit is statistically not sufficient. Therefore, our function includes a different and useful property that always enables to find a proper distribution for the data.

In order to make an overall evaluation we are convinced that `FitAllDiscrete(x)` is practical as a user requires to have only basic knowledge about R. She can do input modeling easily by importing the data set x into R and then calling the function. No detailed knowledge is required about the statistical methods and their applications. The user must have only a basic knowledge on the main criteria we used to be able to interpret the results. `FitAllDiscrete(x)` is a more powerful tool than the other two softwares as it applies more powerful statistical methods as we have explained above. In addition `FitAllDiscrete(x)` is fast. It requires only about half a second to model a data set with sample size 1000.

Consequently, with the development of the function `FitAllDiscrete(x)` we were able to accomplish our aim of designing a simple and powerful input modeling tool. It is also clearly a better tool than the Arena input analyzer and EasyFit. We hope that our new input modeling tool `FitAllDiscrete(x)` can meet the expectations of the practitioners and will be frequently used for the input modeling of discrete data sets.

REFERENCES

1. Pınar, B., *Input Modeling*, M.S. Thesis, Boğaziçi University, 2012.

2. Pegden, C. and R.E. Shannon, *Introduction to Simulation Using Siman*, 2nd edition, McGraw-Hill, Inc. New York, USA, 1995.

3. Biller, B. and C. Güneş, "Introduction To Simulation Input Modeling", *2010 Winter Simulation Conference*, pp. 49-58, 2010.

4. Johnson, M.E and M. Mansooreh, "Simulation Input Data Modeling", *Annals of Operations Research*, Vol. 53, No. 1, pp. 47-75, 1994.

5. Allen, T.T., *Introduction to Discrete Event Simulation and Agent–based Modeling*, Springer-Verlag, London, 2011.

6. Banks, J., J.S. Carson, B.L. Nelson and D.M. Nicol, *Discrete Event System Simulation*, 4th Edition, Prentice Hall, NJ, USA, 2005.

7. Biller, B. and C.G. Çorlu, "Copula-Based Multivariate Input Modeling, *Surveys in Operations Research and Management Science*, Vol. 17 No. 2, pp. 69-84, 2012.

8. Biller, B. and B.L. Nelson, "Answers to Top Ten Input Modeling Questions", *2002 Winter Simulation Conference*, pp. 35-40, 2002.

9. Liong, C.Y. and C.S.E. Loo, "A Simulation Study of Warehouse Loading and Unloading Systems Using Arena", *Journal of Quality Measurement and Analysis*, Vol. 5, No. 2, pp. 45-56, 2009.

10. Mathwave, *Who Should Use EasyFit?*, 2014, `http://www.mathwave.com/easyfit-distribution-fitting.html`, [Accessed December 2014].

11. Arena, *Arena Simulation Software*, 2014, https://www.arenasimulation.com, [Accessed December 2014].

12. Altıok, T. and B. Melamed, *Simulation Modeling and Analysis with Arena*, Elsevier Inc., NJ, USA, 2007.

13. Mathwave, *EasyFit Software*, 2014, http://www.mathwave.com, [Accessed December 2014].

14. Mathwave, *What Parameter Estimation Methods Are Used in EasyFit?*, 2014, http://mathwave.com/blog/2011/10/28/what-parameter-estimation-methods-are-used-in-easyfit, [Accessed December 2014].

15. Myung, I. J., "Tutorial on Maximum Likelihood Estimation", *Journal of Mathematical Psychology*, Vol. 47, pp. 90–100, 2003.

16. Zheng, S., "Maximum Likelihood Estimation", *Lecture Notes for Statistical Theory Course, Missouri State Univ.*, 2014, http://people.missouristate.edu/songfengzheng/Teaching/MTH541/Lecture%20notes/MLE.pdf, [Accessed December 2014].

17. Kadane, J.B. and N.A. Lazar, "Methods and Criteria for Model Selection", *Journal of the American Statistical Association*, Vol. 99, No. 1, pp. 279-290, 2004.

18. Kuha, J., "AIC and BIC Comparisons of Assumptions and Performance", *Sociological Methods Research*, Vol. 33, No. 2, pp. 188-229, 2004.

19. Wagenmakers, E.J. and S. Farrell, "AIC Model Selection Using Akaike Weights", *Psychonomic Bulletin & Review*, Vol. 11, No. 1, pp. 192-196, 2004.

20. Arnold T.B. and J.W. Emerson, "Nonparametric Goodness of Fit Tests for Discrete Null Distributions", *The R Journal*, Vol. 3, No. 2, pp.34, 2011.

21. Murison B., "Properties of Maximum Likelihood Estimates", *Lecture Notes for Distribution Theory and Inference Course, Univ. of New England*, 2000 http://turing.une.edu.au/~stat354/notes/node69.html, [Accessed December 2014].

