

2001

Algorithms for operations on probability distributions in a computer algebra system

Diane Lynn Evans
College of William & Mary - Arts & Sciences

Follow this and additional works at: <https://scholarworks.wm.edu/etd>



Part of the [Mathematics Commons](#), and the [Statistics and Probability Commons](#)

Recommended Citation

Evans, Diane Lynn, "Algorithms for operations on probability distributions in a computer algebra system" (2001). *Dissertations, Theses, and Masters Projects*. William & Mary. Paper 1539623382.
<https://dx.doi.org/doi:10.21220/s2-bath-8582>

This Dissertation is brought to you for free and open access by the Theses, Dissertations, & Master Projects at W&M ScholarWorks. It has been accepted for inclusion in Dissertations, Theses, and Masters Projects by an authorized administrator of W&M ScholarWorks. For more information, please contact scholarworks@wm.edu.

ALGORITHMS FOR OPERATIONS ON
PROBABILITY DISTRIBUTIONS
IN A COMPUTER ALGEBRA SYSTEM

A Dissertation

Presented to
The Faculty of the Department of Applied Science
The College of William & Mary in Virginia

In Partial Fulfillment
Of the Requirements for the Degree of
Doctor of Philosophy

by
Diane Lynn Evans
July 2001

UMI Number: 3026405

Copyright 2001 by
Evans, Diane Lynn

All rights reserved.

UMI[®]

UMI Microform 3026405

Copyright 2001 by Bell & Howell Information and Learning Company.

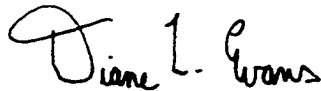
All rights reserved. This microform edition is protected against
unauthorized copying under Title 17, United States Code.

Bell & Howell Information and Learning Company
300 North Zeeb Road
P.O. Box 1346
Ann Arbor, MI 48106-1346

APPROVAL SHEET

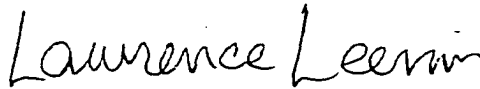
This dissertation is submitted in partial fulfillment of
the requirements for the Degree of

Doctor of Philosophy

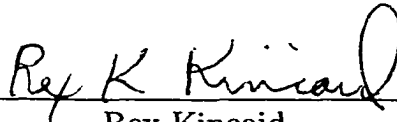


Diane L. Evans, Author

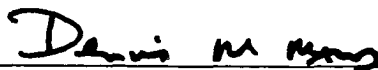
APPROVED, July 2001



Lawrence Leemis



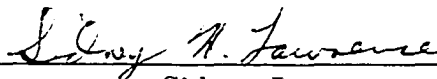
Rex Kincaid



Dennis Manos



John Drew



Sidney Lawrence
Mathematics Department

Contents

Acknowledgements	v
List of Tables	vi
List of Figures	vii
Abstract	x
1 Introduction	2
1.1 Notation and Nomenclature	11
1.2 Introductory Examples	12
2 Data Structure	17
2.1 Standard Discrete Data Structure Formats	27
2.2 The Six Functional Representations	33
2.3 Algorithms for Fundamental Procedures	51
3 Order Statistics	56
3.1 Implementation for Discrete Populations	58
3.2 Examples	70
3.3 Range Statistics	79
3.4 Eliminating Resampling Error in Bootstrapping	88
4 Convolutions and Products	97
4.1 Conceptual Framework	104
4.2 Algorithm	117
4.3 Implementation	120
4.4 Examples	123
4.5 Products of Random Variables with Finite Supports	133
5 Transformations	145
5.1 Theory	146
5.2 Implementation	149
5.3 Applications	164

