Universidade Federal de Pernambuco

Centro de Ciências Exatas e da Natureza

Programa de Pós-Graduação em Estatística

Luis Gustavo Bastos Pinho

Building new probability distributions: the composition method and a computer based method

Recife

2017

Universidade Federal de Pernambuco
Centro de Ciências Exatas e da Natureza
Programa de Pós-Graduação em Estatística

Building new probability distributions: the composition method and a computer based method

Trabalho apresentado ao Programa de Luis Gustavo Bastos Pinho do Programa de Pós-Graduação em Estatística da Universidade Federal de Pernambuco como requisito parcial para obtenção do grau de Doutor em Estatística.

Orientador: *Prof. Dr. Gauss Moutinho Cordeiro*
Co-orientador: *Prof. Dr. Juvêncio Santos Nobre*

Recife
2017

Catalogação na fonte
Bibliotecária Monick Raquel Silvestre da S. Portes, CRB4-1217

P654b Pinho, Luis Gustavo Bastos
Building new probability distributions: the composition method and a computer based method / Luis Gustavo Bastos Pinho. – 2017.
103 f.: il., fig., tab.

Orientador: Gauss Moutinho Cordeiro.
Tese (Doutorado) – Universidade Federal de Pernambuco. CCEN, Estatística, Recife, 2017.
Inclui referências e apêndices.

1. Estatística. 2. Probabilidade. I. Cordeiro, Gauss Moutinho (orientador). II. Título.

310 CDD (23. ed.) UFPE- MEI 2017-66

LUIS GUSTAVO BASTOS PINHO

BUILDING NEW PROBABILITY DISTRIBUTIONS: THE COMPOSITION METHOD AND A COMPUTER BASED METHOD

Tese apresentada ao Programa de Pós-Graduação em Estatística da Universidade Federal de Pernambuco, como requisito parcial para a obtenção do título de Doutor em Estatística.

Aprovada em: 17 de janeiro de 2017.

BANCA EXAMINADORA

Prof. Gauss Moutinho Cordeiro
UFPE

Prof. Getúlio José Amorim do Amaral
UFPE

Prof. Francisco Cribari Neto
UFPE

Prof. Rodrigo Bernardo da Silva
UFPB

Prof. Marcelo Bourguignon Pereira
UFRN

Now I know there's a better way
Let my heart ride out for a brighter day
Now it's time to breathe in the open air
With a mind so free, anyway
It's time for a change...
It's time to break free!

—TIME TO BREAK FREE (Gamma Ray)

ABSTRACT

We discuss the creation of new probability distributions for continuous data in two distinct approaches. The first one is, to our knowledge, novelty and consists of using Estimation of Distribution Algorithms (EDAs) to obtain new cumulative distribution functions. This class of algorithms work as follows. A population of solutions for a given problem is randomly selected from a space of candidates, which may contain candidates that are not feasible solutions to the problem. The selection occurs by following a set of probability rules that, initially, assign a uniform distribution to the space of candidates. Each individual is ranked by a fitness criterion. A fraction of the most fit individuals is selected and the probability rules are then adjusted to increase the likelihood of obtaining solutions similar to the most fit in the current population. The algorithm iterates until the set of probability rules are able to provide good solutions to the problem. In our proposal, the algorithm is used to generate cumulative distribution functions to model a given continuous data set. We tried to keep the mathematical expressions of the new functions as simple as possible. The results were satisfactory. We compared the models provided by the algorithm to the ones in already published papers. In every situation, the models proposed by the algorithms had advantages over the ones already published. The main advantage is the relative simplicity of the mathematical expressions obtained. Still in the context of computational tools and algorithms, we show the performance of simple neural networks as a method for parameter estimation in probability distributions. The motivation for this was the need to solve a large number of non linear equations when dealing with SAR images (SAR stands for synthetic aperture radar) in the statistical treatment of such images. The estimation process requires solving, iteratively, a non-linear equation. This is repeated for every pixel and an image usually consists of a large number of pixels. We trained a neural network to approximate an estimator for the parameter of interest. Once trained, the network can be fed the data and it will return an estimate of the parameter of interest without the need of iterative

methods. The training of the network can take place even before collecting the data from the radar. The method was tested on simulated and real data sets with satisfactory results. The same method can be applied to different distributions. The second part of this thesis shows two new probability distribution classes obtained from the composition of already existing ones. In each situation, we present the new class and general results such as power series expansions for the probability density functions, expressions for the moments, entropy and alike. The first class is obtained from the composition of the beta-G and Lehmann-type II classes. The second class, from the transmuted-G and Marshall-Olkin-G classes. Distributions in these classes are compared to already existing ones as a way to illustrate the performance of applications to real data sets.

Keywords: New probability distributions. G-classes. Estimation of distribution algorithms. Estimation.

RESUMO

Discutimos a criação de novas distribuições de probabilidade para dados contínuos em duas abordagens distintas. A primeira é, ao nosso conhecimento, inédita e consiste em utilizar algoritmos de estimação de distribuição para a obtenção de novas funções de distribuição acumulada. Algoritmos de estimação de distribuição funcionam da seguinte forma. Uma população de soluções para um determinado problema é extraída aleatoriamente de um conjunto que denominamos espaço de candidatos, o qual pode possuir candidatos que não são soluções viáveis para o problema. A extração ocorre de acordo com um conjunto de regras de probabilidade, as quais inicialmente atribuem uma distribuição uniforme ao espaço de candidatos. Cada indivíduo na população é classificado de acordo com um critério de desempenho. Uma porção dos indivíduos com melhor desempenho é escolhida e o conjunto de regras é adaptado para aumentar a probabilidade de obter soluções similares aos melhores indivíduos da população atual. O processo é repetido por um número de gerações até que a distribuição de probabilidade das soluções sorteadas forneça soluções boas o suficiente. Em nossa aplicação, o problema consiste em obter uma função de distribuição acumulada para um conjunto de dados contínuos qualquer. Tentamos, durante o processo, manter as expressões matemáticas das distribuições geradas as mais simples possíveis. Os resultados foram satisfatórios. Comparamos os modelos providos pelo algoritmo a modelos publicados em outros artigos. Em todas as situações, os modelos obtidos pelo algoritmo apresentaram vantagens sobre os modelos dos artigos publicados. A principal vantagem é a expressão matemática reduzida. Ainda no contexto do uso de ferramentas computacionais e algoritmos, mostramos como utilizar redes neurais simples para a estimação de parâmetros em distribuições de probabilidade. A motivação para tal aplicação foi a necessidade de resolver iterativamente um grande número de equações não lineares no tratamento estatístico de imagens obtidas de SARs (synthetic aperture radar). O processo de estimação requer a solução de uma equação por métodos iterativos e isso é repetido para

cada pixel na imagem. Cada imagem possui um grande número de pixels, em geral. Pensando nisso, treinamos uma rede neural para aproximar o estimador para esse parâmetro. Uma vez treinada, a rede é alimentada com as janelas referente a cada pixel e retorna uma estimativa do parâmetro, sem a necessidade de métodos iterativos. O treino ocorre antes mesmo da obtenção dos dados do radar. O método foi testado em conjuntos de dados reais e fictícios com ótimos resultados. O mesmo método pode ser aplicado a outras distribuições. A segunda parte da tese exibe duas classes de distribuições de probabilidade obtidas a partir da composição de classes existentes. Em cada caso, apresentamos a nova classe e resultados gerais tais como expansões em série de potência para a função densidade de probabilidade, expressões para momentos, entropias e similares. A primeira classe é a composição das classes beta-G e Lehmann-tipo II. A segunda classe é obtida a partir das classes transmuted-G e Marshall-Olkin-G. Distribuições pertencentes a essas classes são comparadas a outras já existentes como maneira de ilustrar o desempenho em aplicações a dados reais.
Palavras-chave: Novas distribuições de probabilidade. Classe-G. Algoritmos de estimação de distribuição. Estimação.

List of Figures

2.1 Tree representation of the normal distribution pdf. 22
2.2 Histogram and quantile plot for the Wheaton river data set. 33
2.3 Histogram and quantile plot for the ball bearings data set. 34

3.1 Some shapes of the Beta-L2-Gumbel pdf for selected parameter values 42
3.2 Some shapes of the Beta-L2-Log-logistic pdf for selected parameter values 43
3.3 Some shapes of the Beta-L2-Fréchet pdf for selected parameter values 44
3.4 Three of the fitted densities. 49

4.1 QTMO-Weibull pdfs and hrfs. 57
4.2 Galton's skewness and Moor's kurtosis for a QTMO-Weibull family. 58
4.3 QTMO-log-logistic pdfs and hrfs with $\beta = 1$. 58
4.4 Galton's skewness and Moor's kurtosis for a QTMO-log-logistic family. 59
4.5 QTMO-standard normal pdfs and hrfs. 59
4.6 Galton's skewness and Moor's kurtosis for the QTMO-standard normal. 60
4.7 Some possible shapes for the pdf and hrf of the QTMO-Kumaraswamy family. . . . 61
4.8 Galton's skewness and Moor's kurtosis for a QTMO-Kumaraswamy. 61
4.9 Histogram and box-plot for the coverage data. 62
4.10 Histogram and fitted model. Quantile plot for the QTMO-KW distribution. 63

5.1 A general MLP diagram. 69
5.2 Results for the exponential distribution. 73
5.3 Results for the estimation of normal distribution's mean. 74
5.4 Results for the estimation of normal distribution's standard deviation. 75

5.5 Results for the estimation of the shape parameter in the exponentiated exponential distribution. 76
5.6 Comparison of the shape parameter estimation results in the exponentiated exponential distribution. 77
5.7 SAR image of an oil slick. 78
5.8 Comparison between the MLP and other moment based methods. 79
5.9 Segmented SAR image of an oil slick. 80

List of Tables

2.1 Results support the PIPE method distribution 29
2.2 Estimates and log-likelihood for the models adjusted to the Wheaton river data set. . 32
2.3 Estimation for the ball bearings data set. 34

3.1 Parameter estimates and relative selection criteria for the current example. 48
3.2 Relative selection criteria for the distributions in Fischer and Vaughan (2010). 48

4.1 Parameter estimates and relative selection criteria for the forest coverage example. . 64

5.1 Summary for the errors in the normal distribution estimation. 72
5.2 Summary for the errors in the normal distribution estimation. 76

CONTENTS

1 Introduction and overview **14**

2 Continuous Probability Distributions Generated by the PIPE Algorithm **19**

2.1 Introduction 20

2.2 The PIPE algorithm 21

2.3 The new proposal 25

2.4 Simulation studies 27

2.5 Real data application and comparisons 30

2.6 Addressing issues and final comments 35

3 Beta L2 Model **37**

3.1 Introduction 37

3.2 The Beta-L2-G class 38

3.3 General properties of the new family 39

3.4 Some members of the Beta-L2-G class 41

3.4.1 Beta-L2-Gumbel 41

3.4.2 Beta-L2-Log-logistic 43

3.4.3 Beta-L2-Fréchet 44

3.5 Estimation 44

3.6 Application to real data 46

3.7 Concluding remarks 50

4 The Quadratic Transform-Marshal-Olkin-G class of distributions **51**

4.1 Introduction 51

4.2 Transmuted distributions 52
4.3 The Marshall-Olkin class of distributions 53
4.4 The new class of distributions 54
4.5 Some families in the QTMO-G class 56
4.5.1 QTMO-Weibull 56
4.5.2 QTMO-log-logistic 56
4.5.3 QTMO-normal 58
4.5.4 QTMO-Kumaraswamy 60
4.6 Estimation 60
4.7 Application to a real data set 62
4.8 Concluding remarks 64

5 Estimation procedures using multilayer perceptrons for univariate models 66
5.1 Introduction 66
5.2 The generalized method of the moments 68
5.3 Multilayer perceptron (MLP) neural networks 68
5.4 Simulation 70
5.4.1 Exponential distribution 71
5.4.2 Normal distribution 72
5.4.3 Exponentiated exponential distribution 72
5.5 SAR image segmentation - real data application 77
5.6 Final remarks 80

References 81

CHAPTER 1

Introduction and overview

RESUMO

Nesse capítulo apresentamos uma visão geral da tese. Discutimos o desenvolvimento de novas distribuições de probabilidade para dados contínuos. É feita uma breve revisão dos métodos mais conhecidos, que consistem da composição de funções de distribuição acumuladas já existentes, onde discutimos suas vantagens e desvantagens. A contribuição principal dessa tese é a apresentação de um método que permite obter novas distribuições através de algoritmos evolutivos. Essa abordagem trata, de certa forma, o problema de seleção de modelos. Também mostramos como o texto está dividido.

In this work, we present two contributions to the field of generated probability distributions. One is based on currently popular techniques and the other is based on an evolutionary algorithm.

The number of new probability distributions greatly increased in the last few decades. This is partially due to the fact that it is possible to compound two different distributions in order to obtain a third one. Many distributions obtained in this way adapt nicely to various data sets as seen in the literature. This subject was reviewed to a great extent recently in Tahir and Nadarajah (2015) and Tahir and Cordeiro (2016). Each used technique defines a usually very large class of probability distributions. For each class, listed several papers on particular families in that class. The authors used the term *G-classes* to refer to distributions generated from such compositions. Some G-classes of greater popularity are now presented.

Exponentiated-G (Exp-G)

This method is attributed to Lehmann (1953) and adds one parameter to an existing distri-

bution. If $G(x)$ is a cummulative distribution function (cdf), then

$$F(x) = G(x)^{\alpha},\ \alpha > 0$$

is also a cdf and it is called the exponentiated-G (Exp-G) distribution with power parameter α. The distribution G is called the baseline distribution in this and in similar contexts (other G-classes). This class is also called Lehmann type I distribution. The Lehmann type II distribution is given by

$$F(x) = 1 - [1 - G(x)]^{\alpha},\ \alpha > 0.$$

These distributions became much more popular after the papers by Gupta et al. (1998) and Gupta and Kundu (1999, 2001, 2002) on the exponentiated-exponential distribution. The exponentiated-Weibull distribution was presented in Mudholkar and Srivastava (1993) and is considered, generally speaking, a very good distribution with many papers discussing it, such as Mudholkar et al. (1995), Mudholkar and Hudson (1996), Choudhury (2005), Singh et al. (2005), Qian (2012) and Barrios and Dios (2012) among many others.

Beta-G

If $G(x)$ is a cdf, then

$$F(x) = I_{G(x)}(a,b) = \frac{1}{B(a,b)} \int_0^{G(x)} t^{a-1}(1-t)^{b-1} dt,\ a,b > 0,$$

is the beta-G cdf, where $B(a,b) = \int_0^{\infty} t^{a-1}(1-t)^{b-1} dt$ represents the beta function and

$$I_x(a,b) = \frac{1}{B(a,b)} \int_0^{x} t^{a-1}(1-t)^{b-1} dt,$$

represents the incomplete beta function ratio. The pdf of the beta-G distribution is given by

$$f(x) = \frac{1}{B(a,b)} g(x) G(x)^{a-1} [1 - G(x)]^{b-1},$$

in which $g(x)$ is the pdf respective to $G(x)$. The beta-normal distribution is the first known beta-G member and was proposed in Eugene et al. (2002). This class was studied extensively by Jones (2004), who showed that this class arises from the distributions of order statistics. The beta-G distributions can have heavy tails and assimetry.

It is possible to write its pdf as a mixture of exp-G distributions (with the same baseline distribution). By using the binomial theorem, we can write

$$f(x) = \sum_{i=0}^{\infty} w_i (a+i) g(x) G(x)^{a+i-1},$$

where

$$w_i = w_i(a,b) = \frac{(-1)^i \binom{b-1}{i}}{(a+i)B(a,b)}.$$

This allows us to express the moments of a beta-G random variable as a mixture of moments of exp-G random variables, which are usually easier to obtain. Several moments from beta-G random variables are given in Cordeiro and Nadarajah (2011) by means of probability weighted moments (PWMs) of $G(x)$. The (u,v) order PWM of a random variable $Y \sim G$ is given by $\tau_{u,v} = \mathbb{E}\{Y^u G(Y)^v\}$. In Zografos and Balakrishnan (2009), this class is discussed and methods for estimation and the characterization by maximum entropy are proposed.

Gamma-G

This class was proposed in Zografos and Balakrishnan (2009). If $G(x)$ is a cdf with survival function $\bar{G}(x) = 1 - G(x)$ and pdf $g(x)$, the gamma-G cdf is given by

$$F(x) = \frac{1}{\Gamma(\delta)} \int_0^{-\log \bar{G}(x)} t^{\delta-1} \mathrm{e}^{-t} dt, \ \delta > 0,$$

and its pdf is

$$f(x) = \frac{1}{\Gamma(\delta)} \left[-\log \bar{G}(x)\right]^{\delta-1} g(x).$$

Ristić and Balakrishnan (2012) proposed a slightly different generator given by

$$F(x) = 1 - \frac{1}{\Gamma(\delta)} \int_0^{-\log G(x)} t^{\delta-1} \mathrm{e}^{-t} dt, \ \delta > 0,$$

with pdf

$$f(x) = \frac{1}{\Gamma(\delta)} \left[-\log F(x)\right]^{\delta-1} g(x),$$

It is also possible to express the gamma-G pdf as a mixture of exp-G densities. This was given by Nadarajah, Cordeiro and Ortega (2015).

There are many other classes and several of them are given by Tahir and Nadarajah (2015).

A very general approach for generating distributions is given in Alzatreeh et al. (2015). In their paper, they provide properties for the G-classes obtained by

$$F(x) = \int_0^{W(G(x))} r(x)dx,$$

where G is the baseline distribution, $r(x)$ is a continuous pdf and $W(G(X))$ matches the support of $r(x)$. Aly and Benkherouf (2011) discussed the composition of continuous and discrete distribution and showed very general properties of the generated classes.

There are advantages to the composition methods. The new parameters tend to provide better fits of the G-classes distributions as the probability density functions can assume different shapes. The mathematical properties of the G-classes distributions are usually not very difficult to obtain and in many cases they follow from writing the new densities as mixtures of exp-G pdfs. This has been done in many papers. Some of the parameters in G-classes even have physical interpretations. There is, however, a trade-off in the use of such new models. They usually have many parameters and estimating them is usually troublesome. The maximum likelihood method will usually suffer from having very spiky or very flat areas in the "log-likelihood" function. This may lead to numerical difficulties, large standard errors and overall poor identifiability of the model. Also, for some applications in signal processing and other areas of engineering, complex expressions for the distributions' cdf and pdf may be a problem. Another problem comes from the increase in the number of distributions. Model selection is now considerably harder than before.

Because of these issues with the G-classes, we propose in this text a different method for generating new distributions for continuous data. This method is based on the *Probabilistic Incremental Program Evolution* (PIPE) algorithm proposed by Salustowicz and Schmidhuber (1997). Using this algorithm we are able to obtain new continuous distributions that are generated specifically for each one of the data sets considered, while obtaining relatively simple cdfs and pdfs. This approach deals with model selection and model estimation at the same time and yields very satisfactory results. This is the main contribution of this text.

This text is divided as follows. In Chapter 2, we present a new method for generating continuous probability distributions. We also provide real data application for some data sets. In Chapter 3, we present a new class of distributions obtained by the composition of the beta-G

and Lehmann type II classes. Chapter 4 presents another class of distributions. This one is obtained from the composition of the Transmuted-G (Shawn and Buckley, 2007) and Marshall-Olkin-G (Marshall and Olkin, 1997). A new procedure for parameter estimation is presented in Chapter 5. This procedure is based on a certain neural network and makes the task of parameter estimation in some extended classes both easier and faster.

CHAPTER 2

Continuous Probability Distributions Generated by the PIPE Algorithm

RESUMO

Nesse capítulo, investigamos o uso do algoritmo *Probabilistic Incremental Programming Evolution* (PIPE) como uma ferramenta para a construção de funções de distribuição acumuladas para modelar determinados conjuntos de dados. O algoritmo, em nossas aplicações, gera várias funções candidatas a modelar os conjuntos de dados considerados. Essas candidatas são geradas seguindo um conjunto de regras estocásticas. O conjunto de regras então evolui ao longo de iterações para gerar candidatas melhores de acordo com um dado critério de otimalidade. Essa abordagem compete com a adição de parâmetros a distribuições existentes. Há duas vantagens principais em utilizar o novo método. A primeira é a possibilidade de controlar explicitamente a complexidade das funções condidatas, especificando quais funções e operadores matemáticos podem ser utilizados e quão longa pode ser a expressão matemática correspondente. A segunda vantagem é que essa abordagem trata seleção de modelo e estimação ao mesmo tempo. Esse método é apropriado para situações em que não é possível deduzir um modelo de probabilidade diretamente das características do fenômeno estudado. Essa também é uma alternativa aos métodos não-paramétricos clássicos. A vantagem é que utilizando o PIPE, em geral, podemos obter expressões mais tratáveis. O desempenho em dados artificiais e dados reais é bastante satisfatório. Para aplicações em dados reais, o algoritmo obteve verossimilhanças melhores ou comparáveis ao modelos já utilizados na literatura, mas com expressões matemáticas notoriamente mais simples.

2.1 Introduction

In this chapter, we describe an approach to probability modeling based on evolutionary algorithms. Consider a set of observations of a continuous random variable representing the outcome of a given experiment. Suppose that there is not enough understanding of the problem to allow for the construction of a specific cdf or to suggest the use of an existing one. Given such a data set we attempt to find the "best possible" continuous probability distribution to model the data regarding some fitness or optimality criteria. This is achieved by exploring a search space of candidate functions in a way described in the next sections.

The method presented here is an alternative to the G-classes of distributions as well as to classical non-parametric methods. It has two main motivations: 1) there is an increasing difficulty in choosing a model for a given data set because of the increase in the number of new distributions; 2) most of the new models use formulas that are expressed with elementary functions such as logarithms and exponentials and arithmetic operators. Given a data set and an optimality criterion, our goal is to obtain a suitable cdf for the data using only these elementary functions and arithmetic operators for at most a certain number of times. This may lead to models with simpler mathematical expressions. Thus, we consider model selection and model generation in a single approach. The main advantage of this method over the other two commented above is that we are able to explicitly control the complexity of the resulting cdf. In this text we use the word complexity in its basic meaning. A function is complex if its mathematical expression is lengthy or uses advanced special functions like gamma, beta, Bessel functions and alike. By having a simple expression for the cdf, we can easily obtain quantities related to the data, such as moments, for example. This chapter is divided as follows. In section 2, we describe the *Probabilistic Incremental Program Evolution* (PIPE) algorithm developed by Salustowicz and Schmidhuber (1997), which is the base technique employed in this proposal. It is slightly modified for our purpose. Section 3 describes and exemplifies the new proposal. Simulation studies are presented in Section 4 to illustrate the performance of the method regarding a mean quadratic error between the proposed function and the empirical distribution function at the data points. In Section 5, we consider data sets from papers on

recent distributions, comparing the fitness of the distributions in those papers to the fitness of the distribution obtained by the method proposed here. Issues and final remarks are addressed in Section 6. The source code for the implementations of the described method are available in the Appendix. Instructions of how to use and examples are also included.

2.2 The PIPE algorithm

The PIPE algorithm was presented in Salustowicz and Schmidhuber (1997) and it is capable of producing programs according to a set of probability rules. These rules are improved over iterations so that the generated programs are more likely to solve a given problem. In this section we follow closely the explanation in the original paper.

A *program* is a set of instructions given in a certain order. Each of these instructions may depend on a (possibly empty) set of terminal symbols, which usually denote constants or user inputs. Let $F = \{f_1, f_2, \ldots, f_k\}$ be a set of k functions and $T = \{t_1, t_2, \ldots, t_l\}$ be a set of l terminals. For instance, to write a program that calculates the value of the probability density function (pdf) for the normal or exponential distributions at a point x and a given set of parameters, it is sufficient to take $F = \{-, \times, \div, \exp, \sqrt{}\}$ and $T = \{x, \pi, 2, 1, -1, R, R^+\}$, where $\div$ the protected division (does not allow division by zero), x represents an user input, R represents a real constant and R^+ represents a positive real constant. The normal distribution pdf can be described as

$$\left(1 \div (\sqrt{2 * \pi} * R^+) * \exp((-1 \div (2 * R^+ * R^+)) * (x - R) * (x - R)\right),$$

for example.

Each program can be represented by an n-ary tree, where n is the maximum possible of arguments for a function in F. For the normal distribution example we may use the tree in Figure 2.1. The tree representing a program is not unique unless we specify a set of rules for parsing a program to a tree, however this is negligeable for our purpose.