22. Nocedal, J., S.J. Wright, *Numerical Optimization*, 2nd edition, Springer, USA, 2006.

23. Wolfram, "Numerical Nonlinear Global Optimization", *Wolfram Matematica Tutorial*, Wolfram Research, Inc., 2008, http://www.wolfram.com/learningcenter/tutorialcollection/Constrained\,Optimization/ConstrainedOptimization.pdf, [Accessed December 2014].

24. Gratton, S., C.W. Royer, L.N. Vicente and Z. Zhang, "Direct Search Based on Probabilistic Descent", preprint 14-11, Dept. Mathematics, Univ. Coimbra, 2014, http://www.mat.uc.pt/~lnv/papers/ds-random.pdf, [Accessed December 2014].

25. Soofi, A.S. and L. Cao, *Modeling and Forecasting Financial Data: Techniques of Nonlinear Dynamics*, Springer, USA, 2002.

26. Hörmann, W.,"Statistical Inference", *Lecture Notes for Statistical Inference Course*, Boğaziçi University, 2010.

27. Mazerolle, M. J., "Appendix: Making Sense Out of Akaike's Information Criterion (AIC)", 2004, http://theses.ulaval.ca/archimede/fichiers/21842/apa.html, [Accessed December 2014].

28. Kozak, R.A., C.L. Staudhammer and S.B. Watts, "Poisson Distribution" *Introductory Probability and Statistics*, Cambridge University Press, UK, 2008.

29. Bird, J., "Poisson Distribution", *Understanding Engineering Mathemathics*, Routlegde, NY, 2014.

30. Mills, T.C., "The Binomial Distribution", *Analysing Economic Data: A Concise*

Introduction, Palgrave Macmillan, UK, 2014.

31. Wikipedia, *Negative Binomial Distribution*, 2014, http://en.wikipedia.org/wiki/Negative_binomial_distribution, [Accessed December 2014].

32. Wikipedia, *Uniform Distribution*, 2014, http://en.wikipedia.org/wiki/Probability_distribution, [Accessed December 2014].

33. The Univ. of Alabama Virtual Laboratories, *Logarithmic Distribution in Probability and Statistics*, 2014, http://www.math.uah.edu/stat/special/Logarithmic.html, [Accessed December 2014].

34. Wilson, R.J., "Logarithmic Series Distribution and Its Use in Analyzing Discrete Data", *Proceedings of the Survey Research Methods, American Statistical Association*, 1988, http://www.amstat.org/sections/SRMS/Proceedings/, [Accessed December 2014].

35. Princeton, "Normal Distribution", *Lecture Notes, Princeton University*, https://www.princeton.edu/~achaney/tmve/wiki100k/docs/Normal_distribution.html, 2014, [Accessed December 2014].

36. Casella G. and R.L. Berger, *Statistical Inference*, 2nd Edition, Duxbury, USA, 2002.

37. The Univ. of Alabama Virtual Laboratories, *The Pareto Distribution in Probability and Statistics*, 2014, http://www.math.uah.edu/stat/special/Pareto.html, [Accessed December 2014].

38. Kobayashi H., B.L. Mark and W. Turin, *Probability, Random Processes and Statistical Analysis*, Cambridge, UK, 2012.

39. Romeu, J.L., "Kolmogorov-Smirnov: A Goodness of Fit Test for Small Samples", *Reliability Analysis Center*, 2014, https://src.alionscience.com/pdf/K_STest.

pdf, [Accessed December 2014].

40. Horn, S.D, "Goodness-of-Fit Tests for Discrete Data: A Review and an Application to a Health Impairment Scale" *Biometrics*, Vol. 33, No. 1, pp. 237-247, 1977.

41. Narsky, I., "Goodness of Fit: What Do We Really Want to Know?", *Phystat2003, SLAC, Stanford, California*, pp. 8-11, 2003.

42. Engineering Statistics Handbook, *Chi-Square Goodness-of-Fit Test*, 2013, http://www.itl.nist.gov/div898/handbook/eda/section3/eda35f.htm, [Accessed December 2014].

43. Fisher, R. A., *Statistical Methods for Research Workers (7th ed.), an internet resource developed by Christopher D.*, Green York University, Toronto, Ontario, posted april 2000, http://psychclassics.yorku.ca/Fisher/Methods/chap4.htm, [Accessed December 2014].

44. Yearwood, V.Y, *Chi Square and the Problem of Small Frequencies*, 2008, http://isites.harvard.edu/fs/docs/icb.topic473170.files/Vincent_presentation.ppt, [Accessed December 2014].

45. Wikipedia, *Empirical Distribution*, 2014, http://en.wikipedia.org/wiki/Empirical_distribution_function, [Accessed December 2014].

46. VassarStats, *The Power of Chi-Square Goodness of Fit Test*, 2014, http://vassarstats.net/chi_beta.html, [Accessed December 2014].

47. Wikipedia, *Chi-Square Test*, 2014, http://en.wikipedia.org/wiki/Pearson's_chi-squared_test, [Accessed December 2014].