6	Minimums and Maximums	170
6.1	PDF of the Minimum	172
6.2	PDF of the Maximum	185
7	Algorithms for Operations on Continuous Distributions	195
7.1	Existence Conditions for PDFs	195
7.2	Method of Moments Estimation	200
7.3	Maximum Likelihood Estimation with Right Censoring	207
7.4	Mixture and Truncate Procedures	212
8	Survival Distributions Satisfying Benford's Law	218
8.1	Benford's Law	218
8.2	Parametric Survival Distributions	220
8.3	Conditions for Conformance to Benford's Law	222
8.4	Variate Generation	230
8.5	Conclusions	232
9	Input Modeling	233
9.1	Examples	234
9.2	Further work	245
10	APPLications	247
10.1	Kolmogorov–Smirnov Test Statistic for Estimated Parameters	247
10.2	Others	258
11	Future Work	271
A	Algorithm for OrderStat	274
B	Maple Code for NextCombination and NextPermutation	277
C	Determining Candidate Sums for the Heap	282
D	Algorithm for BruteForceMethod	284
E	Algorithm for MovingHeapMethod	285
F	APPL Code for Benford	287
	Bibliography	288

Acknowledgements

I would like to thank:

My committee members: Dr. Drew, Dr. Kincaid, Dr. Manos, and Dr. Lawrence for their careful reading and suggestions of my dissertation and for being great instructors, both in and out of the classroom;

Dr. Andrew Glen for allowing me to become part of “APPL” and showing me the ropes to becoming a Maple programmer;

The Operations Research faculty at The College of William & Mary for outstanding instruction and a strong probability and statistics foundation;

Dr. Frank Carroll for being my mentor, friend, and mathematical “sounding board” for many years;

The Clare Boothe Luce foundation for their generous fellowship that allowed me to continue my education and have the freedom to delve into my research;

Dr. Larry Leemis for being himself: an excellent teacher, researcher, and advisor. I have spent three of the best years of my life working with him and will always admire and respect him in many ways. The time he has spent with me will always be appreciated, and I hope that someday I may also make such a positive impact, mathematically and otherwise, on another person’s life.

List of Tables

1.1	Observed horse kick fatalities.	15
2.1	Discrete random variable support categories.	24
2.2	The six functional representations of a random variable X	34
2.3	Distribution representation relationships.	35
3.1	Categorization of discrete order statistics with associated examples.	71
3.2	Rat survival data.	90
3.3	Bootstrap estimates of the standard error of the median.	91
3.4	Bootstrap estimates of the standard error of the mean.	93
4.1	Comparison of <code>BruteForceMethod</code> and <code>MovingHeapMethod</code>	123
4.2	Probability table for a convolution.	130
4.3	The exact probabilities and normal PDF approximations for $\Pr(S = s)$ for $s = 7, 8, \dots, 21$	131
5.1	Categories for computing the PDF of the random variable $Y = g(X)$ when X is a discrete random variable with support Ω_X in a <i>Dot</i> support format.	157
5.2	Life tests on a three-component system.	168
6.1	Categories for computing the PDF of the minimum of two independent, non-identically distributed random variables X and Y	171
7.1	Maximum 24-hour precipitation for 36 inland hurricanes (1900–1969).	204
8.1	Conformance to Benford’s law for parametric survival distributions.	221
9.1	Kolmogorov–Smirnov test statistic values for various distributions that were fit to the ball bearing failure times in APPL via maximum likeli- hood estimation.	241
10.1	Variance of a truncated standard normal distribution T for increasing values of the lower truncation point t	269