Programs can be created randomly by traversing a structure called *Probabilistic Prototype*

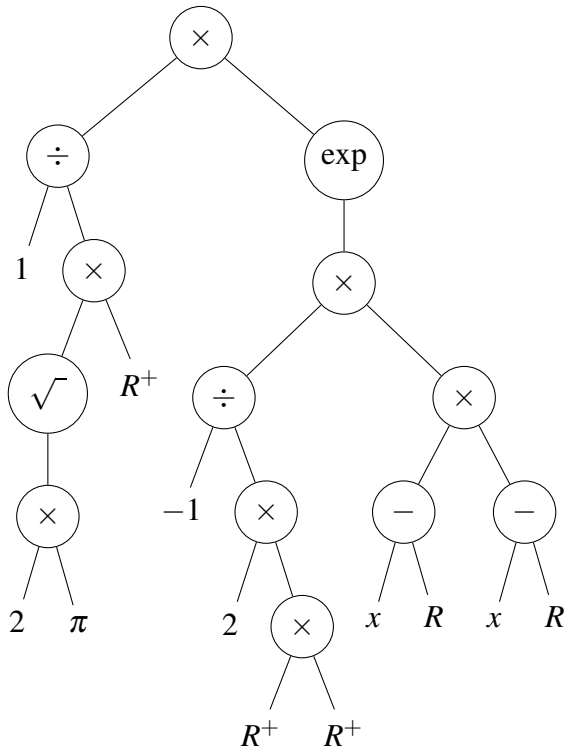


Figure 2.1 Tree representation of the normal distribution pdf.

Tree (PPT). The PPT is a n-ary tree with n, again, representing the maximum arity of an instruction in F. The node at depth $d \geq 0$ and horizontal position $w \geq 0$ (width) is represented by $N_{d,w}$. Each node contains a probability vector $\mathbf{P}_{d,w}$ whose entries are $\mathbf{P}_{d,w}(I)$ for each $I \in F \cup T$ such that

$$\sum_{I \in F \cup T} P_{d,w}(I) = 1, \ \forall N_{d,w}.$$

That is, each node has the probability distribution of the possible instructions in the programs at the respective node of their tree representation. The PPT is traversed in a depth first fashion from left to right, starting at $N_{0,0}$. For each accessed node, an instruction I is selected with probability $P_{d,w}(I)$ and denoted $I_{d,w}$. If $I_{d,w} \in F$, then a subtree is created for each argument of $I_{d,w}$. If $I_{d,w} \in T$ then it is replaced by an instance $V_{d,w}(I_{d,w})$ of that terminal. This instance equals $I_{d,w}$ if $P_{d,w}(I_{d,w})$ is greater than a certain threshold T_I and equals a randomly generated number $R_{d,w}$ otherwise. For each terminal instruction $I \in T$ there corresponds a threshold T_I and these are not changed throughout the iterations. The authors in Salustowicz and Schmidhuber

(1997) also consider the *growing* and *pruning* of the PPT to reduce the memory requirements of the algorithm. Initially there is only the node $N_{0,0}$. If $I_{d,w} \in F$ is chosen and the subtree for its arguments are missing in the PPT, then additional nodes are created (growing). Conversely, if the probability of accessing a certain node in the PPT is too small, the node is deleted from the PPT (pruning).

PIPE has two learning mechanics: elitist learning and generation-based learning. These two mechanics alternate until a stopping criterion is met. Generation-based learning comprises five distinct phases.

1. **Creation of a program population.** A population of programs is created according to the rules mentioned earlier. These programs are enumerated as PROG_j, $0 < j \leq PS$, with PS denoting the population size. Probabilities in each node are initialized in a random way but maintaining their sum equal to 1.

2. **Population evaluation.** Each PROG_j in the population is evaluated regarding a certain fitness function. This is a numeric value assigned by a function $\text{FIT}(\text{PROG}_j)$. The programs are ranked in ascending order of those values. The best program in the current population is denoted PROG_b while the best program found so far is denoted PROG^{el}.

3. **Learning from the population.** The probabilities in each node of the PPT are modified as to increase the likelihood of PROG_b being generated. The following steps can be stored as the content of an `adaptPPTtowards`(PROG_b) routine at the time of the implementation by the reader. First the probability $P(\text{PROG}_b)$ is obtained as $\prod P_{d,w}(I_{d,w})$ for each instruction $I_{d,w}$ used in the production of the candidate PROG_b. A target probability is calculated as

$$P_{\text{TARGET}} = P(\text{PROG}_b) + lr[1 - P(\text{PROG}_b)] \times \frac{\varepsilon + \text{FIT}(\text{PROG}^{el})}{\varepsilon + \text{FIT}(\text{PROG}_b)},$$

in which the constant lr denotes the learning rate of the algorithm, and ε is a user-defined positive real constant. The fraction in the right hand side of the equation implements the

fitness-dependent-learning (fdl). If ε is chosen such that $\varepsilon \ll \text{FIT}(\text{PROG}^{el})$ then generations with lower quality (higher fitness values) programs do not influenciate much the learning process, allowing for the use of smaller populations. Once P_{TARGET} is obtained, all the probabilities $P_{d,w}(I_{d,w})$ for the instructions used in PROG_b are increased iteractively as seen in algorithm 1, where c^{lr} denotes a constant that influences the number of iterations and the precision. The choice of this constant is subjective. Lower values will imply more iterations and more precision while higher values will do the opposite. Then, each terminal used in the construction of PROG_b is stored in the respective node of the PPT, that is, $\text{I}_{d,w} := \text{V}_{d,w}(\text{I}_{d,w})$ for each terminal instruction $\text{I}_{d,w}$ used in PROG_b.

4. **Mutation of the PPT.** In this step, the nodes acessed during the production of PROG_b are mutated with a probability P_{M_P} given by

$$P_{M_P} = \frac{P_M}{(l+k)\sqrt{|\text{PROG}_b|}},$$

where P_M is a user defined parameter controlling the overall probability of mutation. The previous formula is empirically justified in Salustowicz and Schmidhuber (1997). If a node is to be mutated, the probability $P_{d,w}(\text{I}_{d,w})$ is changed to $P_{d,w}(\text{I}_{d,w}) + mr \cdot (1 - P_{d,w}(\text{I}_{d,w}))$, in which mr represents a mutation rate. Notice that this change is small if $P_{d,w}(\text{I}_{d,w})$ is already large. After the mutation step every modified node is normalized to keep the sum of probabilities equal to 1.

5. **PPT pruning.** If $P_{d,w}(\text{I}_{d,w})$ becomes too small for a certain node $\text{N}_{d,w}$ and instruction $\text{I}_{d,w} \in F$ then the subtrees corresponding to the possible arguments of $\text{I}_{d,w}$ are deleted from the PPT.

Algorithm 1: Updating the PPT.

```
repeat
    forall I_{d,w} in PROG_b do
        P_{d,w}(I_{d,w}) := P_{d,w}(I_{d,w}) + c^{lr} · lr · (1 − P_{d,w}(I_{d,w}));
until P(PROG_b)>P_TARGET;
```

After the generation-based learning, elitist learning takes place by repeating the previous procedure using PROG^{el} instead of PROG_b. However, during the elitist learning mutation is not performed. The PPT is then pruned accordingly.

2.3 The new proposal

Given a data set, we propose the use of the PIPE method, with minor modifications, to generate a continuous function that resembles the empirical cumulative distribution of the data. We argue that, for a suitable choice of the sets F and T, it is possible to achieve a good fit of the data while controlling the complexity of the model by limiting the maximum height of the PPT. The minor modifications considered are:

1) During the generation of a program, if a node $\text{N}_{d,w}$ is to receive an instruction $\text{I}_{d,w} \in T$ representing an user input variable (a data point), we set the respective threshold to 1.

2) The maximum size of the program, measured by the height of the tree representing it, is controlled. Programs are not allowed to grow indeterminately. To achieve this, we modify the nodes at the maximum depth of the PPT by forcing $\text{P}_{d,w}(\text{I}) = 0$ for every $\text{I} \in \text{F}$ and normalizing the distribution $\mathbf{P}_{d,w}$ again.

3) No pruning or growing is performed, since memory consumption for small PPTs is not of concern in this context.

4) At each generation we randomly choose to adapt the PPT towards PROG_{el} or towards PROG_b. Mutation occurs regardless of the choice.

These modifications simplified the code while yielding good results in our investigations. Also, they help to incorporate some aspects of the problem at hand. In this paper we choose the functions and terminal sets to be $F = \{\times, \div, +, \text{pow}(\cdot,\cdot), \exp(-\cdot), \log(\cdot)\}$ and $I = \{x, R\}$, where $\div$ represents the protected division, a^b is represented by $\text{pow}(a,b)$, $\exp(-x)$ is the usual e^{-x}, $\log(x)$ is the protected logarithm (does not accept negative arguments), x represents an user

input and R represents a random number between 0 and a specified maximum value. This is the set of functions and terminals we use throughout the paper unless stated otherwise. The fitness function is given by:

$$\mathrm{FIT(PROG)} = -\frac{1}{n}\sum_{i=1}^{n}(\mathrm{PROG(x_i)} - \widehat{F}_n(x_i))^2,$$

where x_i is ith observation of the data set, n is size of the data set, and $\widehat{F}_n$ is the empirical distribution function. Our implementation will run for a number of generations and then stop. The overall sequence of an implementation for the PIPE algorithm is as follows.

- Set the following parameters: the elements in F∪T, size of the program population, number of generations, maximum size of the programs, learning rate, ε, mutation rate, maximum value of a generated random number and the probability of adapting the PPT towards the elite program.
- Initialize the PPT and assign probabilities to the vectors in every node. Let the PPT be as high as the maximum size of a program. Remember to set the probability of non-terminal symbols to zero for the leaf nodes.
- Read the data and obtain the empirical distribution function.
- For every generation:
  - Generate a random population of functions from the PPT.
  - Evaluate the fitness of every function.
  - Find the best program in the current generation and obtain PROB(PROG$_b$). Replace PROG^{el} if needed.
  - Adapt the PPT towards PROG$_b$ or PROG^{el} randomly.
  - Mutate the PPT, if it was adapted towards PROG$_b$.
  - Normalize $\mathrm{P}_{d,w}$ for every node of the PPT.

2.4 Simulation studies

In this section we investigate how well the models proposed by the PIPE algorithm fit the data when compared to the true distribution of the data. For several distributions with cdf F and several sample sizes n, we generate an artificial data set of size n from F and use the proposed method to generate a cdf for the data. We compare the proposed distribution to the distribution F by means of the logarithm of the likelihood function based on the data (log-likelihood). The Anderson-Darling and Kolmogorov-Smirnov tests (see, for example, Stephens 1974) are also performed to verify the adequacy of the model to the data. The simulation can be described as follows.

- Generate a data set S from F.
- Estimate the parameters of F using the data and the maximum likelihood method.
- Run the proposed method and obtain a cdf G.
- Perform both the Anderson-Darling and Kolmogorov-Smirnov tests, at the 10% level of significance. This level choice is arbitrary.
- Calculate the log-likelihood of G based on the data and compare to that of F.

Results of the simulation are presented in Table 2.1. The Kolmogorov-Smirnov (KS) test considers the overall fit of the distribution, whereas the Anderson-Darling (AD) test is built to emphasize the fit in the tails of the distribution (Stephens, 1974). The columns of Table 2.1 represent, respectively, the sample size, the distribution from which the data was simulated, the logarithm of the maximized likelihood for the model in the previous column with its parameters being estimated by the maximum likelihood method, the logarithm of the likelihood of the PIPE model and the p-values for the KS and AD tests. Criteria such as the Akaike information criterion (AIC) and Bayesian information criterion (BIC) are not used since we do not have a solid understanding of how would the comparison apply to the PIPE generated distributions. For all of the simulation runs, the parameters of the PIPE algorithm were set as follows: 100 candidates per generation, 3000 generations, $T_I = 0.2$ for all terminal numbers, 100 as the

maximum random number generated, height of the PPT set to 3, learning rate of 0.001, $c^{lr} = 0.01$, $\varepsilon = 0.01$, $P_M = 0.1$, mutation rate of 0.1 and the probability of adapting the PPT towards the elite program was set to 0.5. These values were chosen after a few trials with different values. The algorithm is not much sensitive to those values, except for the learning rate. Keep in mind that the learning rate represents a trade-off between speed and quality of the search for good candidates. The column LL represents the logarithm of the maximized likelihood function.

We chose the normal distribution because it is widely used in practical situations, the exponential and gamma distributions for being popular lifetime distributions, the Cauchy distribution for its heavy tails, a skew-*t* distribution (Fernandez and Steel 1998) for the heavy tails and skewness, the Pareto distribution for its shift, the beta distribution for being confined in the $(0,1)$ interval and a mixture of two normal distributions for bimodal data. Certainly it was not expected that the PIPE distributions would outperform the distributions that originated the data, even though it happened for a few runs. During this simulation we noticed that the data generated from the normal, exponential, gamma, normal mixture, skew-t and Pareto distributions were easily modeled by the proposed distributions from the PIPE algorithm. One run of the algorithm was most of the times enough to obtain a distribution that would achieve a log-likelihood close to or better than the original model. We ran the algorithm at most five times with different seeds for the random number generation and presented the best results in here. The Cauchy distribution proved to be the more challenging distribution. Better levels of log-likelihood than those shown in Table 2.1 for the Cauchy distribution were achieved but at the expense of failing the AD test at the 0.1 significance level. Solutions for this problem would possibly include the addition or removal of elements in the F set or the possibility of applying different weights to the data points in the tails of the data set when calculating the fitness values, if the user knows beforehand that the data supplied is heavy tailed. We noticed no increasing nor decreasing of the performance of the algorithm regarding the sample size.

Table 2.1 Results support the PIPE method distribution

		LL		p-value	
n	Distribution	Fitted	PIPE	KS	AD
10	Normal(0,1)	−9.41	−11.34	0.5360	0.6047
	Cauchy(0,1)	−22.60	−21.77	0.8633	0.6579
	Exponential(1)	−15.62	−14.55	0.8282	0.6536
	Gamma(3,1)	−19.94	−18.68	0.6581	0.9247
	Beta(3,5)	6.20	5.30	0.9448	0.8216
	Pareto (1,3)	−9.99	−9.28	0.8101	0.8564
	MN(0,5,1,1,0.25)	−12.14	−12.80	0.7953	0.9348
	Skew-t(2,2)	−23.90	−22.94	0.9585	0.9699
30	Normal(0,1)	−49.13	−49.16	0.8098	0.6490
	Cauchy(0,1)	−77.36	−93.89	0.2368	0.0111
	Exponential(1)	−15.61	−32.99	0.1538	0.2048
	Gamma(3,1)	−59.53	−61.58	0.6300	0.6858
	Beta(3,5)	18.96	17.09	0.8287	0.7421
	Pareto(1,3)	−81.87	−83.96	0.8036	0.8051
	MN(0,5,1,1,0.25)	−36.13	−40.15	0.6238	0.6667
	Skew-t(2,2)	−91.50	−79.11	0.4857	0.2997
50	Normal(0,1)	−70.32	−70.35	0.4798	0.2208
	Cauchy(0,1)	−114.37	−133.22	0.7690	0.5053
	Exponential(1)	−56.80	−60.99	0.7635	0.6453
	Gamma(3,1)	−92.53	−94.34	0.7588	0.7505
	Beta(3,5)	24.16	24.33	0.8891	0.9453
	Pareto(1,3)	−152.28	−152.32	0.9630	0.9471
	MN(0,5,1,1,0.25)	−59.16	−65.44	0.7566	0.5720
	Skew-$t(2,2)$	−104.19	−97.11	0.3121	0.1469

2.5 Real data application and comparisons

In this section, we apply the PIPE method to a pair of data sets that were modeled in already published papers on probability distribution families. For each re-visited paper we compare the likelihood of the distributions proposed in that paper to the one suggested by the PIPE method. To verify the adequacy of the model proposed by the PIPE method to the data we resort to the KS and AD tests, the visual inspection of the theoretical quantiles plotted against the empirical quantiles and the plot of the probability density function overlapping the data histogram. To the quantile plot we add a simulated 95% envelope for the empirical quantiles obtained by bootstrapping the original sample. This is done by generating a large number B of pseudo-samples by sampling, with replacement, the original data set. Order each pseudo-sample to obtain its empirical quantiles. There will be, after the simulation, B samples of each quantile. For each quantile q find the quantiles 2.5% and 97.5% of its respective sample and use them to plot the envelope.

The results suggest that good fitness levels for real data can be attained by the distributions proposed by the PIPE method, in agreement to what was suggested by the simulations. These distributions presented simpler mathematical expressions for the cdf and pdf when compared to the distributions in the previous paper on those data sets. Whenever estimation was necessary, it was done by the `fitdistr()` routine from the `MASS` package in `R`. For some models, however, we reported the estimates of the parameters and the respective logarithm of the likelihood from the original papers.

The parameters for the PIPE algorithm in this section are the same used in the simulation section.

Wheaton river data set

The following data consist of 72 observations of the exceedances of flood peaks, measured in m^3/s, of the Wheaton River located near Carcross in the Yukon Territory, Canada. These data were analyzed by several authors, amongst which we cite a few. In Akinsete et al. (2008) the

four parameter beta-Pareto (BP) distribution was used to model the data. Its density is given by

$$f(x) = \frac{k}{\theta B(\alpha,\beta)} \left\{1 - \left(\frac{x}{\theta}\right)^{-k}\right\}^{\alpha-1} \left(\frac{x}{\theta}\right)^{-k\beta-1},$$

with $x \geq \theta$ and α, β, θ, $k > 0$. A better fit, regarding the most common criteria such as Akaike information criterion (AIC), was found by Alshawarbeh et al. (2012) using the beta-Cauchy (BC) distribution. The pdf for the beta-Cauchy distribution is given by

$$f(x) = \frac{\lambda}{\pi B(\alpha,\beta)} \left\{\frac{1}{2} + \frac{1}{\pi}\arctan\left(\frac{x-\theta}{\lambda}\right)\right\}^{\alpha-1} \left\{\frac{1}{2} - \frac{1}{\pi}\arctan\left(\frac{x-\theta}{\lambda}\right)\right\}^{\beta-1} \frac{1}{\lambda^2 + (x-\theta)^2},$$

where $-\infty < x < \infty$, $0 < \alpha, \beta, \lambda < \infty$ and $-\infty < \theta < \infty$. Finally, Cordeiro et al. (2013) used the exponentiated generalized Gumbel (EGGU) distribution to achieve an even better fit for the data. The pdf for the EGGU distribution is

$$\begin{aligned} f(x) =& \alpha\beta\sigma^{-1} \left\{1 - \left[1 - \exp\left\{-\exp\left(-\frac{x-\mu}{\sigma}\right)\right\}\right]^{\alpha}\right\}^{\beta-1} \left[1 - \exp\left\{\exp\left(-\frac{x-\mu}{\sigma}\right)\right\}\right]^{\alpha-1} \\ & \times \exp\left\{-\exp\left(-\frac{x-\mu}{\sigma}\right)\right\} \exp\left(-\frac{x-\mu}{\sigma}\right), \end{aligned}$$

with $-\infty < x < \infty$, $0 < \alpha, \beta, \sigma < \infty$ and $-\infty < \mu < \infty$.

Bourguignon et al. (2013) also analyzed the data, but they used the Kumaraswamy-Pareto (KWP) distribution with pdf

$$f(x) = \frac{abk\beta^k}{x^{k+1}} \left[1 - \left(\frac{\beta}{x}\right)^k\right]^{a-1} \left\{1 - \left[1 - \left(\frac{\beta}{x}\right)^k\right]^a\right\}^{b-1},$$

for $x \geq \beta$ and $0 < a, b, k, \beta < \infty$. The histogram for this data set is very skewed to the right as seen in Figure 2.2.

We obtained, using the PIPE method, for this data set the following cdf:

$$F(x) = \frac{1.17258x}{x + 11.6991},\ 0 \leq x < x_1,$$

with $F(x_1) = 1$, $x_1 = 67.7894$. The corresponding pdf is given by

$$f(x) = \frac{13.7181}{(x + 11.6991)^2},\ 0 \leq x < x_1.$$

Table 2.2 Estimates and log-likelihood for the models adjusted to the Wheaton river data set.

	ϕ	$\widehat{\phi}$	LL
BP	(α,β,k,θ)	$(7.69, 85.75, 0.02, 0.10)$	–
KWP	(a,b,k,β)	$(2.86, 85.85, 0.05, 0.10)$	-271.20
BC	$(\alpha,\beta,\lambda,\theta)$	$(387.65, 1.46, 2.05, 0.08)$	-260.48
EGGU	(a,b,μ,σ)	$(0.11, 0.48, 2.63, 1.63)$	-256.90
PIPE	–	–	-235.83

For this example we removed $\exp(-\cdot)$ from F, to search for simpler alternatives. Table 2.2 shows the information of the fitting of the models in the previous papers. We used the estimates and information available from the original papers. The information on the likelihood of the BP model was not available in Akinsete et al. (2008) and was omitted here as well. Figure 2.2 shows the pdfs of the EGGU and the PIPE model overlapping the data's histogram and the quantile plot with 95% bootstrap simulated envelope for the PIPE model. The histogram suggests that the proposed distribution is suitable for the data. The quantile plot shows a good agreement between the model and the data. The largest observation in this data set is 64.0, which is much larger than the second largest one, 39.0, considering the sample standard-deviation of 12.41m^3/s. The PIPE model seems to capture the behavior of this possibly extreme value very well. The KS test returned a p-value equal to 0.3197 and AS test returned 0.3341 suggesting that there is no evidence of a bad fit. As in the previous example, the PIPE distribution achieved the highest likelihood. We emphasize that the distribution obtained from the PIPE method has a much simpler analytic expression for its cdf and cdf than the other candidates.

Ball bearing data set

The next data set is used in many papers on lifetime distributions. It consists of 23 observations of the fatigue failure times, measured in millions of revolutions, of ball bearings. In Nassar and Nada (2012) the authors proposed to use the beta-exponential-geometric (BEG) distribution,

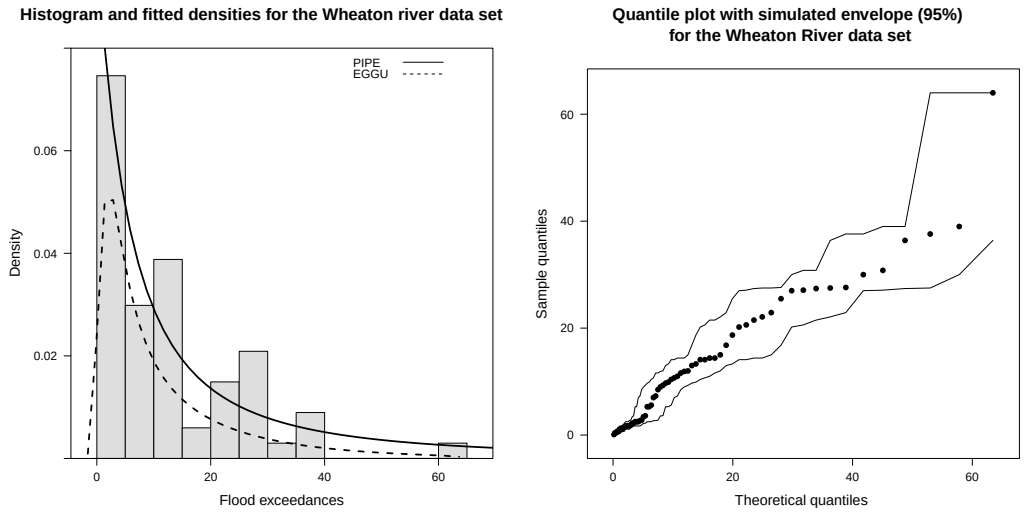


Figure 2.2 Histogram and quantile plot for the Wheaton river data set.

which they presented, and it fitted the data nicely. Its pdf is given by

$$f(x) = \frac{1}{B(a,b)} \left(\frac{1-\mathrm{e}^{-\beta x}}{1-p\mathrm{e}^{-\beta x}} \right)^{a-1} \frac{\beta(1-p)^b \mathrm{e}^{-b\beta x}}{(1-p\mathrm{e}^{-\beta x})^{b+1}},$$

for $x > 0$, $p \in (0,1)$, a, b, $\beta > 0$. This is the result of the composition of the beta distribution and the exponential geometric distribution (Adamidis and Loukas, 1998). The gamma and exponentiated Weibull (EW) (Mudholkar and Srivastava 1993), with cdf $F(x) = \{1 - \exp[-(x/\lambda)^k]\}^\alpha$, $\lambda > 0$, $\alpha > 0$, $x > 0$, distributions are also considered here. The EW distribution is a good benchmark since it is known to be able to fit a very wide variety of data. The data are:

17.23, 28.92, 33.00, 41.52, 42.12, 45.60, 48.80, 51.84, 51.96, 54.12, 55.56, 67.80, 68.64, 68.64, 68.88, 84.12, 93.12, 98.64, 105.12, 105.84, 127.92, 128.04, 173.4.