List of Figures

1.1	APPL tree.	7
1.2	Actual and estimated PDF for the HorseKickFatalities data.	16
2.1	PDF for Example 2.14.	36
2.2	CDF for Example 2.22.	55
3.1	Categorization of discrete order statistics.	59
3.2	<i>ProbStorage</i> array for a sample drawn without replacement from a distribution with finite support.	67
3.3	Support values associated with the joint distribution of X_1 and X_2	69
4.1	Array A with active cell (1, 1), which contains the entry $A_{1,1} = -5$	110
4.2	Array A after $A_{1,1}$ has been removed and added to the one-dimensional sum array s	111
4.3	Six binary trees.	112
4.4	Heap H containing entries $A_{1,2} = -3$ and $A_{2,1} = -2$	113
4.5	Array A after $A_{1,2} = -3$ is removed and appended to s	113
4.6	Array A with active cells (1, 3) and (2, 1).	114
4.7	Heap H containing entries $A_{2,2} = -2$ and $A_{1,3} = 0$	114
4.8	Array A after $A_{2,2} = -2$ is removed.	115
4.9	Array A with active cells (1, 3), (2, 2), and (3, 1).	116
4.10	Heap H with entries $A_{1,3} = 0$, $A_{2,2} = 0$ and $A_{1,3} = 2$	116
4.11	Array A with its seventeenth active cell (3, 4).	117
4.12	Array A split into four quadrants for the product algorithm.	134
4.13	Product array A for subcase two.	137
4.14	Product array A for subcase three.	139
4.15	Product arrays A_- and A_+ for subcase five.	140
4.16	Dueling product arrays A_{1-} and A_{2-} , and A_{1+} and A_{2+} for subcase nine.	144
4.17	The <code>MovingHeapDuelMethod</code> as it progresses simultaneously through arrays A_{1-} and A_{2-} for subcase nine.	144
5.1	The transformation $Y = g(X) = (X - 2)^2$ for $X = 0, 1, 2, 3, 4$	148
6.1	The PDF of the minimum when a four-sided die is rolled twice.	173
6.2	The minimum values $\Omega_Z = \{1, 4, 5, 6\}$ for X and Y in Example 6.3.	176

6.3	The support values for a geometric random variable with support $\Omega_X = \{1, 2, \dots\}$ and a negative binomial random variable with support $\Omega_Y = \{2, 3, \dots\}$	179
6.4	The support values for the random variable X with infinite support and the random variable Y with finite support, where $\min\{\Omega_X\} = \min\{\Omega_Y\}$	181
6.5	The support values for the random variable X with infinite support and the random variable Y with finite support, where $\min\{\Omega_X\} > \min\{\Omega_Y\}$	183
6.6	The support values for the random variable X with infinite support and the random variable Y with finite support, where $\min\{\Omega_X\} < \min\{\Omega_Y\}$	184
6.7	The maximum values $\{4, 5, 6, 7, 9\}$ for X and Y in Example 6.10.	188
6.8	The support values for the random variable X with infinite support and the random variable Y with finite support, where $\min\{\Omega_X\} = \min\{\Omega_Y\}$	190
6.9	The support values for the random variable X with infinite support and the random variable Y with finite support, where $\min\{\Omega_X\} < \min\{\Omega_Y\}$	193
7.1	The graph of $f(x) = 3 x - 1$ for $-1 \leq x \leq 1$	197
7.2	The graph of $f(x) = 1.0002 x - 1 - 0.0001$ for $0.9999 \leq x \leq 1.0001$	200
7.3	Empirical and fitted exponential CDFs for the ball bearing data set.	206
9.1	Coefficient of variation, γ , versus skewness, γ_3 , for the gamma, Weibull, log normal, and log logistic distributions.	235
9.2	Empirical and fitted Weibull CDFs (using the method of moments) for the ball bearing data set.	236
9.3	Empirical and reciprocal exponential fitted CDFs for the ball bearing failure times.	238
9.4	Q-Q plot of ball bearing failure times with fitted (method of moments) Weibull distribution.	239
9.5	P-P plot of ball bearing failure times with fitted (method of moments) Weibull distribution.	240
9.6	Product-limit survivor function estimate and fitted Weibull survivor function for the 6-MP treatment group.	242
9.7	Cumulative intensity function estimate and fitted power law intensity function for the CarFailures data.	244
10.1	The empirical and fitted exponential distribution for one data value x_1	249
10.2	The empirical and fitted exponential distribution for two data values $x_{(1)}$ and $x_{(2)}$	250
10.3	Lengths A, B, C , and D from Figure 10.2 for $0 < y \leq 1$	251
10.4	$D_2 = \max\{A, B, C, D\}$	255

10.5	The empirical CDF of Sample and the theoretical $U(0, 1)$ CDF. . . .	260
10.6	Empirical CDF of 1000 Kolmogorov–Smirnov statistics and the theoretical Kolmogorov–Smirnov CDF for $n = 5$	261
10.7	Power curves for the test statistic $Y = X_1 + X_2 + X_3$ (solid line) and test statistic $X_{(3)}$ (dashed line) for Example 10.5.	267
10.8	Overlaid plots of the standard normal and standard IG(0.8) distributions.	270
C.1	Array A where $x_1 < x_2 < \dots < x_n$ and $y_1 < y_2 < \dots < y_m$	283
C.2	Array A for determining candidate sums for the heap.	283

Abstract

In mathematics and statistics, the desire to eliminate mathematical tedium and facilitate exploration has led to computer algebra systems. These computer algebra systems allow students and researchers to perform more of their work at a conceptual level. The design of generic algorithms for tedious computations allows modelers to push current modeling boundaries outward more quickly.