Table 2.3 shows the fitted distributions and the logarithm of the maximized distribution. The PIPE method proposed the function

$$F(x) = 0.000847396^{\mathrm{e}^{-x/26.272}}, \; -\infty < x < \infty$$

as cdf. The Kolmogorov-Smirnov test for the PIPE distribution returned a p-value of 0.8974,

Table 2.3 Estimation for the ball bearings data set.

	ϕ	$\widehat{\phi}$	LL
BEG	(p,a,b,β)	$(0.35,5.38,2.58,0.01)$	-113.06
EW	(α,k,λ)	$(4.49,1.06,34.83)$	-113.06
Weibull	(k,λ)	$(2.10,81.88)$	-113.73
Gamma	(α,β)	$(3.99,0.06)$	-113.10
PIPE	–	–	-113.50

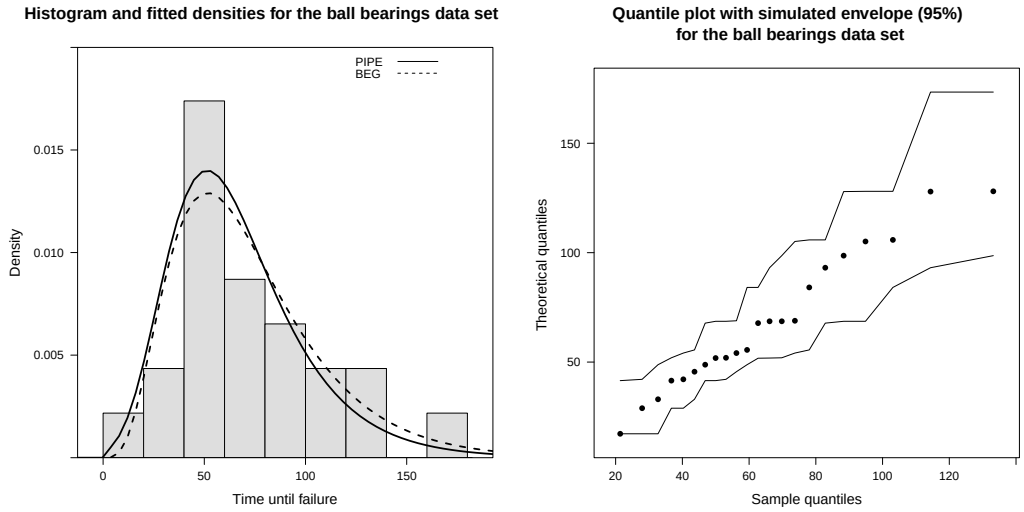


Figure 2.3 Histogram and quantile plot for the ball bearings data set.

whereas for the Anderson-Darling test the value was 0.8007. Figure 2.3 suggests the PIPE distribution is adequate for the data and this agrees with the previous tests. The logarithm of the likelihood for this model was -113.5005, which is slightly lower than the one from the BEG distribution. However, it is computationally much simpler to obtain values for $P(X \leq x)$ under the PIPE model than under the BEG model – even a handheld calculator suffices. There is the inconvenience of $P(X < 0) \neq 0$, however it is a small value for this particular application.

2.6 Addressing issues and final comments

In this chapter we suggested the use of the PIPE method for generating possibly new continuous probability distributions from a given data set. The method described here allows the development of relatively simple distributions that performed better than several more complicated available distributions in two data sets. Its use was also illustrated by several runs of simulations with artificial data. The simulation studies suggest that the PIPE algorithm can properly handle data modeling from a wide variety, including heavy tailed, bimodal and skewed data.

We recommend the use of this method if there is no evidence in the problem being studied that may lead to the use of a specific distribution, existing or new. The main advantage over the use of distributions in the so called G-classes (as the beta-G and gamma-G classes) is that it is possible to control the complexity of the new distribution using the PIPE method. The cdf of the G-classes distributions tend to have mathematically complicated formulae.

There are also situations where we do not recommend the use of this method. For instance, if it is possible to develop an specific distribution from the underlying physical properties of the problem, we do not recommend using the PIPE method unless there is a clear advantage in doing so. Another situation where we do not recommend its use is when the problem satisfies all the theoretical criteria for the use of an existing distribution, such was the waiting times in Poisson processes or small measurement errors that are easily handled by the normal distribution. Adequacy measures and the likelihood criteria should never replace proper mathematical analysis of the problem.

Future efforts may be able to describe the mathematical properties of the search in the space of the candidates distributions. A useful refinement that we were not able to provide is to obtain an algorithm for generating a function that integrates up to a constant. If there was such an algorithm we would be able to find distributions with support on the whole real line or in the $(0, \infty)$ half of the line in a much easier way. However, finding a distribution that has support on bounded intervals is also reasonable.

We observed many other interesting events during the development of this paper, while

working on other examples. Changing the numbers in the cdfs found by the PIPE method for unknown parameters and estimating them by maximum likelihood led to values very close to those proposed by the algorithm for many data sets (real and simulated). In some data sets we changed the seed for the random number generators or modified some of the initial conditions, such as the values in $\mathbf{P}_{d,w}$. These runs of the algorithm led to different solutions (cdfs) since it is a non-deterministic algorithm. However, these solutions were usually very similar to each other. For the ball bearings and the Wheaton river data sets there were two groups of very similar solutions that were suggested by the PIPE method depending on the initial settings and the method did not find anything outside those groups. Last, the output of the algorithm seems to depend heavily on the choice of F. Adding or removing elements to F changes drastically the behavior of the algorithm in our experience. We suggest starting from very few elements and adding more as needed. We also suggest keeping the mutation probability high enough. It seems to play a major role in the final result.

Overall, our opinion is that the PIPE algorithm is an interesting and promising alternative in the field of data modeling.

CHAPTER 3

Beta L2 Model

RESUMO

Nesse capítulo, apresentamos uma nova classe de distribuição obtida através da composição de duas classes já existentes. Essa composição permite adicionar até três novos parâmetros, os quais podem contribuir para um melhor ajuste do modelo a uma variedade maior de conjuntos de dados. Alguns modelos já existentes são casos particulares dessa nova classe. Apresentamos resultados gerais para uma família genérica nessa nova classe. O resultado principal é a uma representação da função densidade de probabilidade como mistura de distribuições exponencializadas, o que ajuda a obter várias propriedades matemáticas das novas famílias tais como momentos e funções geradoras de momentos. Também apresentamos a caracterização pelo princípio da máxima entropia para essa classe. Uma aplicação a um conjunto de dados reais é utilizada para ilustrar o uso de uma família da nova classe. Comparamos o novo modelo a nove outros modelos já existentes. Estimação dos parâmetros é brevemente discutida.

3.1 Introduction

In this chapter, a new wide class of continuous distributions is investigated. This class is obtained by adding three parameters to a parent continuous distribution with cumulative distribution function (cdf) $G(x)$.

Alzaatreh et al. (2013) generate a new class by using an existing distribution as baseline and a transformation of a given cdf. Their very general framework is defined as follows: Let X be a random variable with probability density function (pdf) $g(x)$ and cdf $G(x)$. This cdf will act as a baseline distribution. Let T be a continuous random variable with pdf $r(t)$ and support

$[a,b]$. The cdf of a new family of distributions is defined by

$$F(x) = \int_a^{W(G(x))} r(t)dt,$$

where $W(G(x))$ ranges from a to b, is differentiable and monotonically non-decreasing, with $\lim_{x\to-\infty} W(F(x)) = a$ and $\lim_{x\to+\infty} W(G(x)) = b$. The cdf $F(x)$ is thus $R(W(G(x)))$, where $R(t)$ denotes the cdf of the random variable T. In their paper, the authors provided several examples of choices for T and W. The class of distributions proposed in this chapter comes by taking $W(u) = u$ and T as a beta-exponential variate (Nadarajah and Kotz, 2006). This investigation is mainly motivated by two reasons. First, families from the beta-G class of distributions have a good performance in many applications. Among the papers with real data applications of the beta-G, we cite: Nadarajah and Gupta (2004), Nadarajah and Kotz (2004), Fischer and Vaughan (2010), Paranaíba et al. (2011) and Cordeiro et al. (2012). The extra parameter may improve further the usefulness of these families. Second, the generated class is a combination of the beta- G and Lehmann type II classes of distributions. This class appears very briefly in Alzaatreh et al. (2013) but is not discussed in enough details and was only used in a few examples of their work. Its ability to outperform several other classes, as shown later in this paper, makes a solid reason for further investigations.

The chapter is organized as follows. In Section 3.2, we define the new class of distributions. In Section 3.3, we present some general properties of this class, such as moments, moment generating function (mgf) and entropies. In Section 3.4, we provide special models in this class. In Section 3.5, we address estimation of the model parameters by maximum likelihood. Finally, Section 3.6 presents an application of two families in the new class.

3.2 The Beta-L2-G class

The class investigated in this paper is actually equivalent to the composition of two generators: the beta generator and the Lehmann type II generator. Then, a distribution in the new class with baseline cdf $G(x)$ is referred to as the Beta-L2-G distribution. The class pdf is given

by

$$f(x) = \frac{c}{B(a,b)}\,[1-G(x)]^{bc-1}\,\{1-[1-G(x)]^{c}\}^{a-1}\,g(x), \tag{3.1}$$

where $G(x)$ and $g(x)$ are the cdf and pdf of the baseline distribution.

Using the generalized binomial expansion twice, we can easily prove that

$$f(x) = \frac{c\,g(x)}{B(a,b)}\sum_{j,k=0}^{\infty}(-1)^{j+k}\binom{a-1}{j}\binom{(b+1)c-1}{k}G(x)^{k}. \tag{3.2}$$

A useful expansion for the pdf in (3.1) can be derived using the concept of exp-G distribution. Based on equation (3.2), we can write

$$f(x) = \sum_{k=0}^{\infty} v_{k+1}\,h_{k+1}(x), \tag{3.3}$$

where $h_{k+1}(x)$ is the exp-G$(k+1)$ density function with power parameter $k+1$ and

$$v_{k+1} = \frac{c}{(k+1)\,B(a,b)}\sum_{j=0}^{\infty}(-1)^{j+k}\binom{a-1}{j}\binom{(b+1)c-1}{k}.$$

This is done in many papers on members of the G-classes, and quickly allows to obtain most mathematical properties of the new family based on the corresponding properties of their exp-G counterparts. Nevertheless, when $G(x)$ and $g(x)$ are simple, the mathematical properties can be derived via numerical integration from Equation (3.1) without much computational effort. In the next section, we obtain some of these mathematical properties using the mixture representation (3.3).

3.3 General properties of the new family

In this section, we discuss some general properties of the Beta-L2-G class. Henceforth, let $X \sim$ Beta-L2-$G(\boldsymbol{\eta},\boldsymbol{\phi})$ be a random variable whose pdf is given by (3.1), with $\boldsymbol{\phi}$ representing the vector of parameters of $G(x)$ and $\boldsymbol{\eta} = (a,b,c)^{\top}$, and let $Y_{k+1} \sim$ exp-G$(k+1)$, for $k \geq 0$.

The general expression for the moments of the new family can be obtained from Equation (3.3). Let $\mu_j = \mathbb{E}(X^j)$ be the j-th ordinary moment of X and $\nu_{j,k+1} = \mathbb{E}(Y_{k+1}^{j})$. From Equation

(3.3), it follows that

$$\mu_j = \sum_{k=0}^{\infty} v_{k+1} \nu_{j,k+1}.$$

The same can be stated for the mgf of X regarding that one of Y_k. It is simply given by

$$M_X(t) = \sum_{k=0}^{\infty} v_{k+1} M_{k+1}(t),$$

where $M_{k+1}(t)$ denotes the mgf of Y_{k+1}. Using well known recursive formulas, the cumulants and the central and factorial moments are easily obtained. Based on them, mean deviations, Bonferroni and Lorenz curves and other characteristics of X are readily available. Further, manipulation of the sums and integer power of sums will reveal a mixture representation for the density function of the order statistics in this family. Those calculations are lengthy and not given here, but they mimic very closely what is done in, for example, Pinho et al. (2015) and references therein.

One characteristic that does not follow directly from the series representation in (3.2) is the Shannon entropy. It refers to the amount of uncertainty (or surprisal) associated to a random variable. It is an important concept in many areas of knowledge, specially theory of information, physics and probability. Although there are many other entropy measures, this one is possibly the most popular and was introduced in the seminal paper by Shannon (1948). For a continuous distribution $F(x)$ with density $f(x)$, the Shannon entropy is given by

$$\mathscr{H}_{Sh}(f) = \mathbb{E}\{(-\log[f(X)])\} = -\int_{-\infty}^{+\infty} \{\log[f(x)]\} f(x) dx.$$

Alzaatreh et al. (2013) used the Beta-L2-G to illustrate a theorem involving the Shannon entropy in their framework. From Lemma 2 in Alzaatreh et al. (2013), it follows that

$$\begin{aligned}\mathscr{H}_{Sh}(f) = &-\mathbb{E}\{\log g[G^{-1}(1-\mathrm{e}^{-X})]\} + \log[c^{-1}B(a,b)] + (a+b-1)\psi(a+b) \\ &-(a-1)\psi(a) - b\psi(b) - c^{-1}[\psi(a+b) - \psi(b)].\end{aligned}$$

We use a different formula to provide the maximum entropy characterization of this class. This alternative formula is given in the Appendix. The entropy of Shannon can be used to identify probability models as seen in Jaynes (1957). Consider a class of distributions defined by a set of constraints such as

$$\mathscr{F} = \{f(x) | \mathbb{E}_X[L_i(X)] = t_i, i = 1, 2, \ldots, m\},$$

where $t_i \in \mathbb{R}$, $\forall i$. We can choose a member of $\mathscr{F}$ as the pdf of a random variable X if it maximizes the Shannon entropy under these constraints. The chosen pdf is called the maximum entropy distribution. This approach ensures that no other assumptions except those from the constraints are made. For instance, we can prove that if the first and second moments are constrained, the maximum entropy distribution is the normal distribution or that if we only that the data at hand are positive, the maximum entropy distribution is the exponential distribution. More often than not, the calculations of the Shannon's entropy provide clues of what are the constraints involved in the maximum entropy characterization.

For the Beta-L2-G distribution, the maximum entropy characterization is given by these three constraints:

- $\mathbb{E}[\log[1-G(X)])] = [\psi(b) - \psi(a+b)]$;
- $\mathbb{E}[\log\{1-[1-G(X)]^c)\}] = \psi(a) - \psi(a+b)$;
- $\mathbb{E}[\log g(X)] = \mathbb{E}\{\log[G^{-1}(1-Z^{-c})]\}$, where $Z \sim Beta(b,a)$ and $\psi(\cdot)$ is the digamma function.

The proof can be found in the Appendix B, it uses the different (but equivalent) ways of expressing the Shannon entropy for this class.

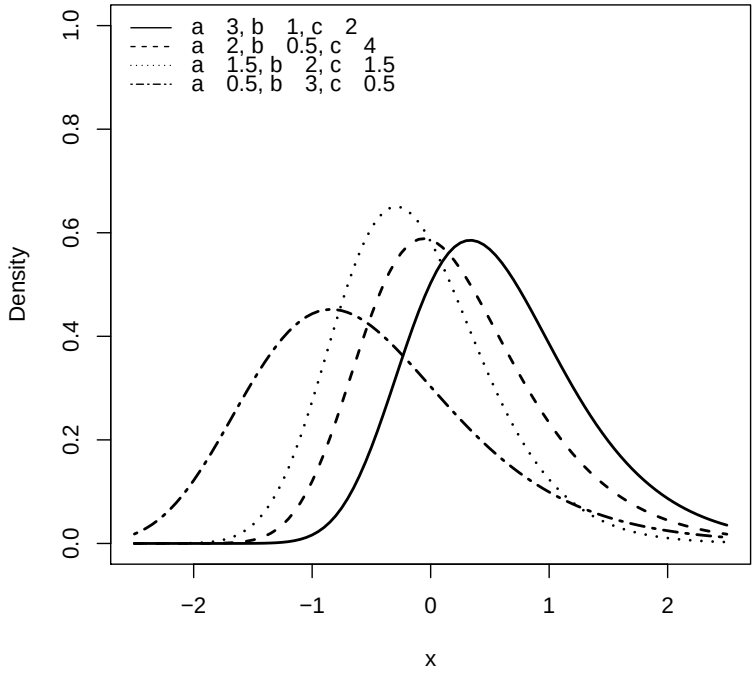
3.4 Some members of the Beta-L2-G class

In this section, we present some characteristics of the generated family for particular choices of the cdf $G(x)$.

3.4.1 Beta-L2-Gumbel

This family comes by inserting $G(x) = \exp[-\exp(-z)]$, with $z = \beta^{-1}(x-\mu)$, into Equation (3.1). Figure 3.1 displays some shapes of the new density function for $\mu = 0$ and $\beta = 1$.

Figure 3.1 Some shapes of the Beta-L2-Gumbel pdf for selected parameter values



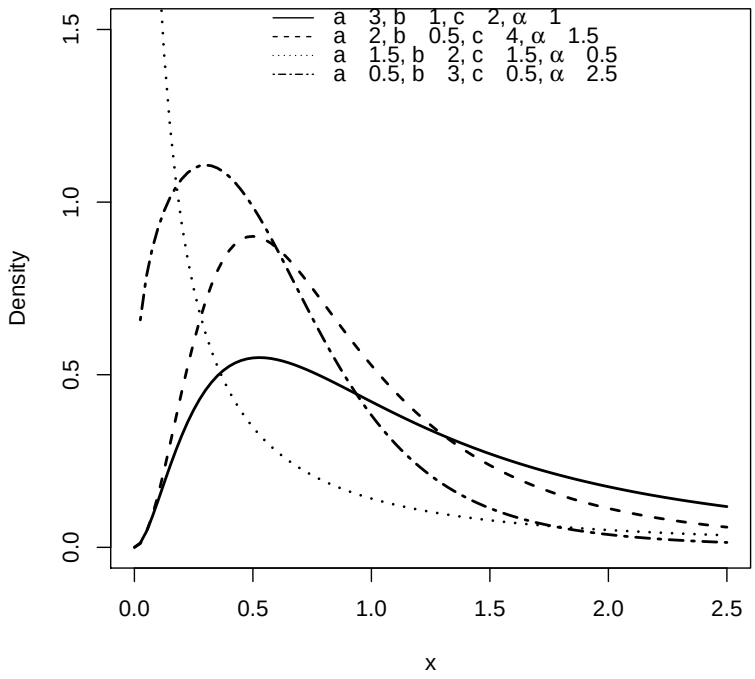
3.4.2 Beta-L2-**Log-logistic**

This family follows from inserting

$$G(x) = 1 - \frac{1}{1 + (\frac{x}{\lambda})^{-\alpha}}$$

into Equation (3.1). Figure 3.2 displays some possible shapes of the new density function. In Section 3.6, we use this new family to model breaking strengths of glass fibres. The results indicate a superior fit of the Beta-L2-log-logistic distribution when compared to several others old and new distributions.

Figure 3.2 Some shapes of the Beta-L2-Log-logistic pdf for selected parameter values



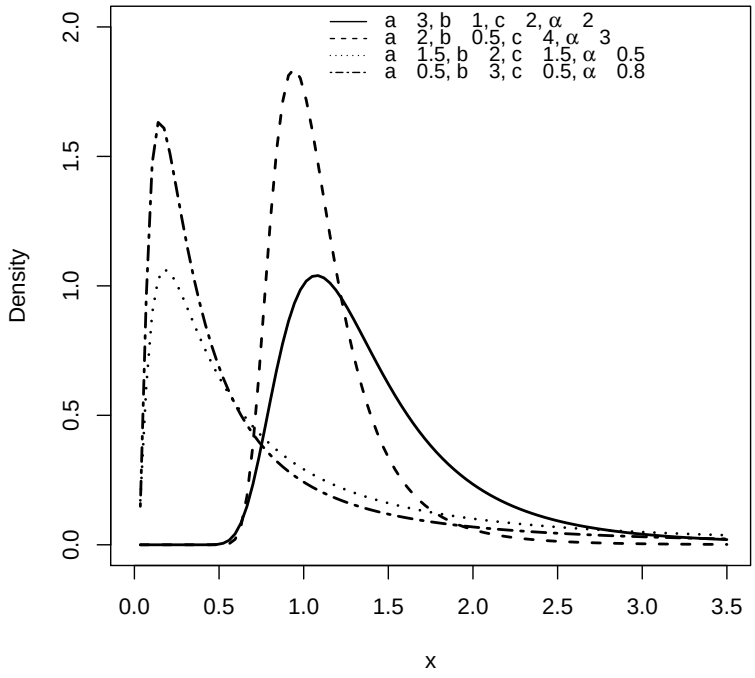
3.4.3 Beta-L2-Fréchet

This family is obtained by inserting

$$G(x) = 1 - \mathrm{e}^{-z^{-\alpha}},\ \alpha > 0,$$

where $z = \beta^{-1}(x-\mu)$ and $x > \mu$, into Equation (3.1). Figure 3.3 displays some possible shapes of the Beta-L2-Fréchet density function.

Figure 3.3 Some shapes of the Beta-L2-Fréchet pdf for selected parameter values



3.5 Estimation

Estimation of the Beta-L2-G model parameters can be accomplished by the maximum likelihood method. Based on a random sample $x_1, \ldots, x_n$, the logarithm of the likelihood function

for the parameters in (3.2) is given by

$$\ell = \ell(a,b,c,\boldsymbol{\phi}^\top) = -n\log(c^{-1}B(a,b)) - (bc-1)\sum_{i=1}^{n}\log[1-G(x_i)] + (a-1)\sum_{i=1}^{n}\log(1-[1-G(x_i)]^c) + \sum_{i=1}^{n}\log[g(x_i)].$$

Then, the components of the score function are

$$\begin{aligned}
\frac{\partial \ell}{\partial a} &= \sum_{i=1}^{n}\log(1-[1-G(x_i)]^c) + n[\psi(a)-\psi(a+b)],\\
\frac{\partial \ell}{\partial b} &= c\sum_{i=1}^{n}\log[1-G(x_i)] + n[\psi(b)-\psi(a+b)],\\
\frac{\partial \ell}{\partial c} &= nc^{-1} + b\sum_{i=1}^{n}\log[1-G(x_i)] - (a-1)\sum_{i=1}^{n}\frac{[1-G(x_i)]^c}{1-[1-G(x_i)]^c}\log[1-G(x_i)]
\end{aligned}$$

and

$$\frac{\partial \ell}{\partial \phi_j} = (bc-1)\sum_{i=1}^{n}\frac{1}{1-G(x_i)}\frac{\partial G(x_i)}{\partial \phi_j} + (a-1)\sum_{i=1}^{n}\frac{c[1-G(x_i)]^{c-1}}{1-[1-G(x_i)]^c}\frac{\partial G(x_i)}{\partial \phi_j} + \sum_{i=1}^{n}\frac{1}{g(x_i)}\frac{\partial g(x_i)}{\phi_j},$$

where $\boldsymbol{\phi}^\top = (\phi_1,\ldots,\phi_k)$ denotes the parameters of $G(x)$ and $1 \le j \le k$. Setting these derivatives to zero and solving the resulting equation system yields the maximum likelihood estimators (MLEs) of the model parameters. Unfortunately, the $k+3$ equations cannot be simplified any further for a generic distribution G and require the use of an iterative numerical method such as the Newton-Raphson or quasi-Newton procedures, even in simple cases. Under general regularity conditions, the asymptotic distribution of $(\hat{a},\hat{b},\hat{c},\hat{\boldsymbol{\phi}}^\top)^\top$ is $N_{k+3}(\mathbf{0},\boldsymbol{K}^{-1})$, where $\mathbf{K} = \boldsymbol{K}(a,b,c,\boldsymbol{\phi}^\top)$ is the expected information matrix. The matrix $\boldsymbol{K}$ can be replaced by the observed information evaluated at the MLEs matrix for constructing asymptotic confidence intervals for the parameters.

Care is advised when extracting a numerical approximation for the matrix $\mathbf{K}$ from the iterative methods used to obtain the estimates of the parameters. For some methods, such as the Broyden-Fletcher-Goldfarb-Shanno (BFGS) method, an approximation of the Hessian matrix is used in the calculations at each iteration. This approximation may not be reliable if the convergence of the methods happens too fast. If the number of iterations is small (e.g. five or six

iterations), using the BFGS method, the output given for the approximate Hessian matrix may be unreliable, even when the point estimates are very accurate. Bootstrap confidence intervals are a reliable alternative in these cases. The convergence of the estimation procedures usually depends on the choice of the starting values of the parameters. We advise first using a non-deterministic optimization routine, such as simulated annealing, to obtain the initial guesses of the parameters and then using the Newton or quasi-Newton methods. This greatly helps to find decent first guesses, although it adds to the total computational time, this is usually reliable, especially when dealing with simulation and bootstrap. One recent, fully implemented, very useful routine is found in the C library `libcgrpp` (Silva et al., 2011). It implements the C-GRASP metaheuristic mainly as suggested in Hirsch et al. (2010).

Other estimation methods such as the method of moments (see Cramèr, 1946, Section 33) or the generalized method of moments (Hansen, 1982) may be used. Particularly, the generalized method of moments may be used in conjunction with the maximum entropy characterization to produce estimates of the parameters. Another alternative is the use of the log-cumulants in the estimation process as described in Nicolas (2002), which possesses good statistical properties, such as low variance, according to Anfinsen and Eltoft (2011). This is, however, a discussion which may be long and thus fit to be presented in a separate work.

3.6 Application to real data

The following data set consists of 63 observations of the breaking strength of glass fibers with length of 1.5cm. These data were obtained at the UK National Physical laboratory and studied by Smith and Naylor (1987), Jones and Faddy (2004) and Fischer and Vaughan (2010), among others. We fit several models to the current data:

- the Beta-L2-log-logistic (BL2LL) distribution;
- the Beta-L2-logistic (BL2L) distribution with baseline given by

$$G(x) = \frac{1}{1+\mathrm{e}^{-\frac{x-\mu}{\sigma}}},\ x, \sigma > 0;$$

- the generalized gamma (GG) distribution with cdf

$$F(x) = \frac{\gamma(dp^{-1}, [xa^{-1}]^p)}{\Gamma(dp^{-1})};$$

  where $\gamma(a,z) = \int_0^z t^{a-1}\mathrm{e}^{-t}dt$ represents the incomplete gamma function.

- the exponentiated-Weibull (EW) distribution (Mudholkar and Srivastava, 1993) with cdf

$$F(x) = \left[1 - \mathrm{e}^{-(\frac{x}{\lambda})^k}\right]^{\alpha},\ x,\ k,\ \lambda,\ \alpha > 0;$$

- the gamma distribution, with cdf $F(x) = \gamma(\alpha, \beta x)/\Gamma(\alpha)$; and

- the Weibull distribution, which is the EW distribution with $\alpha = 1$.

We also consider six models investigated by Fischer and Vaughan (2010): the normal distribution, beta-normal (BN) distribution, beta-logistic (BL) distribution, beta-hyperbolic secant (BHS) distribution, beta-Student's t (BT) distribution and beta-generalized hyperbolic secant (BGHS) distribution. The references for these distribution can be seen in Fischer and Vaughan (2010), most of them follow from, for example, Jones (2004).

For the first six models, the MLEs of the parameters, the logarithm of the maximized likelihood function (LL) and the Akaike Information Criterion (AIC) are listed in Table 3.1. These values are obtained in the `R` software. The parameters are estimated by the function `fitdistr()` from package `MASS` with initial guesses for the parameters found by simulated annealing function `GenSA()` from the `GenSA` package.

Since the BL2LL and BL2L models are five-parameter distributions, we compare to the five-parameter BT and BGHS distributions and to the four-parameter BN, BL and BHS distributions from Fischer and Vaughan (2010). The values for the LL and AIC obtained in Fischer and Vaughan (2010) are given in Table 3.2. The BHS distribution yields the best overall fit based on the LL and AIC statistics. However, Fischer and Vaughan found that these distribution presented a few identification problems. It is a common issue with these new generated distributions to have a likelihood function with very flat areas.

From the new distributions presented here, the BL2LL yields the best fit according to both the LL and AIC statistics. The generalized gamma and exponentiated-Weibull distributions are

Table 3.1 Parameter estimates and relative selection criteria for the current example.

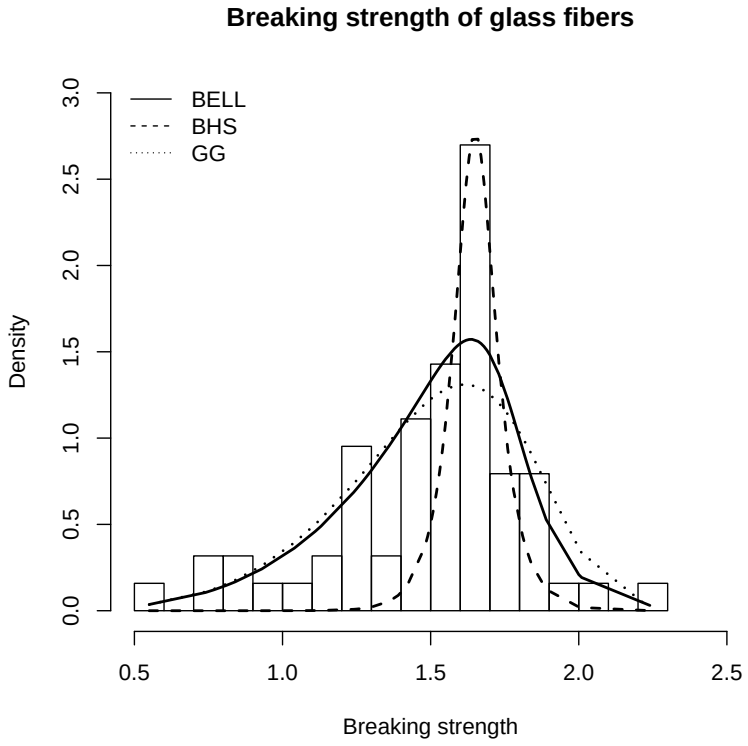
Distribution	Parameter	Estimate	Std. Error	LL	AIC
BL2LL	a	0.1262	0.0249	-8.73	27.45
	b	0.0462	0.0116		
	c	11.6021	0.1084		
	α	1.5746	0.0447		
	λ	25.4154	0.0542		
BL2L	a	0.1407	0.0220	-10.48	30.97
	b	1.8299	0.5680		
	c	0.1080	0.0040		
	μ	1.7088	0.0040		
	σ	0.0404	0.0031		
GG	a	1.7693	0.1053	-14.59	35.18
	d	4.8102	0.8926		
	p	7.7626	2.0217		
EW	α	0.6712	0.2489	-14.67	35.35
	k	7.2846	1.7069		
	λ	1.7181	0.0861		
Gamma	α	17.4385	3.0778	-23.95	51.90
	β	11.5730	2.0722		

Table 3.2 Relative selection criteria for the distributions in Fischer and Vaughan (2010).

Distribution	LL	AIC	Distribution	LL	AIC
Normal	-17.91	39.82	BHS	-10.02	28.03
BN	-14.06	36.11	BT	-11.41	32.82
BL	-10.49	28.99	BGHS	-9.90	29.80

very flexible distributions and usually perform well in many applications. They are used here as benchmarks for the fitness of the new models. Both BL2LL and BL2L models perform better than the generalized gamma and exponentiated-Weibull distributions. The BL2LL model also performs better than the distributions in Fischer and Vaughan (2010), one of which is a particular case of the new class. Surprisingly, the standard errors for estimating the new families' parameters are relatively small and provide no indication of poorly identified density functions even though they have five parameters. So, at least for this application, it is one advantage of the new models. Figure 3.4 displays three of the fitted density functions overlapping the histogram of the data.

Figure 3.4 Three of the fitted densities.



3.7 Concluding remarks

In this Chapter, we explore the Beta-Lehmann 2-G (Beta-L2-G) class. The main results are a mixture representation for its density function and the maximum entropy characterization. Other structural results follow directly from the mixture representation. The benefits of the addition of three new parameters to an existing baseline distribution are presented by means of an application to a real data set. In the practical example, the Beta-Lehmann 2-log-logistic (BL2LL) model provides the best fit when compared to other models of similar complexity.

CHAPTER 4

The Quadratic Transform-Marshal-Olkin-G class of distributions

RESUMO

Nesse capítulo, apresentamos outra classe de distribuições obtida através da composição de duas classes já existentes. Apresentamos resultados gerais para uma escolha genérica da distribuição baseline. Apresentamos uma representação da função densidade de probabilidade como mistura de distribuições exponencializadas, como no capítulo anterior. Uma aplicação a um conjunto de dados reais é utilizada para ilustrar o uso de uma família da nova classe.

4.1 Introduction

In this chapter we present and analyse another new class of probability distributions that adds two new parameters to an existing distribution. The Quadratic Transformed Marshall Olkin-G, QTMO-G for short, is the composition of the Transmuted-G (Shawn and Buckley, 2007) and Marshall-Olkin-G (Marshall and Olkin, 1997) classes of distributions. This chapter is divided as follows. In Sections 4.2 and 4.3 we give general information on the Transmuted-G and Marshall-Olkin-G classes, respectively. Section 4.4 shows the new class of distributions we are proposing and a mixture representation for the pdf based on a generic baseline distribution. In Section 4.5 we illustrate the behaviour of the pdf and hazard rate function (hrf) of some of the new families in this class. We also present examples of the behaviour of the kurtosis and skewness as function of the additional parameters for selected baseline distributions. Section 4.6 briefly discusses the estimation of the parameters for this class. An application is shown in Section 4.7 and final comments are addressed in Section 4.8.

4.2 Transmuted distributions

In Shawn and Buckley (2007) the authors attempt to infer the change of variables of one random variable that leads to another random variable with given distributions. The motivation for this comes from simulation of random variables. They argue that undergraduate students usually learn to simulate a random variable X with cdf $G(X)$ by generating a uniform random variable U and then setting $x = G^{-1}(u)$ and little emphasis is given to the fact that occurrences of X may be obtained from another random variable Y and a suitable mapping $h : \mathscr{Y} \to \mathscr{X}$, where $\mathscr{X}$ and $\mathscr{Y}$ are the supports of Y and X. Such is the case of the Box-Muller transform and the well known relation between the chi-square and normal distributions. Using computational algebra and Cornish-Fisher expansions it is possible to find such changes of variables or approximations of it.

The same authors proceed to use such mappings with the intent of introducing modulations in existing distributions. This can, for example, induce different skewness and kurtosis to the distributions and is essentially in the same context of the generated distributions. They call the mapping $u \to F(G^{-1}(u))$ a rank transmutation. The mapping $u \to u + \lambda u(1-u)$, $|\lambda| < 1$, is called the quadratic transmutation and this leads to $F(x) = (1+\lambda)G(x) - \lambda G(x)^2$. There are at least two interesting characteristics of this generated cdf. First, if $G(x)$ has a symmetric pdf the transmutation preserves all the even moments and the parameter λ is able to induce extra skewness. Second, the distribution of the square of a random variable with cdf $G(x)$ is the same of the square of the random variable with cdf $F(x)$.

Bourguignon et al. (2016) studied the distributions generated by the quadratic mapping from Shawn and Buckley (2007) as a member of the G-classes. The Transmuted-G distribution has cdf given by $F(x) = (1+\lambda)G(x) - \lambda G(x)^2$, for a baseline $G(x)$. Bourguignon et al. (2016) provided and discusses many of the mathematical properties that are typical of G-classes studies, such as moments, entropies, Kullback-Leibler divergence and estimation, as well as providing applications of members of the Transmuted-G class to real data sets. Shaw and Buckley (2007) commented that some of these rank transmuted distributions are not thoroughly described because they are not always mathematically tractable, even with computa-

tional algebra systems. Indeed, many results described in Bourguignon et al. (2016) must be found via numerical techniques. Fortunately, some of the expressions for the moments of particular Transmuted-G distributions are very concise. Some of them are listed in Bourguignon et al. (2016).

The pdf of the Transmuted-G distribution is given by

$$f(x) = [1 + \lambda - 2\lambda G(x)]g(x), \tag{4.1}$$

in which $g(x)$ is the pdf of $G(x)$. A useful way to express the pdf in ((4.1)) is

$$f(x) = (1 + \lambda)g(x) - 2\lambda G(x)g(x),$$

because $G(x)g(x)$ is the pdf of an exp-G distribution with power parameter equal to 2. This leads to immediate results for the moments of the transmuted-G class.

From this point on, we will refer to the transmuted-G class as the quadratic transformed-G class for two reasons. The first is that Shawn and Buckley (2007) also consider a cubic map which is able to modify the kurtosis of some baseline distributions, though it is less tractable than the quadratic one. The second reason is that "transmuted" does not reflect the nature of the method. The quadratic transform is most useful when there is the need to introduce more skewness to a model and should not be taken as something other than that.

4.3 The Marshall-Olkin class of distributions

Consider a sequence of independent random variables $Y_1, Y_2, \ldots$ all of them with a common cdf $G(x)$ and pdf $g(x)$. Let $X = \min\{Y_1, Y_2, \ldots, Y_N\}$, where N is a positive integer valued random variable with probability generating function $\varphi(\cdot, \theta)$ for $\theta > 0$. The survival function $\bar{F}(x)$ of X is given by

$$\bar{F}(x) = \varphi(\bar{G}(x), \theta), \tag{4.2}$$

where $\bar{G}(x) = 1 - G(x)$. This kind of composition was studied in many details by Aly and Benkherouf (2011). For instance, when $\bar{G}(x) = \exp(-\lambda x)$, $\lambda > 0$ with $\varphi(s, \theta)$ represent-

ing a zero truncated Poisson pgf, namely $\varphi(s,\theta) = \exp[\theta(s-1)][1-\exp(-\theta)]^{-1}$, the resulting distribution is the exponential Poisson (EP) distribution from Kus (2006). When $\bar{G}(x) = \exp(-\lambda x)$ and $\bar{G}(x) = \exp[-(\lambda x)^{\alpha}]$, where $\varphi(s,\theta)$ denotes the probability generating function of a generic power series distribution, we obtain the models in Chahkandi et al. (2009) and Morais (2011), respectively. Other proposals for this kind of mixture can be found in Barreto-Souza et al. (2011), Lu and Shi (2012), Ristic (2012) and in the references sections of these papers.

One of the most popular versions of (4.2) was pioneered by Marshall and Olkin (1997) based on the geometric pgf $\varphi(\cdot,\theta) = s\theta(1-\bar{\theta}s)^{-1}$, where $\bar{\theta} = 1-\theta$. In the same paper, the method was applied to the exponential and Weibull distributions yielding the Marshall-Olkin extended exponential and Marshall-Olkin extended Weibull distributions.

4.4 The new class of distributions

The Quadratic Transform Marshall Olkin - G (QTMO-G) class of distributions is obtained by the composition of the Transmuted-G and Marshall-Olkin-G classes of distributions. The cdf of the QTMO-G is given by

$$F(x) = \alpha(1+\lambda)\left[1-\frac{\overline{G}(x)}{1-\overline{\alpha}\overline{G}(x)}\right] - \lambda\alpha^2\left[1-\frac{\overline{G}(x)}{1-\overline{\alpha}\overline{G}(x)}\right]^2. \tag{4.3}$$

The corresponding pdf is given by

$$f(x) = \alpha\left[1+\lambda+2\lambda\frac{\overline{G}(x)}{1-\overline{\alpha}\overline{G}(x)}\right]\frac{g(x)}{[1-\overline{\alpha}\overline{G}(x)]^2}.$$

As in Bourguignon et al. (2016), this pdf can be written as

$$f(x) = (1+\lambda)h(x,1) - \lambda h(x,2),$$

where $h(x,i)$ is the pdf of a random variable with distribution exp-MO-G with power parameter i.

This pdf can be expressed as a mixture of exp-G pdfs. Several mathematical properties and quantities of the QTMO-G distributions can be obtained directly from this, such as moments,

cumulants, moment generating functions, distribution of the order statistics and more. This mixture is obtained as follows. Let

$$a_i = \alpha(1-\alpha)\sum_{j=i}^{\infty}\binom{j}{i},\ i \geq 0,$$

such that, by using the negative binomial series and the binomial theorem, we have

$$\frac{\alpha\overline{G}(x)}{(1-\overline{\alpha}\overline{G}(x))} = \sum_{i=0}^{\infty} a_i[G(x)]^i.$$

By similar calculations we have

$$\alpha(1-\overline{\alpha}\overline{G}(x))^{-2} = \sum_{i=0}^{\infty} b_i[G(x)]^i,$$

with

$$b_i = \alpha(1-\alpha)\sum_{j=i}^{\infty}(1+j)\binom{j}{i}.$$

With these two expansions we may write

$$\begin{aligned} f(x) &= \left[1+\lambda-2\lambda\sum_{i=0}^{\infty}a_i G(x)^i\right]\left(\sum_{j=0}^{\infty}b_n G(x)^j\right)g(x) \\ &= \left[(1+\lambda)\sum_{j=0}^{\infty}b_j G(x)^j - 2\lambda\sum_{i=0}^{\infty}\sum_{j=0}^{\infty}a_i b_j G(x)^{j+i}\right]g(x) \\ &= \left[(1+\lambda)\sum_{j=0}^{\infty}b_j G(x)^j - 2\lambda\sum_{s=0}^{\infty}c_s G(x)^s\right]g(x), \end{aligned}$$

with $c_s = \sum_{i=0}^{s} a_i b_{s-i}$. Finally,

$$f(x) = \sum_{i=0}^{\infty} d_i g_i(x),$$

with

$$d_i = \frac{(1+\lambda)b_i - 2\lambda c_i}{s+1},$$

and $g_i(x) = (i+1)g(x)G(x)^s$. This expansion leads to a direct connection of the QTMO-G distributions properties to the Exp-G distributions properties, such as moment and moment generating function.

4.5 Some families in the QTMO-G class

In this section we show some aspects of selected members of the QTMO-G class of distributions. For each selected member, we show how the additional parameters affect the skewness and kurtosis. The Galton's skewness (Johnson et al, 1994, p. 40) and Moors' kurtosis (Moors, 1988) are the measures we use to illustrate the behaviour of the skewness and kurtosis as functions of the parameters. These are considered more robust than those usual skewness and kurtosis measures and have the advantage of existing even for distributions without moments. The Galton's skewness is given by

$$G = \frac{Q\left(\frac{3}{4}\right) + Q\left(\frac{1}{4}\right) - 2Q\left(\frac{1}{2}\right)}{Q\left(\frac{3}{4}\right) - Q\left(\frac{1}{4}\right)},$$

and the Moors kurtosis is given by

$$M = \frac{Q\left(\frac{3}{8}\right) - Q\left(\frac{1}{8}\right) + Q\left(\frac{7}{8}\right) - Q\left(\frac{5}{8}\right)}{Q\left(\frac{6}{8}\right) - Q\left(\frac{2}{8}\right)}.$$

4.5.1 QTMO-Weibull

This family is obtained by using

$$G(x) = 1 - \mathrm{e}^{-\theta x^k}$$

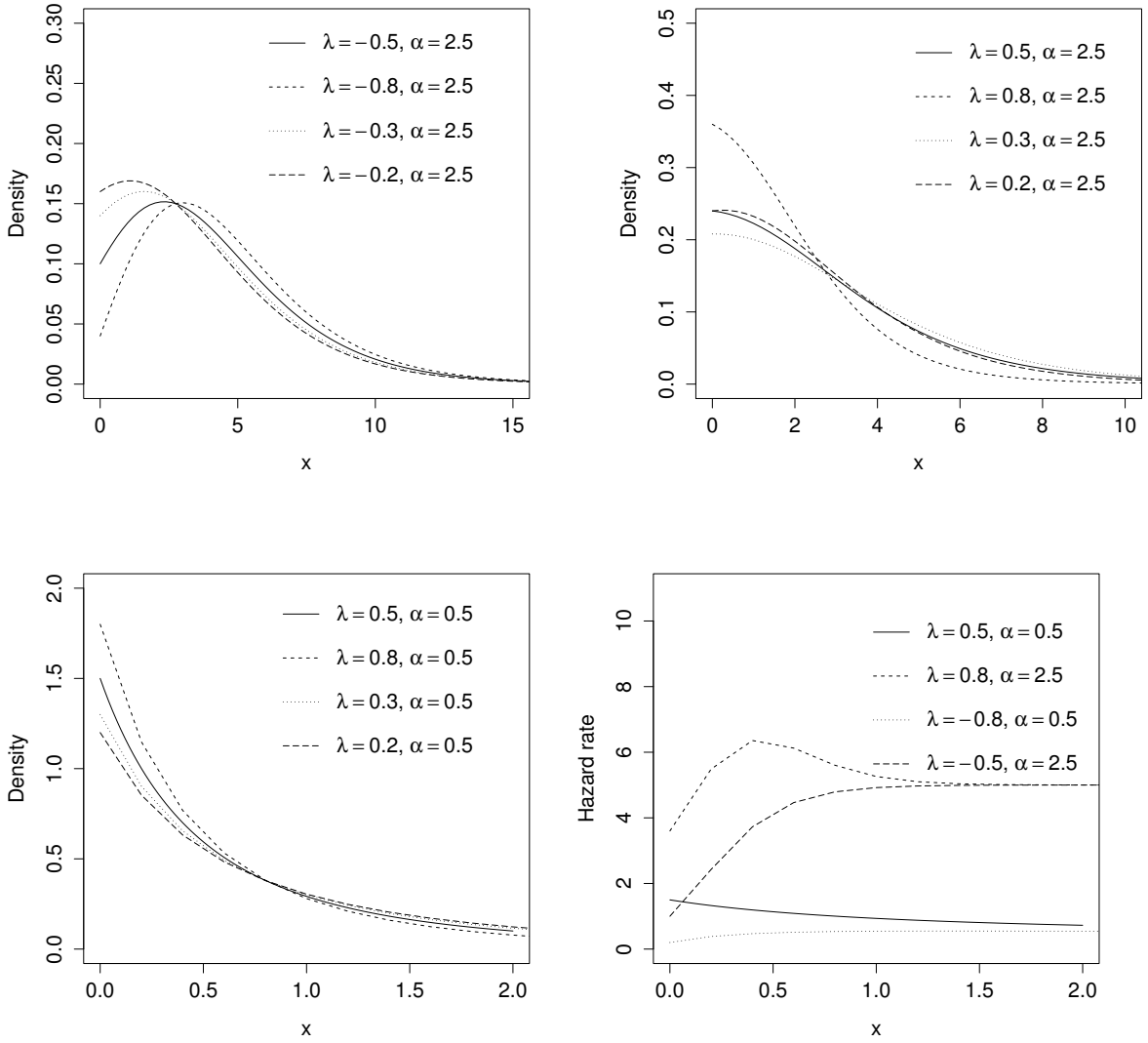
as the baseline distribution in (4.3). We use $\theta = 1$ and $k = 2$. Different shapes for the pdf and hrf of the QTMO-Weibull are shown in Figure 4.1. Plots for the Galton's skewness and Moor's kurtosis are shown in Figure 4.2. Overall, both the kurtosis and skewness increase as λ goes from -1 to 1, with varying convexity depending on the value of α.

4.5.2 QTMO-log-logistic

The cdf for the QTMO-log-logistic is obtained by inserting

$$G(x) = \frac{1}{1 + (x/\beta)^k},$$

Figure 4.1 QTMO-Weibull pdfs and hrfs.



in (4.3). Some of the shapes for the pdf and hrf of this family, for selected values of k and $\beta = 1$, are presented in Figure 4.3.

For the skewness and kurtosis in this example, we use $k = 2$ and $\beta = 1$, allowing λ and α to vary. The result is shown in Figure 4.4. It appears that both the kurtosis and skewness do not depend on α and the decrease as λ goes from 0 to 1. Both the skewness and kurtosis seem increase in this family as λ goes from -1 to 0.

Figure 4.2 Galton's skewness and Moor's kurtosis for a QTMO-Weibull family.

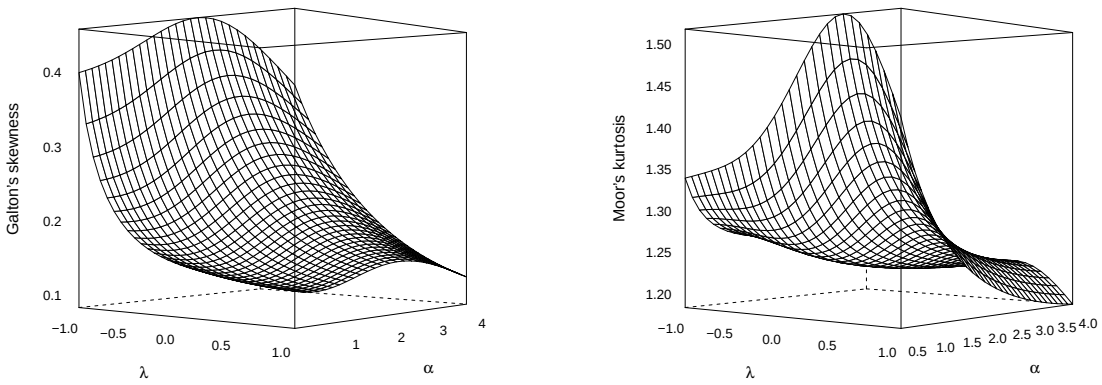
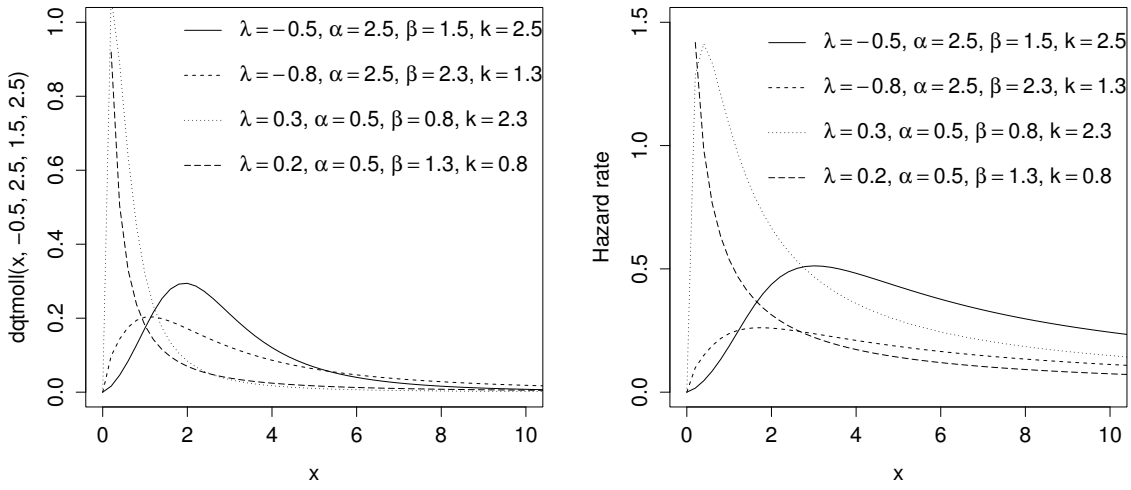


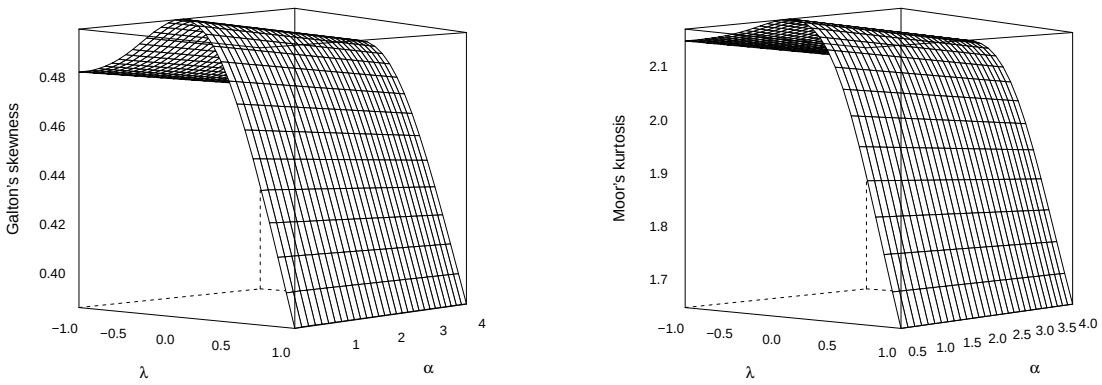
Figure 4.3 QTMO-log-logistic pdfs and hrfs with $\beta = 1$.



4.5.3 QTMO-normal

The cdf for the QTMO-normal distribution is easily obtained by using the normal distribution cdf $\Phi(x;\mu,\sigma)$ in (4.3). Since the normal distribution only has location and scale parameters, we set $\mu = 0$ and $\sigma = 1$. Different shapes for the pdf and hrf are seen in Figure 4.5 while skewness and kurtosis are seen in Figure 4.6. While the skewness behaves much like in the

Figure 4.4 Galton's skewness and Moor's kurtosis for a QTMO-log-logistic family.



QTMO-Weibull distribution, the kurtosis measure seems more complex.

Figure 4.5 QTMO-standard normal pdfs and hrfs.

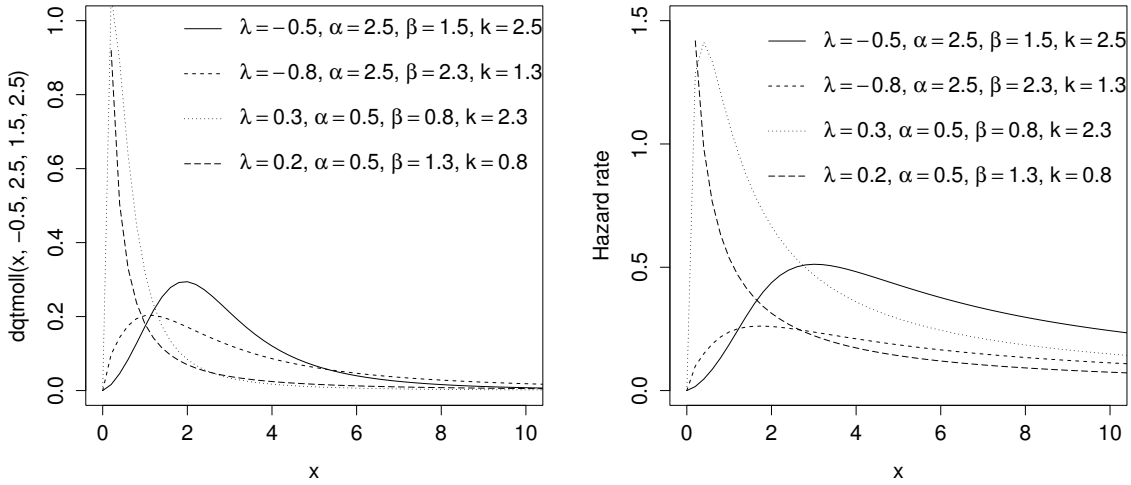
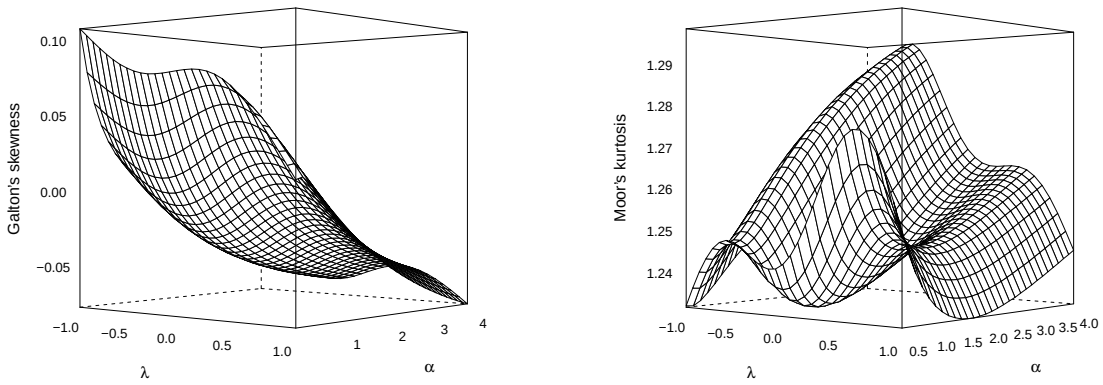


Figure 4.6 Galton's skewness and Moor's kurtosis for the QTMO-standard normal.



4.5.4 QTMO-Kumaraswamy

The Kumaraswamy distribution, used as baseline, and proposed in Kumaraswamy (1980), is well known for its applications in hydrology (Fletcher and Ponnambalam, 1996). The pdf for the Kumaraswamy distribution is

$$g(x) = abx^{a-1}(1-x)^{b-1},\ x \in [0,1],\ a > 0,\ b > 0.$$

Different shapes for the QTMO-Kumaraswamy distribution can be seen in Figure 4.7. The kurtosis surface shows a complexity similar to that of the QTMO-standard normal. The skewness and kurtosis seen in Figure 4.8

4.6 Estimation

In this section, we consider the estimation procedure by the maximum likelihood method for the parameters in the QTMO-G families. Estimates for the parameters, based on a sample $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)^\top$, are obtained by solving the set of equations $\partial l/\partial \theta_i(\boldsymbol{x}) = 0$, $i = 1, \ldots, k$, with $\boldsymbol{\theta} = (\theta_1, \theta_2, \ldots, \theta_k)^\top$ representing the vector of parameters of the model and

Figure 4.7 Some possible shapes for the pdf and hrf of the QTMO-Kumaraswamy family.

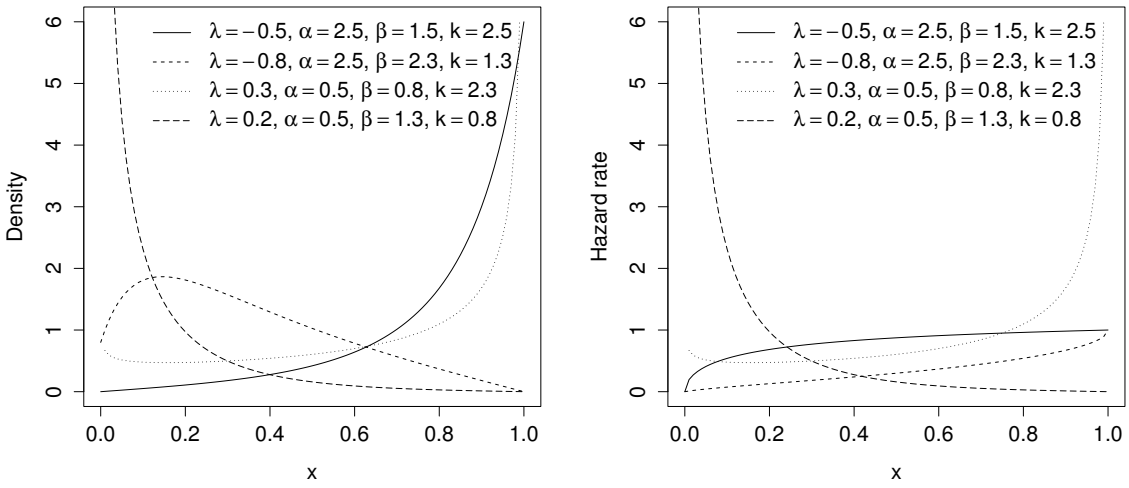
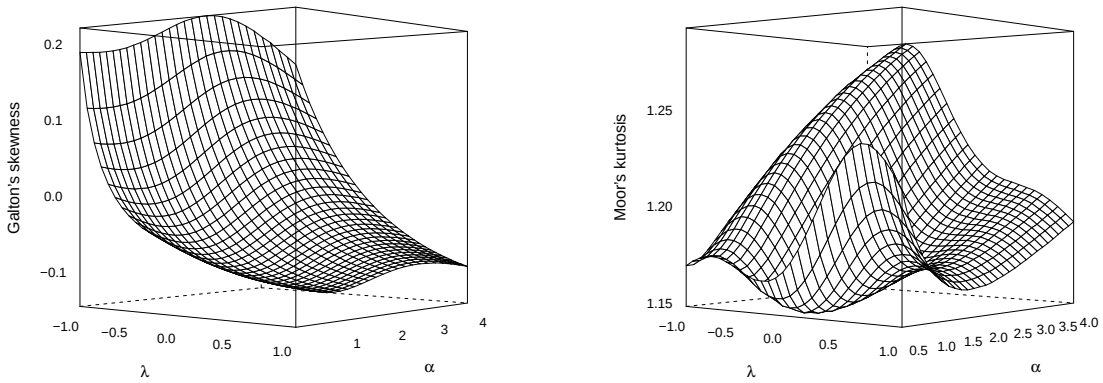


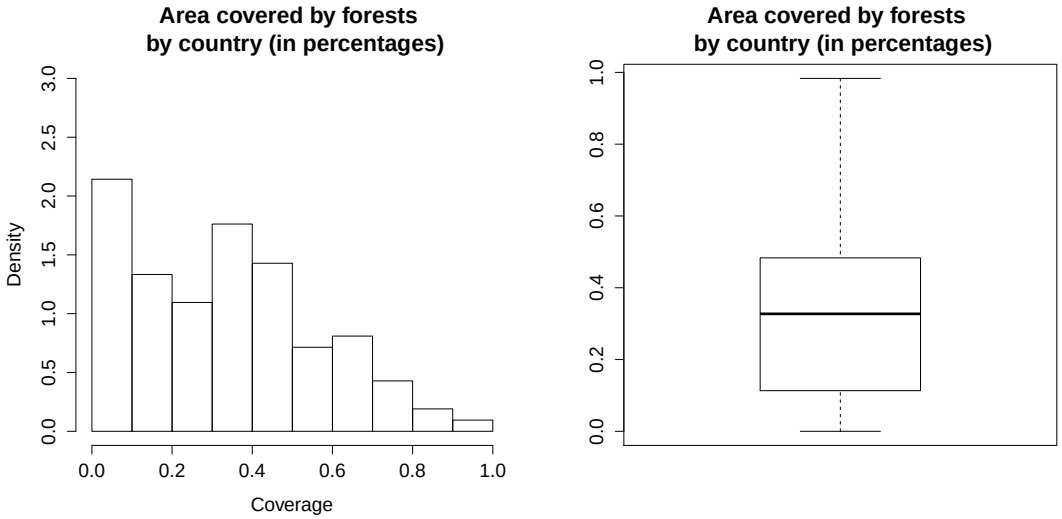
Figure 4.8 Galton's skewness and Moor's kurtosis for a QTMO-Kumaraswamy.



$l(\boldsymbol{x}) = \sum_{i=0}^{n} \log f(x_i)$ representing the logarithm of the likelihood function based on the sample.

Since this set of equations is very likely not to have explicit solutions, iterative methods are employed, as it was the case in the previous chapter. This can be achieved by the `fitdistr()` routine from package `MASS`. We emphasize that choosing appropriate initial guesses in this set up may be frustrating and time consuming in practical applications. Much in the same way as

Figure 4.9 Histogram and box-plot for the coverage data.



it was with the Beta-L2-G class, it is possible to avoid this problem by using heuristic methods, such as simulated annealing. The `GenSA()` routine, from package `GenSA`, in R is able to do so. Heuristic methods can be used on their own as a tool for estimation. This is achieved, for instance, by using the `AdequacyModel` package, from R.

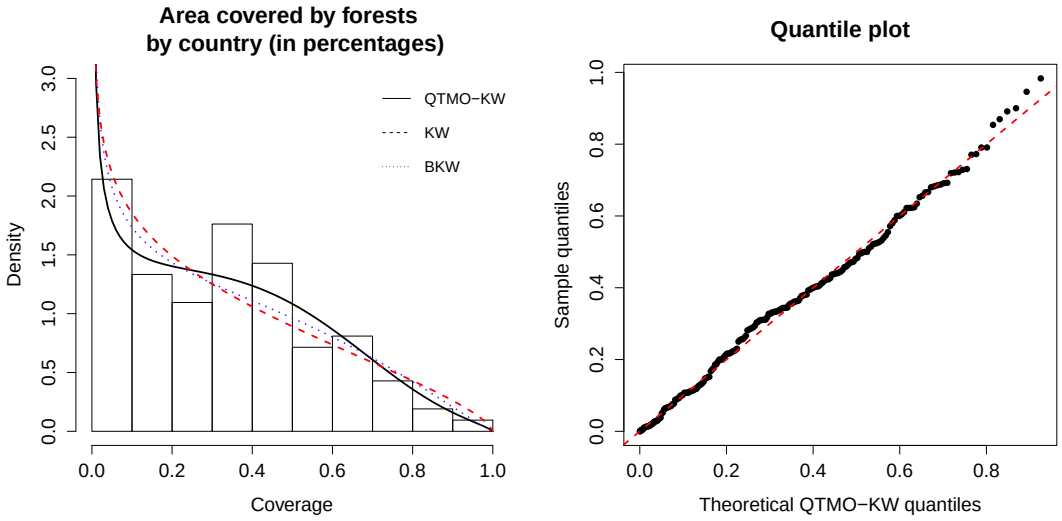
4.7 Application to a real data set

In this section, we present an application of a member of the QTMO-G class of distributions to a real data set to illustrate its use. It also shows the improvement over the baseline distribution regarding the fit to the data.

The data set we consider is included in Appendix C for easy access and it can be obtained online at `http://data.un.org/Data.aspx?d=MDG\&f=seriesRowID\%3a567`. It consists of 210 observations of the proportion of land area covered by forest in 2010 for several countries, expressed as percentage. An histogram and a box-plot for these data are presented in Figure 4.9. The data is clearly skewed and presents a higher frequency near zero and another peak of frequency around 0.4.

To these data we fitted the QTMO-Kumaraswamy distribution. The baseline distribution is

Figure 4.10 Histogram and fitted model. Quantile plot for the QTMO-KW distribution.



confined to the $[0,1]$ interval and that makes it a potentially favourable choice for the dataset. The baseline distribution was also fitted to the data. An asymptotic likelihood ratio test for testing the fit of the QTMO-KW distribution versus the fit of the KW distribution yields a p-value of 0.0028. This suggests that the additional parameters are indeed useful. We also fitted the exponentiated-Weibull (EW) distribution (Mudholkar and Srivastava, 1993) as it is widely known as being able to model a large variety of data and is usually regarded as a benchmark to new models. The Weibull, gamma and beta-KW models were also considered. The beta-KW is another four parameter distribution that extends the Kumaraswamy distribution. Its pdf is given by using the Kumaraswamy distribution as baseline in the composition

$$F(x) = \frac{1}{B(\alpha,\beta)} \int_0^{G(x)} t^{\alpha-1}(1-t)^{\beta-1}.$$

The three best fitting densities are shown in Figure 4.10 overlapping the histogram for the data and the quantile plot for the QTMO-KW distribution. The plots suggest that the QTMO-KW is well suited for modelling the data. The estimated values of the parameters for every distribution, the Akaike Information Criterion (AIC) and the p-value for the Kolmogorov-Smirnov and Anderson-Darling tests are displayed in table 4.1.

The QTMO-G is, according to the AIC and likelihood levels, a better choice than the other competing distributions. The KS and AD tests suggest that all the distributions considered here

Table 4.1 Parameter estimates and relative selection criteria for the forest coverage example.

Distribution	Parameter	Estimate	Std. Error	LL	AIC	KS	AD
QTMO-KW	λ	0.76	0.2182	48.68	-89.36	0.7269	0.9014
	α	5.15	2.8391				
	a	0.56	0.1257				
	b	1.77	0.3197				
Beta-KW	α	0.25	0.0188	46.16	-84.30	0.2369	0.4336
	β	3.82	0.5532				
	a	2.99	0.0088				
	b	0.53	0.0125				
EW	α	0.19	0.0135	47.19	-88.37	0.1582	0.4008
	k	3.92	0.0265				
	λ	0.69	0.0264				
KW	a	0.83	0.0672	44.22	-84.44	0.0863	0.1994
	β	1.68	0.1707				

are suitable to the data at the most usual confidence levels. This makes the comparison more meaningful.

4.8 Concluding remarks

In this chapter, we presented the Quadratic Transform Marshall Olkin-G class of distributions, which is obtained by the composition of two already existing G-classes: the Transmuted-G and Marshall-Olkin-G. The expansion for the pdf of a general member of this class allows to quickly obtain some mathematical properties of this class based on those of the exponentiated class of distributions.

Some mathematical properties of the QTMO-G class are note mathematically tractable.

Closed expressions or series representations for them are not available, even with the aid of computer algebra systems. This is not too detrimental, as most of the quantities can be calculated by using iterative methods with great precision. Estimation of the parameters in this class of distributions benefits greatly from the use of heuristic methods.

The application section provided empirical evidence of the usefulness of this class. The fit of the QTMO-Kumaraswamy to the data was superior to the one of the baseline alone. The QTMO-Kumaraswamy, to these data, also had a superior performance when compared to already existing models of equal or lower complexity.

CHAPTER 5

Estimation procedures using multilayer perceptrons for univariate models

RESUMO

Nesse capítulo, mostramos como utilizar redes neurais simples para a estimação de parâmetros em distribuições de probabilidade. Isso permite aproximar as estimativas obtidas pelo método dos momentos sem a necessidade de métodos iterativos. O método foi testado em conjuntos de dados reais e fictícios com ótimos resultados. O mesmo método pode ser aplicado a outras distribuições. Mostramos uma aplicação a um problema de segmentação de imagens SAR no qual o método reduz consideravelmente o tempo de computação necessário.

5.1 Introduction

The method of moments and generalized method of the moments (Hansen, 1982) are very popular methods for parameter estimation. The first consists of equating some theoretical moments to their numeric sample counterparts and solving the equation system to obtain estimates. The properties of this estimation method are well known. These properties are well documented in, for example, Cramèr (1946). The generalized method of moments, as proposed by Hansen (1982), equates the expected values of functions of the random variable to their numeric counterparts. Solving the system of equations provides the estimates. The equations in Hansen are mainly from orthogonality equations typically arising in Economy problems.

In both methods, solving the equations systems is usually difficult and relies on numerical methods. These methods, in practical applications, may require programming skills of anyone wanting to use non-common distributions, such as those in the generated distributions con-

text (see Tahir and Nadarajah, 2015). We will focus, initially, on the generalized method of moments (GMM) as the basic version is a special case of it.

Let $\underset{\sim}{\theta}$ be the vector of parameters of a distribution with cumulative distribution function (cdf) $F(x)$. Consider a random sample of size n, $(Y_1,\ldots,Y_n)^T = \underset{\sim}{Y}$. If $g(Y_i,\underset{\sim}{\theta})$ is such that $\mathbb{E}[g(Y_i,\underset{\sim}{\theta}_0)] = 0$, when $\underset{\sim}{\theta}_0$ is the "true" value of $\underset{\sim}{\theta}$, and $\hat{m}(\underset{\sim}{\theta}) = \frac{1}{n}\sum g(Y_i,\underset{\sim}{\theta})$, then the GMM estimator of $\underset{\sim}{\theta}$ is found by minimizing

$$\|\hat{m}(\underset{\sim}{\theta})\|_W^2 = \hat{m}(\underset{\sim}{\theta})^T\, W\, \hat{m}(\underset{\sim}{\theta}),$$

where the matrix W is positive-definite. This estimator can be viewed as a function of the data. Given the data $(Y_1,Y_2,\ldots,Y_n)^T$, the numerical value of $\hat{\underset{\sim}{\theta}}$ is

$$\arg\min_{\underset{\sim}{\theta}} = \hat{m}(\underset{\sim}{\theta})^T\, W\, \hat{m}(\underset{\sim}{\theta}).$$

This function can be highly non-linear or may assume a simple form. The former is far more frequent than the latter.

It is well known that neural networks can emulate non-linear functions (Hornik, 1991). We investigate in this chapter, how do multilayer perceptrons (MLP), a class of neural networks, perform in estimating the parameters of probability distributions from the sample moments. We compare the performance of the MLP estimator to the performance of the MM and GMM estimators, since all three of them use basically the same information as input. To use a MLP in such way, we require that $\hat{m}(\underset{\sim}{\theta})$ may be written as $\hat{m}(\underset{\sim}{\theta}) = m_1(\underset{\sim}{Y}) - m_2(\underset{\sim}{\theta})$, where $m_1(\underset{\sim}{Y})$ does not depend on $\underset{\sim}{\theta}$. The input of the network will be $m_1(\underset{\sim}{Y})$ and the output will be the values of $\hat{\underset{\sim}{\theta}}$.

This chapter is divided as follows. In Section 4.2, we provide a brief introduction to the GMM estimation. Section 4.3 provides the basic information regarding MLP neural networks. In Section 4.4, we present applications of this method to several probability distributions. Section 4.5 shows an application of this method to a problem involving the statistical treatment of SAR images. The last section addresses our final comments and possible extensions.

5.2 The generalized method of the moments

In this Section, we describe the GMM procedure as proposed in Hansen (1982). This is only a brief explanation. For further details, readers are referred to the original paper.

Let $\underset{\sim}{Y} = (Y_1, Y_2, \ldots, Y_n)^T$ be a random sample with independent and identically components and let $F_Y(y)$ be their common distribution. The parameters of F are denoted by $\underset{\sim}{\theta}$. We do not require F to be continuous or discrete. Let $g(Y, \underset{\sim}{\theta})$ be a vector valued function such that $I\!E[g(Y_i, \underset{\sim}{\theta}_0)] = \underset{\sim}{0}$, when $\underset{\sim}{\theta}_0$ is the "true" value of $\underset{\sim}{\theta}_0$ and $\underset{\sim}{0}$ is a vector of zeros with appropriate dimension.

Consider the function $\hat{m}(\underset{\sim}{\theta}) = \frac{1}{n}\sum_{i=1}^{n} g(Y_i, \underset{\sim}{\theta})$, which is the sample counterpart of $I\!E[g(Y_i, \underset{\sim}{\theta}_0)]$. The GMM estimator is obtained as $\hat{\underset{\sim}{\theta}} = \arg\min_{\underset{\sim}{\theta}} \hat{m}(\underset{\sim}{\theta})^T W \hat{m}(\underset{\sim}{\theta})$.

For the estimation described above, the following conditions are sufficient to guarantee the strong consistency of $\hat{\underset{\sim}{\theta}}$.

- If S is the parametric space and ξ is some norm, the metric space (S, ξ) is separable and S is compact.
- $g(\cdot, \underset{\sim}{\theta})$ is Borel mensurable for each $\underset{\sim}{\theta}$ in S and $g(y, \cdot)$ is continuous for every y.
- $I\!E[g(Y_i, \underset{\sim}{\theta})]$ exists and is finit for every $\underset{\sim}{\theta} \in S$ and $I\!E[g(Y_i, \underset{\sim}{\theta}_0)] = 0$ also, $\underset{\sim}{\theta}_0$ is the only value such that the expected value is zero.
- Let $\varepsilon(\omega, \underset{\sim}{\theta}, \delta) = \sup\left\{|g(Y(\omega), \underset{\sim}{\theta}) - g(Y(\omega), \underset{\sim}{\alpha})| : \underset{\sim}{\alpha} \in S, \xi(\underset{\sim}{\theta}, \underset{\sim}{\alpha}) < \delta\right\}$. Then $\lim_{\delta \downarrow 0} I\!E[\varepsilon(\omega, \underset{\sim}{\theta}, \delta)] = 0$.

Hansen (1982) also provides a set of sufficient conditions for the asymptotic normality of the estimator. This, however, is not important for the particular use of the GMM in this paper.

5.3 Multilayer perceptron (MLP) neural networks

This section, initially, brings a very brief introduction to MLPs and neural networks. A neural network, loosely speaking, is a system of inputs and outputs vaguely based on some

biological process. Mathematically, let g be a function, $\underset{\sim}{x}$ and $\underset{\sim}{y}$ vectors such that $g(\underset{\sim}{x}) = \underset{\sim}{y}$. Suppose that g is not easy or practical to be computed. A MLP can be used to approximate $g(\underset{\sim}{x})$. This process is made in, usually, three steps.

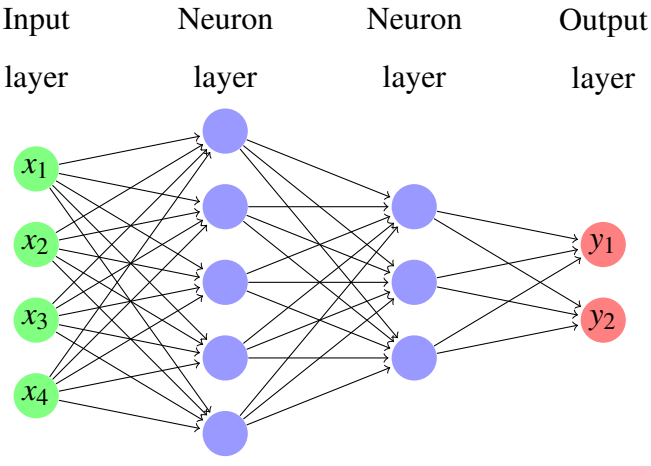
1. Example collection: We provide some examples of pairs $(\underset{\sim}{x}, \underset{\sim}{y})$, such that $\underset{\sim}{y} = g(\underset{\sim}{x})$.

2. Learning or training: The network is trained to correctly assign each $\underset{\sim}{x}$ to a value close to its respective $\underset{\sim}{y}$.

3. Validation: During the training phase, it is possible to overfit the data. That means the network will perform incredibly good in the example set but may perform very poorly for $\underset{\sim}{x}$ outside the example set. The validation phase checks if there is evidence of overfitting.

A neuron is the basic element of a MLP. It receives a value v and returns $\phi(v)$, where $\phi(\cdot)$ is called activation function and usually has range in $[0,1]$ or $[-1,1]$. Neurons are organized in layers. The first layer of neurons is proceeded by a layer of inputs of the network. Between every input and neuron there is a weighted link called synapsis. The weight between neuron j and input i is denoted ω_{ij}^1. The input of this neuron j is $v_j = \sum_{i=1}^{n} \omega_{ij} x_i$, where n is the number of inputs of the network. Refer to Figure 5.1.

Figure 5.1 A general MLP diagram.



Th output of the jth neuron in the first layer will be denoted $\theta_{1j} = \phi(v_j)$. The outputs will be the inputs of the next neurons layer, and so on. Finally, the last layer will have as many

neurons as $g(\underset{\sim}{x})$ has entries. The jth output of the last layer will be denoted θ_j. It is usual to make $x_1 = 1$. A popular choice for $\phi(v)$ is $\phi(v) = (1+e^{-v})^{-1}$, which is known as the sigmoid function.

The learning process consists in adjusting the weights of the network to make its output close to $\underset{\sim}{y}$ for each corresponding $\underset{\sim}{x}$. A popular method of doing so is the back-propagation algorithm. Consider the cost function

$$C = \frac{1}{2}\sum_j (\theta_j - y_j)^2, \qquad y_j \quad \text{in} \quad \underset{\sim}{y}.$$

The weights can be adjusted in an iterative fashion aiming at reducing the value of the cost function. The gradient of C regarding the weights ω_{ij} can be shown to equal

$$\frac{\partial C}{\partial \omega_{ij}} = \delta_j \, \theta_i, \text{ with}$$

$$\delta_j = \begin{cases} (\theta_j - y_j)\theta_j(1-\theta_j), \text{ if } j \text{ is in the output layer,} \\ \left(\sum_k \delta_j \, \omega_{jk} \, \theta_j (1-\theta_j)\right), \text{ if } j \text{ is not in the output layer,} \end{cases}$$

where the summation carries over all neurons k in the layer proceeding j. Consider θ_j as the jth input of the network when adjusting the first layer of neurons. The above formula is valid only when $\phi(v)$ is the sigmoid function. Some common stopping criteria are limiting the number of interactions, stopping when the change in C is lower than a certain threshold, stopping when the percent change in C is small enough.

An epoch is the number of iteractions to update the weights once for every pair (x_i, y_i). After each epoch, it is a good ideia to check for overfitting. This is done by evaluating the performance of the network in a set different than the one used to train the network. When the performance in this test set starts to lower, stop training the network.

5.4 Simulation

In this section, we use MLP neural networks to emulate the method of moments in artificial data to evaluate its performance. For a certain family of probability distribution with k

parameters, we generate 1000 example vectors. Each example is generated as follows. The k parameters are randomly chosen from an uniform distribution in an appropriate interval. Once the parameters are chosen, a random sample of 100 observations is generated from the family using those parameters. The first k sample moments are obtained for this sample. The jth sample moment is defined as $n^{-1}\sum_{i=0}^{n} x_i{}^j$. Each example will consist of the k sample moments and the k true values of the parameters. For each batch of examples, 750 will be randomly selected to the training of the MLP, while 250 will be used for evaluating the performance. For each family, we compare the performance of the MLP estimator to the method of moments estimator (MME). The MLP consists of two layers of, respectively, 20 and 10 neurons. The activation function for the neurons in the first and second layers is the sigmoid function. The output neuron has a linear activation function, it just outputs the weighted sum it holds.

If the MME has a simple expression for some parameter in these distributions, the MLP can have a good performance by approximating the MME. This implies that, if there is another estimator other than the MME, such as the GMME, that is able to achieve better performance, in the sense of achieving a smaller mean squared error (MSE) than the MME, we expect to observe a better performance of the MLP estimator when compared to the MME.

5.4.1 Exponential distribution

The first family we simulate is the exponential distribution with parameter $0 < \lambda < 10$. The probability density function (pdf) of the exponential distribution is given by

$$f(x) = \lambda \mathrm{e}^{-\lambda x},\ x > 0.$$

The MME for λ is given by $\overline{x}^{-1}$, with $\overline{x}$ representing the sample mean. Since $h(x) = x^{-1}$ is such a simple function for the MLP to approximate, good results are to be expected. Figure 5.2 shows the results of the trials for the test data set. The histograms and the box plot suggest that the average and median errors in both methods is close to zero and the MLP estimator is more accurate. This agrees with the previous discussion on the expected performance of the MLP estimator. The lines in the scatter plots are the minimum squares line $Estimate = k \cdot error$. In both cases it is very close to the $y = x$ line. The mean error for the MLP estimator is -0.05 and

for the MME it is 0.03. The mean squared error (MSE) for the MLP estimator is 0.2600, for the MME it is 0.3659. This is expected as the MME for λ is not the estimator that minimizes $E[(\widehat{\lambda} - \lambda)^2]$. For instance, the Bayes estimator $(n-2)/(\overline{x}n)$ is known to achieve an expected MSE smaller than that of the MME for the exponential distribution. The error seems to increase as the true value of the parameter does.

5.4.2 Normal distribution

The second test involves the normal distribution with mean represented by μ and variance by σ^2. The MME for the normal distribution's parameters are simply $\overline{x}$ and $s^2 = n^{-1}\sum_{i=1}^{n}(x_i - \overline{x})^2$. The normal distribution is used in this simulation to illustrate a simple case with two parameters. We let μ and σ^2 range from 0 to 10. For μ, the results are presented in Figure 5.3. The results suggest that the MLP estimator is as good as the MME. The same occurs for the estimation of σ, shown in Figure 5.4. The results for the mean error and mean squared error for μ and σ are displayed in table 5.1. The results with two parameters in the normal distribution are good as expected.

Table 5.1 Summary for the errors in the normal distribution estimation.

	MLP		MM	
	Mean error	MSE	Mean error	MSE
μ	≈ 0	0.0589	0.01	0.6200
σ	≈ 0	0.0220	-0.03	0.0280

5.4.3 Exponentiated exponential distribution

The exponentiated exponential (EE) distribution with cdf given by

$$G(x) = \left(1 - e^{-x/\lambda}\right)^{\alpha}, \quad \alpha > 0,\ \lambda > 0,\ x > 0,$$

is used as a non-trivial example of estimation. The MME for its parameters were discussed in, for example, Gupta and Kundu (1999) and there is no explicit expression for them. We let α

Figure 5.2 Results for the exponential distribution.

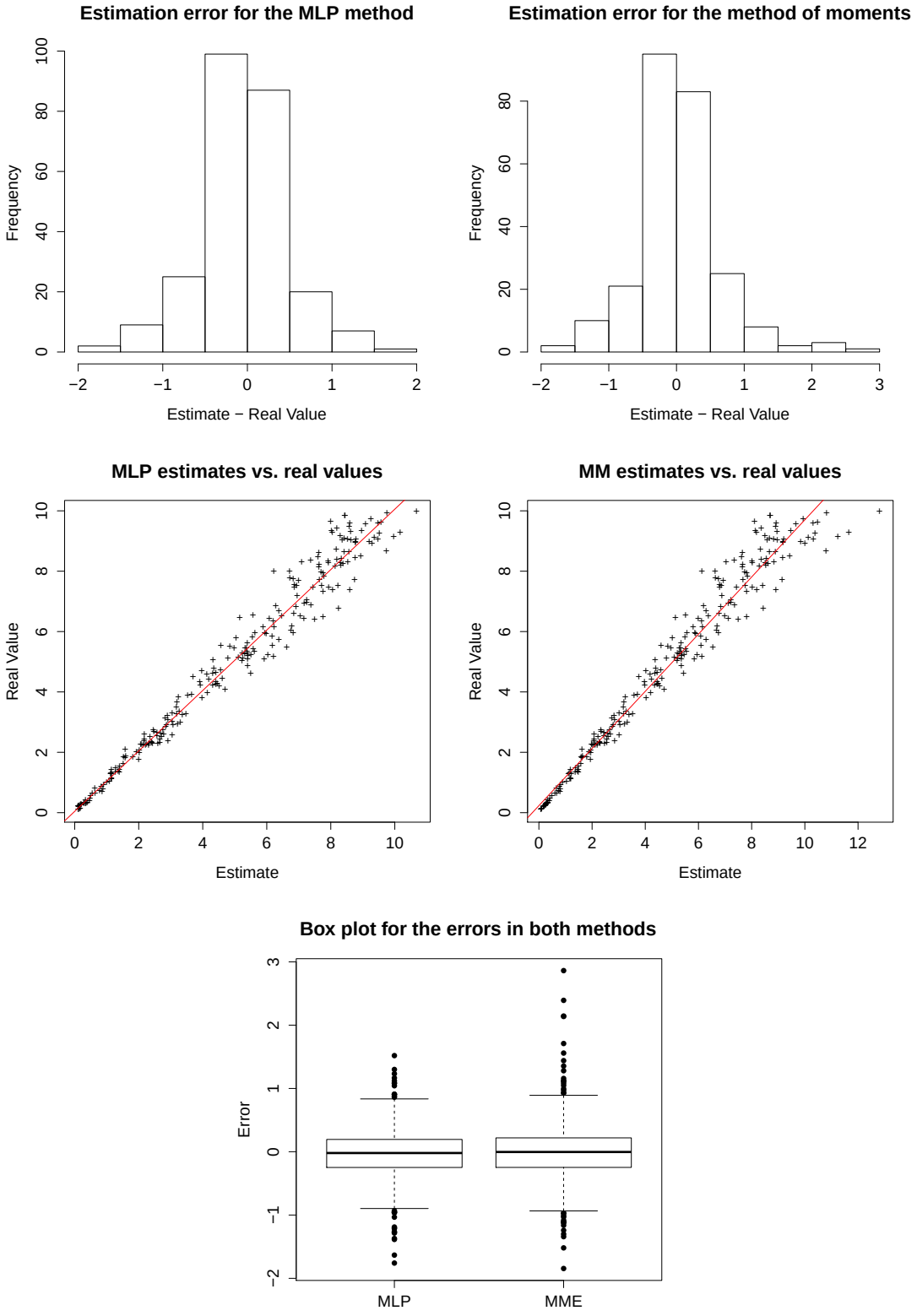


Figure 5.3 Results for the estimation of normal distribution's mean.

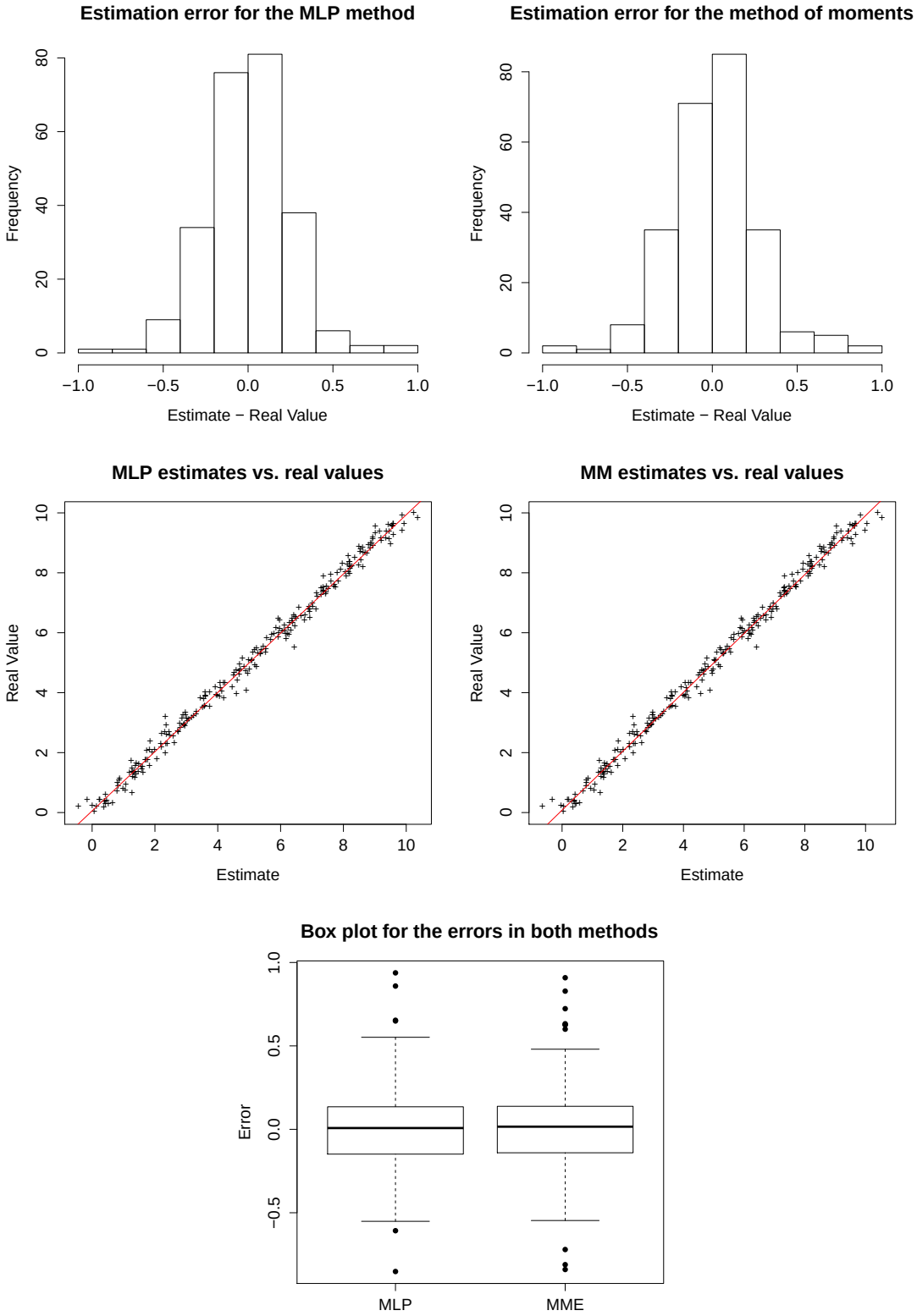


Figure 5.4 Results for the estimation of normal distribution's standard deviation.

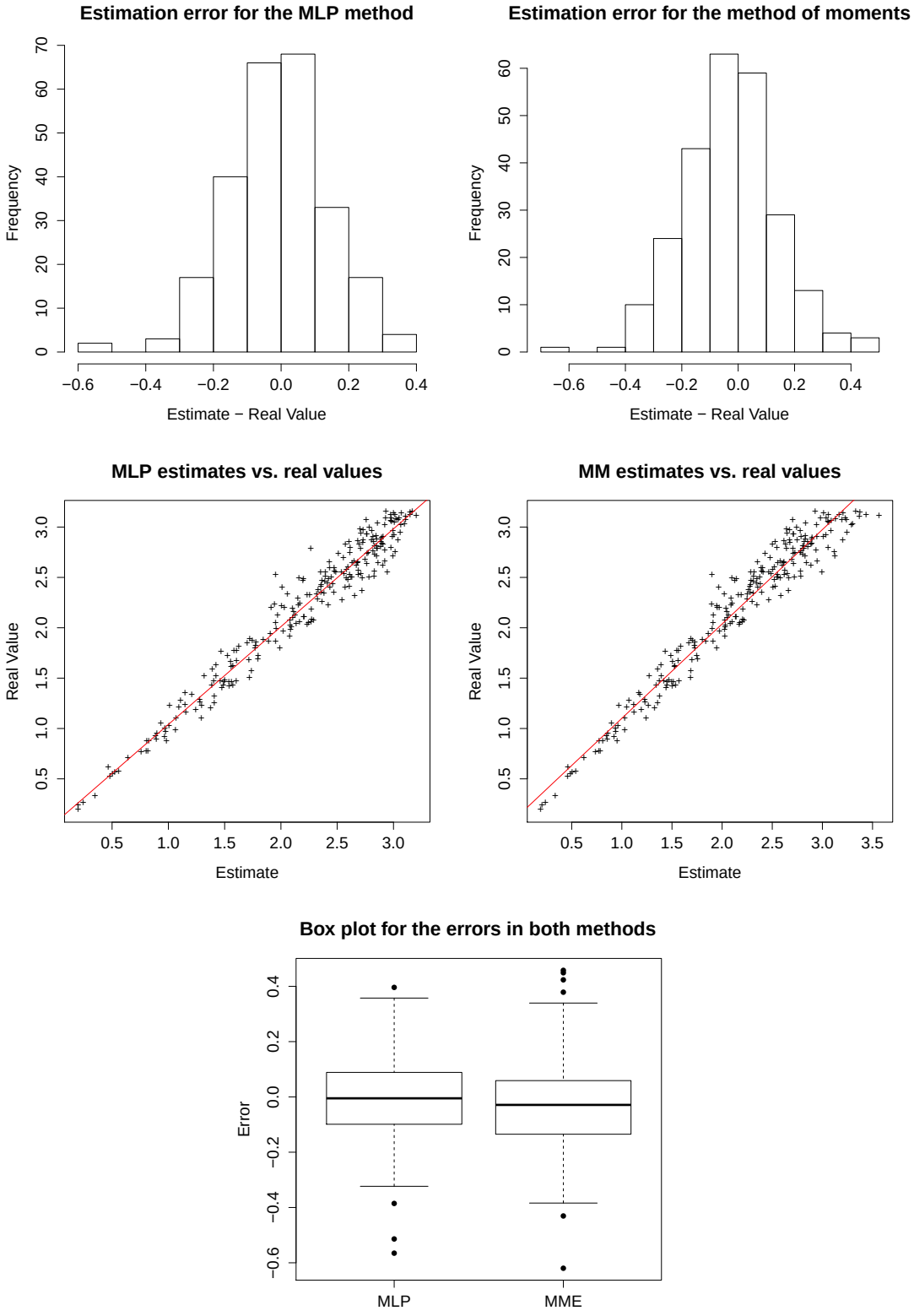
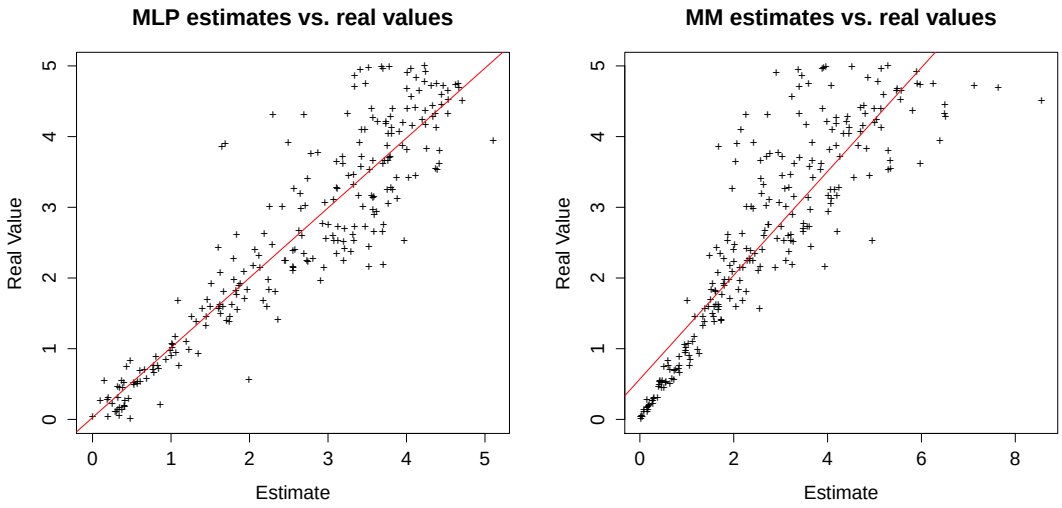


Figure 5.5 Results for the estimation of the shape parameter in the exponentiated exponential distribution.



range from 0 to 5 and λ range from 0 to 1 in this simulation. The inputs of the MLP are the first and second order sample moments. The MME was obtained following the suggestion in Gupta and Kundu (1999). It consists in solving, iteratively, the equation

$$\frac{S}{\overline{X}} = \frac{\sqrt{\psi'(1) - \psi'(\alpha+1)}}{\psi(\alpha+1) - \psi(1)},$$

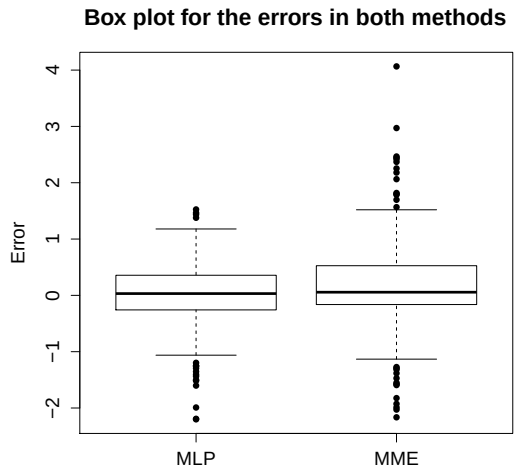
for α and the setting $\widehat{\lambda} = \overline{X}/[\psi(\widehat{\alpha}+1) - \psi(1)]$.

The results for the estimation of α for the MLP and MM estimators is shown in Figure 5.5. The MLP estimator appears to perform better than the MME for α. The box plot in Figure 5.6 suggests that the MLP achieves a more accurate estimate for the value of α. The median errors seem close to zero in both methods. The results for the estimation of λ are very similar and they are omitted in this text. A summary of the estimation is shown in Table 5.2.

Table 5.2 Summary for the errors in the normal distribution estimation.

	MLP		MM	
	Mean error	MSE	Mean error	MSE
α	≈ 0	0.3522	0.16	0.7244
λ	≈ 0	0.0082	≈ 0	0.161

Figure 5.6 Comparison of the shape parameter estimation results in the exponentiated exponential distribution.



5.5 SAR image segmentation - real data application

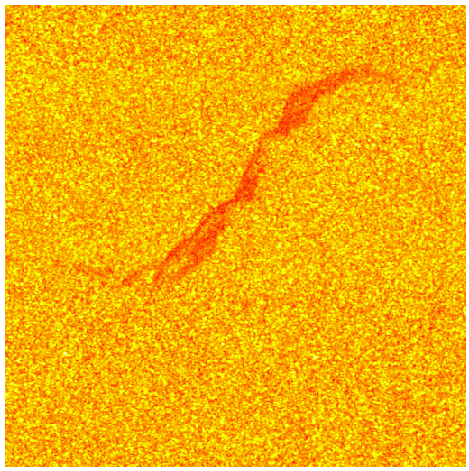
A SAR (synthetic aperture radar) is a device that can be used to obtain very large images of the land. The device is attached to an aircraft and flies over the target area. Antennae in this device emit waves to the ground and capture back the reflected waves. The underlying physical characteristics of the received signals allow for the creation of an image of the area. The problem with this technique is that the reflected wave suffers interference of the incident wave. This interference is usually modelled in a multiplicative fashion and it is called *speckle*. SAR images may be used for several ends. In this application we use the image in Figure 5.7 of the ocean where it is visible an oil slick. Each pixel in this image represents the amplitude of the received signal, which is related to the brightness of the area (recall that the intensity of an wave is proportional to the square of its amplitude). We wish to determine the borders of the oil slick. Such a task is called "segmentation" of the image and the speckle in the signal makes the task harder. There are many ways to achieve this end and a very popular one is based on statistical inference of the characteristics of the surface and of the radar. In Frery et al. (1997) the $\mathscr{G}_A^0$ distribution is used to model SAR image data and discussed in great detail. This model is obtained by assuming different probability distributions for the reflected signal

and the speckle. The $\mathcal{G}_A^0$ model has three parameters and the pdf for this model is given by

$$f(x;N,\gamma,\alpha) = \frac{2N^N\Gamma(N-\alpha)}{\gamma^\alpha\Gamma(-\alpha)\Gamma(N)}\frac{z^{2N-1}}{(\gamma+z^2N)^{N-\alpha}},\ -\alpha,\ \gamma,\ N>0.$$

The parameter N is the number of looks in the image, essentially the number of times the antennae fly over the area. The parameter α is related to the roughness of the surface. It is a key parameter when it comes to segmentation of an image. Values close to zero are typical of highly heterogeneous areas, such as urban areas, while more negative values are typical of very homogeneous areas. The difference if the roughness of the surface in different pixels allows the perception of different objects in the image. The third parameter, γ, is related to the relative power between the reflected and incident signals. Mejail et al. (2000) comment that there

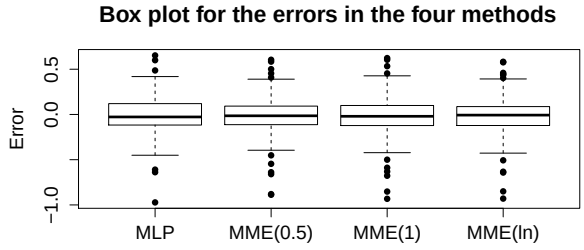
Figure 5.7 SAR image of an oil slick.



usually is a large amount of information on γ and it can be assumed to be known and constant for all the pixels in the image. They propose 3 moment based estimators for α, which they call $\widehat{\alpha}_{1/2}$, $\widehat{\alpha}_1$ and $\widehat{\alpha}_1^{\ln}$. These are, respectively, the solutions of the equations

$$\frac{\Gamma(-\widehat{\alpha}_{1/2})}{\Gamma(-\widehat{\alpha}_{1/2}-1/4)} = (\gamma/N)^{1/4}\frac{\widehat{m}_{1/2}\Gamma(N+1/4)}{\Gamma(N)}$$
$$\frac{\Gamma(-\widehat{\alpha}_{1/2})}{\Gamma(-\widehat{\alpha}_{1/2}-1/2)} = (\gamma/N)^{1/2}\frac{\widehat{m}_{1/2}\Gamma(N+1/2)}{\Gamma(N)}$$
$$\psi(\widehat{\alpha}_1^{\ln}) = \log(\gamma/n)+\psi(N)-2\widehat{m}_1^{\ln},$$

Figure 5.8 Comparison between the MLP and other moment based methods.



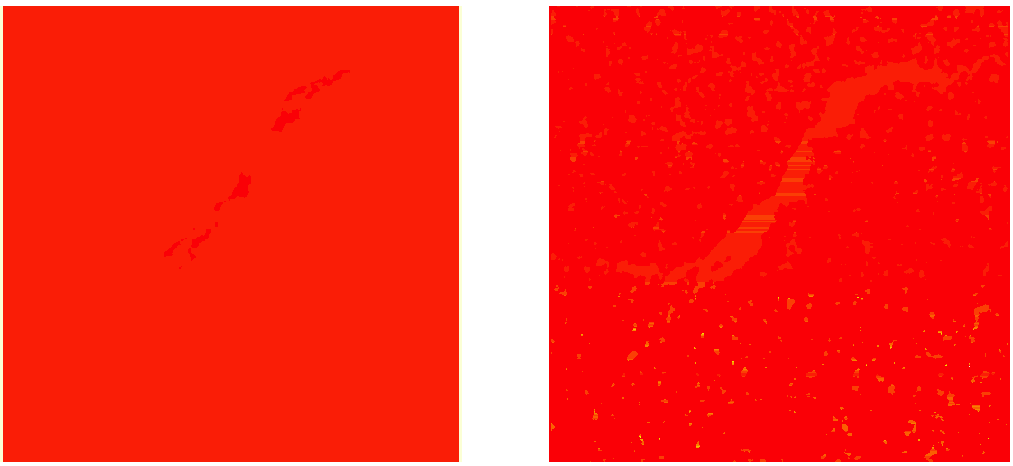
.

where $\widehat{m}_{1/2}$ is the sample mean of the square roots, $\widehat{m}_1$ is the sample mean and $\widehat{m}_1^{\ln}$ is the sample mean of the logarithms of the data.

In this section, we use the MLP estimators to segment the SAR image in Figure 5.7. Before doing so, we compare the performance of the proposed method to $\widehat{\alpha}_{1/2}$, $\widehat{\alpha}_1$, and $\widehat{\alpha}_1^{\ln}$. Mejail et al (2000) evaluate the performance of these estimators by simulating data with N and γ known and estimating α alone. We do the same for the MLP estimator in order to compare its performance to that of the estimators proposed in Mejail et al (2000). We let $N = 4$ and $\gamma = 1$ and we let α range from -0.5 to -5. The sample sizes and the number of examples used to train the MLP are the same as in the previous section. The estimation errors, $\widehat{\alpha} - \alpha$, are calculated for each of the 250 examples in the test data set of the MLP. The results are shown in Figure 5.8. The MLP estimator performs as well as the others in this simulation. However, once the MLP is trained, it is not necessary to solve any non-linear equation for obtaining the estimates.

The image in Figure 5.7 consists of a 512×512 matrix. A MLP was trained for α ranging from -0.5 to -5, γ from 0 to 5 and $N = 4$, which is the number of looks for this image. To segment a SAR image we must assume that the amplitude of the signal associated with every pixel of the image follows a $\mathscr{G}_A^0$ distribution. For each pixel, α must be estimated. Then, every α from every pixel is compared the α of their neighbors in order to identify objects. To estimate every α a 7×7 window centered at the pixel is used. In this application, α is to be estimated approximately 2^{18} times. Avoiding the equations in the MM estimation is useful. The result of the segmentation, by using two different hard limits on the estimated α, is shown in Figure 5.9.

Figure 5.9 Segmented SAR image of an oil slick.



.

5.6 Final remarks

In this chapter, we used a multilayer perception neural network to obtain estimates of parameters in some probability models. This network is trained by presenting several examples of functions of randomly generated them. Once the network is trained, using it to estimate parameters is as easy as feeding the network functions of the data. There is no iterative processes or difficult non-linear systems of equations using this method.

The simulation section provided some information about the performance of the proposed method. The MLP was able to achieve very good results. The SAR image segmentation application was used to illustrate the advantages of having such a fast estimation procedure.

References

[1] Adamidis, K., and Loukas, S., (1998). A lifetime distribution with decreasing failure rate. *Statistics and Probability Letters*, **39:** 35–42.

[2] Akinsete, A., Famoye, F. and Lee, C. (2008). The beta-Pareto distribution. *Statistics*, **42:** 547–563.

[3] Alshawarbeh, E., Lee, C. and Famoye, F. (2012). The beta-Cauchy distribution. *Journal of Probability and Statistical Science*, **10:** 41–57.

[4] Aly, E. and Benkherouf, L. (2011). A new family of distributions based on probability generating functions. *Sankhya B - Applied and Interdisciplinary Statistics*, **73:** 70–80.

[5] Alzaatreh, A., Lee, C. and Famoye, F. (2013). A new method for generating families of continuous distributions. *METRON*, **71**: 63–79.

[6] Anfinsen, T. and Eltoft, T. (2011) Application of the matrix-variate Mellin transform to analysis of polarimetric radar images. *IEEE Transaction in Geoscience and Remote Sensing*, **49**: 2281–2295.

[7] Barreto-Souza, W. and Bakouch, H. S. (2013). A new lifetime model with decreasing failure rate. *Statistics*, **47:**, 465-476

[8] Barrios, R. and Dios, F. (2012). Exponentiated Weibull distribution family under aperture averaging for Gaussian beam waves. *Optics Express*, **20:**, 13055–13064

[9] Chahkandi, M. and Ganjali, M. (2009). On some lifetime distributions with decreasing failure rate. *Computational Statistics & Data Analysis*, **53:** 4433–4330.

[10] Choudhury, A. (2005). A simple derivation of moments of the exponentiated Weibull distribution. *Metrika*, **62:** 17–22.

[11] Cordeiro, G. M. and Nadarajah, S. (2011). Closed form expressions for moments of a class of beta generalized distributions. *Brazilian Journal of Probability and Statistics*, **25:** 14–33.

[12] Cordeiro, G. M., Ortega, E. M. M. and Cunha, D. C. C. (2013), The exponentiated generalized class of distributions, *Journal of Data Science*, **11:**, 1–27.

[13] Cordeiro, G. M., Ortega, E., and Silva, G. (2012). The beta extended Weibull family. *Journal of Probability and Statistical Science*, **10:** 15–40.

[14] Cramé, H. (1946). *Mathematical Methods of Statistics*. Asia Publishing House, Bombay.

[15] Eugene, N., Lee, C., and Famoye, F. (2002). Beta-normal distribution and its applications. *Communication in Statistics: Theory and Methods*, **31:** 497–512.

[16] Fletcher, S. G. and Ponnambalam, K. (1996). Estimation of reservoir yield and storage distribution using moments analysis. *Journal of Hydrology*, **182:** 259–275.

[17] Fischer, M. and Vaughan, D.C. (2010). The Beta-hyperbolic secant (BHS) distribution. *Austrian Journal of Statistics*, **39**: 245–258.

[18] Frery, A. C., Muller, H. J., Yanasse, C. C. F. e Sant'Anna, S. J. S. (1997). A model for extremely heterogeneous clutter. *IEEE transactions on geoscience and remote sensing*, **35:** 648–659.

[19] Gupta, R. C., Gupta, R. D., and Gupta, P. L. (1998). Modeling failure time data by Lehmann alternatives. *Communications in Statistics, Theory and Methods*, **27:** 887–904.

[20] Gupta, R. D. and Kundu, D. (1999). Generalized exponential distributions. *Australian & New Zealand Journal of Statistics*, **41:** 173–188.

[21] Gupta, R. D. and Kundu, D. (2001). Exponentiated exponential distribution: an alternative to gamma and Weibull distributions. *Biometrical Journal*, **43:** 117–130.

[22] Gupta, R. D. and Kundu, D. (2002). Generalized exponential distribution: Statistical inferences. *Journal of Statistical Theory and Applications*, **1:** 101–118.

[23] Hansen, L. P. (1982). Large Sample Properties of Generalized Method of Moments Estimators. *Econometrica*, **50**: 1029–1054.

[24] Hirsch, M. J., Pardalos, P. M. and Resende, M. G. C. (2010). Speeding up continuous GRASP. *Journal of Operational Research*, **205**: 507–521.

[25] Hornik, K. (1991). Approximation Capabilities of Multilayer Feedforward Networks. *Neural Networks*, **4:** 251–257.

[26] Jaynes, E. T. (1957). Information theory and statistical mechanics. *Physical Review*, **106:** 620–630.

[27] Johnson, N. L., Kotz, S. and Balakrishnan N. (1994). *Continuous Univariate Distributions, Vol 1*. 2nd Edition Wiley, New York.

[28] Jones, M. C. (2004). Families of distributions arising from distributions of order statistics. *Test*, **13:** 1–43.

[29] Jones, M. C., and Faddy, M. J. (2004). A skew extension of the t-distribution, with applications. *Journal of the Royal Statistical Society, Series B*, **65**: 159–174.

[30] Kumaraswamy, P. (1980). A generalized probability density function for double-bounded random processes. *Journal of Hydrology*, **46:** 79–88.

[31] Kus, C. (2006). A new lifetime distribution. *Computational Statistics & Data Analysis*, **51:** 4497–4509.

[32] Lehmann, E. L. (1953). The power of rank tests. *Annals of Mathematical Statistics*, **24:** 23–43.

[33] Lu, W. and Shi, D. (2012). A new compounding life distribution: the Weibull–Poisson distribution. *Journal of Applied Statistics*, **39:** 21–38.

[34] Bourguignon, M., Ghosh, I. and Cordeiro, G. M. (2016). General Results for the Transmuted Family of Distributions and New Models, *Brazilian Journal of Probability and Statistics*, **2016:**0–12.

[35] Marques, R. C. P., Medeiros, F. N. and Santos, J. S. (2011). SAR Image Segmentation Based on Level Set Approach and $\mathscr{G}_A^0$ Model. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **34:** 2046–2057.

[36] Marshall, A. W. and Olkin, I. (1997). A new method for adding a parameter to a family of distributions with application to the exponential and Weibull families. *Biometrika*, 84(3):641–652.

[37] Mejail, M. E., Jacobo–Berlles, J., Frery, A. C. and Bustos, O. H. (2000). Parametric roughness estimation in amplitude SAR images under the multiplicative model. *Revista de Teledetección*, **13:** 37–49.

[38] Moors, J.J.A. (1988). A quantile alternative for kurtosis. *Journal of the Royal Statistical Society (Series D)*, **37**, 25–32.

[39] Morais, A. L. and Barreto-Souza, W. (2011). A compound class of Weibull and power series distributions. *Computational Statistics & Data Analysis*, **55:** 1410–1425.

[40] Mudholkar, G. S. and Hutson, A.D. (1996). The exponentiated Weibull family: Some properties and a flood data application. *Communications in Statistics-Theory and Methods*, **25:** 3059–3083.

[41] Mudholkar, G. and Srivastava, D. (1993). Exponentiated Weibull family for analyzing bathtub failure-real data. *IEEE Transaction on Reliability*, **42:** 299–302.

[42] Mudholkar, G. S., Srivastava, D. K., and Freimer, M. (1995). The exponentiated Weibull family: a reanalysis of the bus-motor-failure data. *Technometrics*, **37:** 436–445.

[43] Nadarajah, S., Cordeiro, G. M. and Ortega, E. M. (2015). The The Zografos–Balakrishnan–G Family of Distributions: Mathematical Properties and Applications

Mathematical properties and applications. *Communications in Statistics – Theory and Methods*, **1:** 18–215.

[44] Nadarajah, S. and Gupta, A. K. (2004). The beta Fréchet distribution. *Far East Journal of Theoretical Statistics*, **14:** 15–24.

[45] Nadarajah, S. and Gupta, A. K. (2007). The exponentiated gamma distribution with application to drought data. *Calcutta Statistical Association Bulletin*, **59:** 29–54.

[46] Nadarajah, S. and Kotz, S. (2004). The beta Gumbel distribution. *Mathematical Problems in engineering*, **4:** 323–332.

[47] Nadarajah, S. and Kotz, S. (2006). The beta exponential distribution. *Reliability Engineering & System Safety*, **91:** 689–697.

[48] Nicolas, J. M. (2002). Introduction aux statistique de deuxième espèce: Application des log-moments et des log-cumulants à l'analyse des lois d'images radar. *Traitement du Signal*, **19**: 139–167. In French.

[49] Paranaíba, P. F., Ortega, E. M. M., Cordeiro, G. M., and Pescim, R. R. (2011). The beta Burr XII distribution with application to lifetime data. *Computational Statistics and Data Analysis*, **55:** 1118–1136.

[50] Paranaíba, P. F., Ortega, E. M. M., Cordeiro, G. M., and Pescim, R. R. (2011). The beta Burr XII distribution with application to lifetime data. *Computational Statistics and Data Analysis*, **55:** 1118–1136.

[51] Qian, L. (2012). The Fisher information matrix for the three-parameter exponentiated Weibull distribution under type II censoring. *Statistical Methodology*, **9:**, 320–329.

[52] Ristić, M. M. and Balakrishnan, N. (2012). The gamma–exponentiated exponential distribution. *Journal of Statistical Computation and Simulation*, **8:** 1191–1206.

[53] Salustowicz, R. and Schmidhuber, J. (1997). Probabilistic incremental program evolution. *Evolutionary Computation*, **5:**, 123–141.

[54] Shannon, C. (1948). A mathematical theory of communication. *Bell System Technical Journal*, **27:** 379–432.

[55] Shaw, W. and Buckley, I. (2007). The alchemy of probability distributions: beyond Gram-Charlier expansions, and a skew-kurtotic-normal distribution from a rank transmutation map. Research report.

[56] Silva, R. M. A., Resende, M. G. C., Pardalos, P. M. and Hirsch, M.J. (2011). A Python/C library for bound-constrained global optimization with continuous GRASP. *AT&T Labs Research*. Technical Report. Florham Park, USA.

[57] Smith, R. L., and Naylor, J. C. (1987). A comparison of maximum likelihood and Bayesian estimators for the three-parameter Weibull distribution. *Applied Statistics*, **36**: 358–369.

[58] Tahir, M. and Cordeiro, G. M. (2016). Compounding of distributions: a survey and new generalized classes. *Journal of Statistical Distributions and Applications*, **3:** 13.

[59] Tahir, M. and Nadarajah, S. (2015). Parameter induction in continuous univariate distributions: Well established G-classes. *Anais da Academia Brasileira de Ciências*, **87**: 539–568.

[60] Zografos, K. and Balakrishnan, N. (2009). On families of beta- and generalized gamma-generated distributions and associated inference. *Statistical Methodology*, **6:** 344–362.

APPENDIX A

The C code for the PIPE algorithm

```c
// How to compile:
// gcc tese.c -o tese -std=c99 -L/usr/local/lib  -lgsl -lgslcblas
// -lm -lmatheval

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <assert.h>
#include <matheval.h>
#include <gsl/gsl_rng.h>
#include <gsl/gsl_integration.h>
#include <gsl/gsl_errno.h>

#define N_ELEMENTS 9 //the number of elements in each node of the PPT
#define N_POPULATION 100 //the number of functions in each population
#define N_GENERATIONS 1000 // number of generations
#define BUFFER_SIZE 1024 // size of the buffer for the functions
#define N_DATA 67 // number of observations
#define MAXHEIGHT 4// the max height of the tree representing a function
#define MAXNODES   60// max number of nodes used
#define LEARNING_RATE 0.1 //the learning rate
#define EPS_PIPE 0.1 // epsilon for the learning factor
#define MUTATION_PARAMETER (1e-4) // mutation parameter
#define MUTATION_RATE 0.01 //mutation rate
#define GSL_MYSEED 189273 // seed for the main function

// codes for the elements in each ppt node
#define C_TIMES 0
#define C_DIVIDED 1
#define C_PLUS 2
#define C_MINUS 3
#define C_EXP 4
#define C_LOG 5
#define C_X 6
#define C_NUMBER 7
```

```
36 #define C_POWER 8
37
38 struct List{ //this stores the elements of F
39   char element[20]; //each string stores a function "written" a text
40                     //(e.g. exp, log, +, etc)
41   struct List *next;
42 }*elements;
43 typedef struct List list;
44
45 struct PPT_{ //this stores the probabilities in the PPT in a binary tree
46   float x[N_ELEMENTS]; // the probabilities of choosing a certain
       element
47                        // from the list
48   struct PPT_ *left;
49   struct PPT_ *right;
50 }*PPT;
51 typedef struct PPT_ ppt;
52
53 struct Candidate{ // each of the candidate  functions
54   char function[BUFFER_SIZE]; //the function written as a string
55   int nodes[MAXHEIGHT]; // this keeps track of which nodes of the PPT
56                         // were used in which order
57 };
58 typedef struct Candidate candidate;
59
60 // prints a list
61 // used for development tests
62 void printList(){
63   if (elements==NULL){
64     printf("nothing!\\n");
65   } else {
66     list *l = elements;
67     while(l!=NULL){
68       printf("%s\\n",l->element);
69       l = l->next;
70     }
71   }
72 }
73
74 void add(char *s){ // adds a string to the list of elements of F
75   if (elements==NULL){
76     elements = (list *) malloc(sizeof(list));
77     strcpy(elements->element,s);
78     elements->next = NULL;
```

```
79    } else {
80      list *l = elements;
81      list *lastLink = (list*)malloc(sizeof(list));
82      strcpy(lastLink->element,s);
83      while(l->next!=NULL){
84        l = l->next;
85      }
86      l->next = lastLink;
87    }
88  }
89
90  void get(int i, char* c){ // gets the ith element of the elements of F
91    int actual = 0;
92    list *l = elements;
93    if (elements==NULL){
94      printf("Nothing to get from an empty list!\\n");
95    } else {
96      while (actual<i){
97        if(l->next!=NULL){
98          l = l->next;
99          actual++;
100       } else {
101         printf("No element in the position %d.\\n",i);
102         exit(-1);
103         break;
104       }
105     }
106     strcpy(c,l->element);
107   }
108 }
109
110 // chooses an element from a list based on a vector of probabilities
111 // returns the position of the element (from 0)
112 int sampleElement(float *probs,char *s,gsl_rng * rng){
113   float r = (float) gsl_rng_uniform (rng);
114   float acc = 0.0; // accumulator
115   int i = -1;
116   while(acc<=r \&& acc<0.99){
117     i++;
118     acc = acc + probs[i];
119   }
120   get(i,s);
121   return i;
122 }
```

```
123
124 // end of list functions
125
126 // PPT functions
127
128 // recursivelly populates the PPT
129 // must allocate memory for PPT before using this
130 void createPPT(ppt* temp,float* x,int height){ //creates the PPT with
131                                                 //probabilities given by
                                                      x
132   for(int i = 0;i<N_ELEMENTS;i++){
133     (temp->x)[i] = x[i];
134   }
135
136   // if the tree is not tall enough, go on and add the children
137   if(height>0){
138   // allocate memory for the children
139     temp->left = (ppt*) malloc(sizeof(ppt));
140     temp->right = (ppt*) malloc(sizeof(ppt));
141     createPPT(temp->left,x,height-1);
142     createPPT(temp->right,x,height-1);
143   }
144
145   // when the height reaches zero you're left with
146   // NULL pointers in the children
147   // then recursion ends
148 }
149
150 // helps to print a vector of floats
151 void myprint (float *x){
152   for (int i=0;i<N_ELEMENTS;i++){
153     printf("%.4f ",x[i]);
154   }
155   printf(";\\n");
156 }
157
158 // prints the probabilities in the PPT using inorder
159 void printPPT(ppt *p){
160   if (p != NULL){
161     printf("Probs: ");
162     myprint(p->x);
163     if(p->right!=NULL)
164       printPPT(p->right);
165     if(p->left!=NULL)
```

```
166       printPPT(p->left);
167   }
168 }
169
170 // end of PPT functions
171
172 // generates a candidate from the PPT and stores on c
173 // nodesVisited keeps track of how many nodes were visited
174 // and is passed to recursive calls
175 // each node is registered in nodesUsed as soon as it is selected.
176 void generateFunction(ppt* p, char c[BUFFER_SIZE], int nodesUsed[
    MAXNODES],
177                       int *nodesVisited, int height, gsl_rng * rng){
178   // height indicates the maximum possible height of the tree function
179
180   float r; // random number
181   char aux[5000], auxl[5000], auxr[5000], auxn[100]; // auxiliar string
182
183    // if height is one, then generate a literal or a number
184    if(height==0){
185      float p_number, p_literal; // probabilities of choosing a literal
           or a number
186      float sum = ((p->x)[N_ELEMENTS-1]+(p->x)[N_ELEMENTS-2]);
187      p_number = (p->x)[N_ELEMENTS-1]/sum;
188      p_literal = (p->x)[N_ELEMENTS-2]/sum;
189
190      r = (float) gsl_rng_uniform(rng);
191
192      if(r<p_number){ // choose a number
193        r = (float) gsl_rng_uniform(rng); // generates a number
194        snprintf(auxn,10,"%f",r); // changes the float to string
195        strcpy(c,auxn);
196        nodesUsed[*nodesVisited] = C_NUMBER;
197        (*nodesVisited)++;
198      } else { // choose an x
199        strcpy(c,"x");
200        nodesUsed[*nodesVisited] = C_X;
201        (*nodesVisited)++;
202      }
203
204    } else { // the height is not one, we will use recursion
205      switch(sampleElement(p->x,aux,rng)){
206        case 0: //*
207 //           printf("0\\n");
```

```
208             nodesUsed[*nodesVisited] = C_TIMES;
209             (*nodesVisited)++;
210             generateFunction(p->left,auxl,nodesUsed,nodesVisited,height-1,
                    rng);
211             strcpy(c,"(");
212             strcat(c,auxl);
213             strcat(c,")*(");
214             generateFunction(p->right,auxr,nodesUsed,nodesVisited,height
                    -1,rng);
215             strcat(c,auxr);
216             strcat(c,")");
217             break;
218           case 1: // /
219 //            printf("1\\n");
220             nodesUsed[*nodesVisited] = C_DIVIDED;
221             (*nodesVisited)++;
222             generateFunction(p->left,auxl,nodesUsed,nodesVisited,height-1,
                    rng);
223             strcpy(c,"(");
224             strcat(c,auxl);
225             strcat(c,")/(");
226             generateFunction(p->right,auxr,nodesUsed,nodesVisited,height
                    -1,rng);
227             strcat(c,auxr);
228             strcat(c,")");
229             break;
230           case 2: //+
231 //            printf("2\\n");
232             nodesUsed[*nodesVisited] = C_PLUS;
233             (*nodesVisited)++;
234             generateFunction(p->left,auxl,nodesUsed,nodesVisited,height-1,
                    rng);
235             strcpy(c,"(");
236             strcat(c,auxl);
237             strcat(c,")+(");
238             generateFunction(p->right,auxr,nodesUsed,nodesVisited,height
                    -1,rng);
239             strcat(c,auxr);
240             strcat(c,")");
241             break;
242           case 3: // -
243 //            printf("3\\n");
244             nodesUsed[*nodesVisited] = C_MINUS;
245             (*nodesVisited)++;
```

```
246             generateFunction(p->left,auxl,nodesUsed,nodesVisited,height-1,
                    rng);
247             strcpy(c,"(");
248             strcat(c,auxl);
249             strcat(c,")-(");
250              generateFunction(p->right,auxr,nodesUsed,nodesVisited,height
                    -1,rng);
251             strcat(c,auxr);
252             strcat(c,")");
253             break;
254           case 4: //exp
255 //            printf("4\\n");
256             nodesUsed[*nodesVisited] = C_EXP;
257             (*nodesVisited)++;
258             generateFunction(p->left,auxl,nodesUsed,nodesVisited,height-1,
                    rng);
259             strcpy(c,"exp(");
260             strcat(c,auxl);
261             strcat(c,")");
262             break;
263           case 5: //log
264 //            printf("5\\n");
265             nodesUsed[*nodesVisited] = C_LOG;
266             (*nodesVisited)++;
267             generateFunction(p->left,auxl,nodesUsed,nodesVisited,height-1,
                    rng);
268             strcpy(c,"log(");
269             strcat(c,auxl);
270             strcat(c,")");
271             break;
272           case 6: // x
273 //            printf("6\\n");
274             nodesUsed[*nodesVisited] = C_X;
275             (*nodesVisited)++;
276             strcpy(c,"x");
277             break;
278           case 7: // numero
279  //            printf("0\\n");
280              nodesUsed[*nodesVisited] = C_NUMBER;
281             (*nodesVisited)++;
282             r = (float) gsl_rng_uniform(rng); // generates a number
283           snprintf(auxn,10,"%f",r); // changes the float to string
284             strcpy(c,auxn);
285             break;
```

```
286         case 8: // power
287 //           printf("2\\n");
288           nodesUsed[*nodesVisited] = C_POWER;
289           (*nodesVisited)++;
290           generateFunction(p->left,auxl,nodesUsed,nodesVisited,height-1,
                  rng);
291           strcpy(c,"(");
292           strcat(c,auxl);
293           strcat(c,")\^{}(");
294           generateFunction(p->right,auxr,nodesUsed,nodesVisited,height
                  -1,rng);
295           strcat(c,auxr);
296           strcat(c,")");
297           break;
298       } // end of switch-case
299     }//end of if-else
300 }
301
302 // generates n functions with a given height
303 void generateFunctions(ppt *p, int n, char functions[N_POPULATION][
      BUFFER_SIZE], int nodesUsed[N_POPULATION][MAXNODES], int height,
      gsl_rng *rng){
304   for(int i = 0;i<n;i++){
305     int nodesVisited = 0;
306     generateFunction(p,functions[i],nodesUsed[i],&nodesVisited,height,
          rng);
307   }
308 }
309
310 // converts  several matheval functions to gsl functions
311 void matheval2gsl(gsl_function F[N_POPULATION], char functions[
      N_POPULATION][BUFFER_SIZE]){
312   void *f; //for the matheval to create a function
313   for(int i =0;i<N_POPULATION;i++){
314     f = evaluator_create (functions[i]);
315      assert (f); // functions exists now
316
317     // create a function
318     double function (double x, void *f){
319       return(evaluator_evaluate_x(f,x));
320     }
321     // and place t in a gsl_function structure
322     F[i].function = function;
323     F[i].params = f;
```

```
324   }
325 }
326
327 // calculates the fit of a candidate function
328 float getFit(gsl_function F, float *data, float *dataF){
329   float candidateF[N_DATA];
330   float fit =0;
331   for(int i=0;i<N_DATA;i++){
332     candidateF[i]=GSL_FN_EVAL(&F,data[i]);
333     fit = fit + (candidateF[i]-dataF[i])*(candidateF[i]-dataF[i]);
334   }
335   if(!gsl_isnan(fit)){
336     return fit;
337   }else{
338     return GSL_POSINF;
339   }
340 }
341
342 // binds the functions as strings, the functions as gsl_functions and
       the list of nodes used
343 // as an array of cadidate structs
344 void bindFunctionAndNodes(candidate candidates[N_POPULATION], char c[
       N_POPULATION][BUFFER_SIZE],
345                           gsl_function F[N_POPULATION], int nodesUsed[
                              N_POPULATION][MAXNODES])
346 {
347   for (int i = 0; i<N_POPULATION; i++)  {
348     //TODO
349   }
350 }
351
352 int myCompare(const void *a, const void *b){
353   return (int)(*(float*)a - *(float*)b);
354 }
355
356 void evaluateFit(float fit[N_POPULATION], gsl_function F[N_POPULATION],
       float data[N_DATA], float dataF[N_DATA]){
357   for(int i =0;i<N_POPULATION;i++)
358     fit[i] = getFit(F[i],data,dataF);
359  }
360
361 // finds the index of the best fit
362 int findBestFit (float fit[N_POPULATION]){
363   int bIndex = 0;
```

```
364   for(int i=1; i<N_POPULATION; i++){
365     if(fit[i]<fit[bIndex])
366       bIndex=i;
367   }
368   return bIndex;
369 }
370
371
372 // calculates the probability of a program
373 float getProbProgram(ppt* p, int nodes[MAXNODES],int* currentIndex){
374   float prob; // the probability
375   prob = (p->x)[nodes[*currentIndex]];
376   // if i'm beyond the last node, stop.
377   if(*currentIndex==MAXNODES-1)
378     return(prob);
379
380   // if the current function from this node was  *,/,+ or - then get the
        prob of the right and left functions
381   if(nodes[*currentIndex]<4){
382     *currentIndex = *currentIndex + 1;
383     prob = prob*getProbProgram(p->left,nodes,currentIndex);
384     *currentIndex = *currentIndex + 1;
385     prob = prob*getProbProgram(p->right,nodes,currentIndex);
386   } else if(nodes[*currentIndex]==4||nodes[*currentIndex]==5){//if i got
        an exp or a log
387     // then get the probability of the only child
388     *currentIndex = *currentIndex + 1;
389     prob = prob*getProbProgram(p->left,nodes,currentIndex);
390   }
391   return prob;
392 }
393
394 // adapts the ppt to raise the probability of getting the best program
395 void adapt_PPT_towards2(ppt* p, int prog_b_nodes[MAXNODES], float
      p_prog_b, float p_target, int* currentIndex){
396   // raise the probability for this node's selected element
397   (p->x)[prog_b_nodes[*currentIndex]] += LEARNING_RATE*(1-(p->x)[
        prog_b_nodes[*currentIndex]]);
398   // if i'm beyond the last node, stop.
399   if(*currentIndex==MAXNODES)
400     return;
401
402   // if the current function from this node was  *,/,+ or - then get the
        prob of the right and left functions
```

```
403   if(prog_b_nodes[*currentIndex]<4){
404     *currentIndex = *currentIndex + 1;
405     adapt_PPT_towards2(p->left,prog_b_nodes,p_prog_b,p_target,
          currentIndex);
406     *currentIndex = *currentIndex + 1;
407     adapt_PPT_towards2(p->right,prog_b_nodes,p_prog_b,p_target,
          currentIndex);
408   } else if(prog_b_nodes[*currentIndex]==4||prog_b_nodes[*currentIndex
        ]==5){//if i got an exp or a log
409     // then get the probability of the only child
410     *currentIndex = *currentIndex + 1;
411     adapt_PPT_towards2(p->left,prog_b_nodes,p_prog_b,p_target,
          currentIndex);
412   }
413   return;
414 }
415
416 // adapts the ppt to raise the probability of getting the best program
417 void adapt_PPT_towards(int prog_b_nodes[MAXNODES], float p_prog_b, float
      p_target){
418   int currentIndex;
419   while(p_prog_b<p_target){
420     currentIndex=0;
421     adapt_PPT_towards2(PPT,prog_b_nodes,p_prog_b,p_target,&currentIndex)
          ;
422     currentIndex=0;
423     p_prog_b = getProbProgram(PPT, prog_b_nodes,&currentIndex);
424   }
425 }
426
427 // mutates the PPT
428 void mutate_PPT(ppt* p, float p_prog_b, gsl_rng *rng){
429   float mutation_prob = MUTATION_PARAMETER/sqrt(p_prog_b);
430   float r;
431   for (int i=0; i<N_ELEMENTS;i++){
432     r = gsl_rng_uniform(rng);
433     if(r<mutation_prob){
434       (p->x)[i] += MUTATION_RATE*(1-(p->x)[i]);
435     }
436   }
437   if(p->left!=NULL)
438     mutate_PPT(p->left,p_prog_b,rng);
439
440   if(p->right!=NULL)
```

```
441     mutate_PPT(p->right,p_prog_b,rng);
442
443   return;
444 }
445
446 // keeps the sum of the probabilities equal to 1
447 void normalize_PPT(ppt* p){
448   float sum=0.0;
449   for (int i=0; i<N_ELEMENTS;i++){
450     sum+=(p->x)[i];
451   }
452   for (int i=0; i<N_ELEMENTS;i++){
453     (p->x)[i] /= sum;
454   }
455   if(p->left!=NULL)
456     normalize_PPT(p->left);
457   if(p->right!=NULL)
458     normalize_PPT(p->right);
459   return;
460 }
461
462 int main (void){
463   // setting up the random number generator
464   const gsl_rng_type * T;
465   gsl_rng * r;
466   gsl_rng_env_setup();
467   T = gsl_rng_default;
468   r = gsl_rng_alloc (T);
469   //gsl_rng_set (r, 248);
470   gsl_rng_set (r, GSL_MYSEED);
471
472   // the probabilities for selecting each node
473   float x
        [9]={1.0/9.0,1.0/9.0,1.0/9.0,1.0/9.0,1.0/9.0,1.0/9.0,1.0/9.0,1.0/9.0,1.0/9.0};
474
475   // the elements of the list of node labels
476   add("*");
477   add("/");
478   add("+");
479   add("-");
480   add("exp");
481   add("log");
482   add("x");
```

```
483   add("number");
484   add("power");
485
486   // It is necessary to allocate space for the PPT before creating it
487   PPT = (ppt*)malloc(sizeof(ppt));
488
489   // creating the PPT
490   createPPT(PPT,x,MAXHEIGHT);
491
492  // the data, sorted
493   float data[N_DATA] =
          {0.1,0.4,0.4,0.6,0.6,0.7,1.1,1.1,1.1,1.4,1.5,1.7,1.7,1.7,1.7,1.9,2.2,2.2,2.5,2.
494   // the empirical distribution of the data
495   float dataF[N_DATA] =
          {0.01492537,0.04477612,0.04477612,0.07462687,0.07462687,0.08955224,0.1343284,0.
496
497   // these variables represent the best program across generations (the
          elite)
498   char functionEL[BUFFER_SIZE]; // its expression as a string
499   int nodesEL[MAXNODES]; // which elements from the nodes were used to
          create it
500   gsl_function FEL; // its expression as a gsl function
501   float fitEL = GSL_POSINF; // its fit value
502   float p_prog_el =0.0;
503
504   // repeat for every generation
505   for(int generation=0; generation<N_GENERATIONS; generation++){
506     // setup the describers of the programs in this generation
507
508     // functions generate by the PPT for this generation
509     char functions[N_POPULATION][BUFFER_SIZE];
510
511     // the nodes used in each function of this generation
512     int nodesUsed[N_POPULATION][MAXNODES];
513     // -1 in a node slot means it was not used.
514     for(int i = 0; i< N_POPULATION;i++){
515       for (int j = 0; j<MAXNODES; j++)
516         nodesUsed[i][j] = -1;
517     }
518
519     // functions created before but now as gsl_function structures
520     gsl_function F[N_POPULATION];
```

```
521
522     // the fit values for this generation
523     float fit[N_POPULATION];
524
525     // the index of the best program at this generation
526     int bIndex;
527
528     // end of setting up the describers for the programs
529
530     // generate the mathematical expressions for the functions as
            strings
531     generateFunctions(PPT,N_POPULATION,functions,nodesUsed,MAXHEIGHT,r);
532
533     // generate the gsl_functions from the strings
534     matheval2gsl(F,functions);
535
536     // evaluate the fitness of each individual in this generation
537     evaluateFit(fit, F, data, dataF);
538
539     // chose the most fit in this generation
540     bIndex = findBestFit(fit);
541
542     // this is used to keep track of which node we are visiting during
            the adaptation phase
543     int currentIndex =0;
544
545     // get the probability of getting the best program from the PPT
546     float p_prog_b = getProbProgram(PPT, nodesUsed[bIndex], &
            currentIndex);
547
548     // if a better than the elite program is found in this generation,
            store it
549     if(fit[bIndex]<fitEL){
550       p_prog_el = p_prog_b;
551       strcpy(functionEL,functions[bIndex]);
552       for(int i=0; i<MAXNODES;i++)
553         nodesEL[i] = nodesUsed[bIndex][i];
554       FEL = F[bIndex];
555       fitEL = fit[bIndex];
556     }
557     float p_target = p_prog_b + (1-p_prog_b)*LEARNING_RATE*((EPS_PIPE+
            p_prog_el)/(EPS_PIPE+p_prog_b));
558
559     // adapts the ppt to raise the probability of getting the best
```

```
        program
560     adapt_PPT_towards(nodesUsed[bIndex], p_prog_b, p_target);
561
562     // mutate the PPT
563     mutate_PPT(PPT,p_prog_b,r);
564
565     // normalize the PPT
566     normalize_PPT(PPT);
567
568     //printf(".");
569   } // end of generation loop
570
571   void* f = evaluator_create(functionEL);
572   void* f_prim = evaluator_derivative_x (f);
573   printf("\\nThe best fit was for the program ");
574   for(int i = 0; i< MAXNODES;i++)
575     printf("%d",nodesEL[i]);
576   printf(" and it was %.4f.\\n",fitEL);
577   printf(" %s with probability %.12f.\\n\\n",functionEL,p_prog_el);
578   printf(" Its density is %s.\\n", evaluator_get_string (f_prim));
579
580   float loglik =0;
581   float tempLogLik = 0;
582   for(int i=0;i<N_DATA;i++){
583     tempLogLik = log(evaluator_evaluate_x(f,data[i]));
584     if(tempLogLik!=tempLogLik){ // acontece se tempLogLik eh NaN
585       loglik = loglik +1;
586     }else {
587       loglik = loglik + tempLogLik;
588     }
589   }
590   printf(" Its loglikelihood is %.4f.\\n", loglik);
591
592   gsl_rng_free(r);
593 }
594 }
```

APPENDIX B

SAR image data

This is the data set used in the SAR image modeling example.

0.01149687, 0.01250427, 0.01528162, 0.01570864, 0.01802599, 0.01894287, 0.01911615, 0.01935418, 0.01992964, 0.02017052, 0.02051137, 0.02082554, 0.02185369, 0.02260664, 0.02293175, 0.02296903, 0.02317232, 0.02338895, 0.02358412, 0.02363688, 0.02396332, 0.02407404, 0.02585229, 0.02647153, 0.02665159, 0.02667463, 0.02721631, 0.02775543, 0.02777823, 0.02784315, 0.02792585, 0.02833025, 0.02839673, 0.02856023, 0.02861129, 0.02892729, 0.02931118, 0.02941171, 0.02942552, 0.03024757, 0.03101604, 0.03141554, 0.03224484, 0.03228633, 0.03232158, 0.03384829, 0.03402714, 0.03424702, 0.03427167, 0.03445653, 0.03453897, 0.03474018, 0.03501506, 0.03578667, 0.03598676, 0.03752478, 0.03754972, 0.03763689, 0.03803634, 0.0382402, 0.03879451, 0.03893850, 0.03989288, 0.04063699, 0.04111884, 0.04164984, 0.0416827, 0.04230256, 0.0427892, 0.04341392, 0.04367883, 0.04526951, 0.04538165, 0.0458550, 0.0462939, 0.04638059, 0.04639203, 0.04647379, 0.04675854, 0.04694617, 0.04768521, 0.04783208, 0.0483232, 0.04891223, 0.04971199, 0. 05092829, 0.05177016, 0.05190274, 0.05229843, 0.05260086, 0.05274564, 0.05385335, 0.0539581, 0.0544991, 0.05508701, 0.05515739, 0.05547253, 0.05562469, 0.05611064, 0.05686707, 0.05705985, 0.05840242, 0.05941767, 0.05983544, 0.0608492, 0.06187658, 0.06204657, 0.06370583, 0.06403044, 0.06442861, 0.06560329, 0.0661218, 0.06653712, 0.06816631, 0.07138552, 0.07589816, 0.07643031, 0.0774606, 0.07915612, 0.08028217, 0.08059592, 0.0847016, 0.08566783, 0.09340851, 0.09429808, 0.09665179, 0.1003492, 0.1060528, 0.1092498, 0.1206784, 0.1257918.

APPENDIX C

Maximum entropy characterization for the Beta-L2-G family

Consider the Kullback-Leibler divergence between the density functions $f(x)$ and $h(x)$ given by

$$D(h,f) = \int_{-\infty}^{+\infty} h(x) \log\left[\frac{h(x)}{f(x)}\right] dx.$$

The Gibbs' inequality implies $D(h,f) \geq 0$, where the equality hold iff $h(x)$ and $f(x)$ are equal almost everywhere. Then,

$$\begin{aligned}
0 &\leq \int_{-\infty}^{+\infty} h(x) \log\left[\frac{h(x)}{f(x)}\right] dx, \\
0 &\leq \int_{-\infty}^{+\infty} h(x) \log[h(x)] dx - \int_{-\infty}^{+\infty} h(x) \log[f(x)] dx, \\
\mathscr{H}_{Sh}(h) &\leq -\int_{-\infty}^{+\infty} h(x) \log[f(x)] dx,
\end{aligned}$$

Notice that

$$\log[f(x)] = \log\left[\frac{c}{B(a,b)}\right] + (bc-1)\log[1-G(x)] + (a-1)\log\{1-[1-G(x)^c]\} + \log[g(x)].$$

For the calculations of the Shannon entropy for the Beta-L2-G family, we require $\mathbb{E}\{\log[1-G(X)]\}$ and $\mathbb{E}[\log\{1-[1-G(x)^c]\}]$. After some algebraic manipulation, we obtain $\mathbb{E}\{\log[1-G(X)]\} = \psi(b) - \psi(a+b)$ and $\mathbb{E}[\log\{1-[1-G(X)^c]\}] = \psi(a) - \psi(a+b)$. For $\mathbb{E}\{\log[g(X)]\}$, the substitution $z = [1-G(x)]^c$ gives $\mathbb{E}\{\log[g(X)]\} = \mathbb{E}_Z\{\log[G^{-1}(1-Z^{-c})]\}$, where $Z \sim Beta(b,a)$. Thus, an alternative expression for the Shannon's entropy of the Beta-L2-G class is given by

$$\mathscr{H}_{Sh}(f) = -\log\left[\frac{c}{B(a,b)}\right] - (bc-1)[\psi(a) - \psi(a+b)] - (a-1)[\psi(b) - \psi(a+b) - \mathbb{E}_Z\{\log[G^{-1}(1-Z^{-c})]\}].$$

Under the imposed constraints and the definition of Z given before, the right hand side of the last inequality is precisely $\mathscr{H}_{Sh}(f)$ so that, for the equality to hold, $f(x)$ equals $h(x)$ almost everywhere.

APPENDIX D

Forest coverage data

This is the data set used in the application example for the QTMO-Kumarasawamy distribution.

2.07, 28.33, 0.62, 90.00, 35.55, 46.90, 60.00, 22.22, 10.74, 9.29, 19.43, 47.14, 11.32, 51.44, 1.40, 11.07, 18.60, 41.59, 22.39, 61.06, 41.23, 20.00, 69.12, 52.74, 42.67, 20.02, 62.40, 25.00, 72.10, 36.14, 20.64, 6.69, 57.18, 42.13, 34.10, 21.09, 50.00, 36.28, 9.15, 21.67, 21.94, 54.52, 1.61, 65.62, 66.66, 51.01, 32.71, 34.33, 26.13, 18.72, 34.39, 67.98, 12.82, 0.25, 60.00, 40.75, 35.63, 0.07, 13.85, 57.96, 15.16, 52.30, 11.21, 55.50, 72.86, 29.00, 98.32, 42.34, 85.38, 48.00, 39.45, 31.75, 21.71, 30.27, 50.00, 39.75, 47.27, 33.72, 26.63, 71.90, 77.24, 3.66, 46.40, 22.64, 0.29, 23.01, 52.12, 6.80, 1.88, 10.72, 7.11, 31.10, 31.11, 68.52, 1.11, 1.22, 6.09, 14.81, 0.33, 4.97, 68.24, 53.84, 13.39, 1.44, 44.94, 0.12, 43.75, 34.46, 33.59, 21.58, 34.40, 62.26, 3.33, 10.23, 72.22, 45.71, 0.23, 17.24, 37.83, 33.33, 6.95, 40.37, 20.00, 11.49, 49.62, 48.32, 8.85, 25.42, 10.77, 1.25, 45.89, 30.88, 25.65, 0.95, 9.92, 73.07, 65.21, 33.07, 0.01, 2.18, 86.95, 43.67, 63.43, 44.25, 53.11, 25.70, 30.48, 38.11, 62.23, 11.74, 35.20, 28.58, 49.39, 17.63, 6.45, 42.30, 77.04, 13.04, 69.23, 60.42, 28.12, 0.48, 44.01, 31.01, 89.13, 38.06, 2.89, 40.18, 62.21, 79.06, 10.75, 7.60, 36.40, 28.77, 1.49, 29.43, 94.60, 32.73, 68.73, 31.00, 2.67, 2.92, 37.13, 39.24, 49.89, 5.27, 12.50, 44.05, 6.47, 14.72, 8.78, 79.06, 33.33, 15.15, 16.75, 3.79, 11.88, 37.73, 33.18, 58.82, 9.96, 7.70, 36.06, 52.46, 44.49, 40.00, 2.65, 1.03, 66.54, 40.38.