Probability theory, with its many theorems and symbolic manipulations of random variables is a discipline in which automation of certain processes is highly practical, functional, and efficient. There are many existing statistical software packages, such as SPSS, SAS, and S-Plus, that have *numeric* tools for statistical applications. There is a potential for a *probability package* analogous to these statistical packages for manipulation of random variables. The software package being developed as part of this dissertation, referred to as “A Probability Programming Language” (APPL) is a random variable manipulator and is proposed to fill a technology gap that exists in probability theory.

My research involves developing algorithms for the manipulation of *discrete* random variables. By defining data structures for random variables and writing algorithms for implementing common operations, more interesting and mathematically intractable probability problems can be solved, including those not attempted in undergraduate statistics courses because they were deemed too mechanically arduous. Algorithms for calculating the probability density function of order statistics, transformations, convolutions, products, and minimums/maximums of independent discrete random variables are included in this dissertation.

ALGORITHMS FOR OPERATIONS ON
PROBABILITY DISTRIBUTIONS
IN A COMPUTER ALGEBRA SYSTEM

Chapter 1

Introduction

Since the beginning of the human race, man has striven to overcome obstacles and simplify complexities that have faced him in all walks of life. Attempts to solve difficult problems have produced new inventions throughout history. These inventions have themselves lead to new discoveries and opened up new paths of learning. In mathematics and statistics, the desire to eliminate mathematical tedium and facilitate exploration has lead to computer algebra systems, such as Maple and Mathematica. These computer algebra systems allow students and researchers to perform more of their work at a conceptual level. The design of generic algorithms for tedious computations allows modelers to push current modeling boundaries outward more quickly. Problems once labeled as “intractable” can now be solved.

Upon understanding a certain problem-solving technique with a step-by-step solution process, it is natural to want to automate the process so as not to replicate the same steps when returning to the same or similar problems. This is true for concepts in many, if not all, disciplines of study. Probability theory, with its many theorems (e.g., the sum of independent normal random variables is normally distributed) and symbolic manipulations of random variables (e.g. the product of two random variables), is a discipline in which automation of certain processes is highly practical,

functional, and efficient. The only effort to automate probability manipulations and calculations that I have found to date is the Mathematica-based *mathStatica* due to Rose and Smith (2001). This is surprising when one considers the dozens of packages written by computer algebra system users for other mathematical disciplines, such as abstract algebra, chaos theory, combinatorics, operations research, and real analysis, just to name a few. (For examples of packages available in Maple, see <http://www.mapleapps.com>.)

There are many existing statistical software packages, such as SPSS, SAS, and S-Plus, that have *numeric* tools for statistical applications. In fact, most computer algebra systems, such as Maple and Mathematica, contain built-in statistical libraries with symbolic capabilities for use in statistical computations. Applied statistical calculations (e.g., calculating the sample mean) are usually numeric manipulations of *data* based on known formulas. According to the help menu for Maple Version 6, its statistics package provides various descriptive statistical functions for the analysis of statistical data (e.g., mean, median, standard deviation), the capability to create various statistical plots (e.g., histogram, scatter plot, box plot), and various tools for transforming lists of statistical data (e.g., sorting data, computing moving averages). Also available are subpackages that provide

- random variate generation for certain distributions, such as the standard normal, gamma, and beta distributions,
- numerical evaluation of certain statistical distributions [e.g., calculate $\Pr(X \leq 4.0)$ for a standard normal random variable X],
- one-way analysis of variance, and
- a tool for fitting curves to statistical data.

The procedures in the Maple **stats** package and its subpackages perform numeric

computations and provide plots associated with *data sets*, as indicated in their descriptions. Although this is a valuable feature of the Maple software, these procedures do not define random variables or perform operations on them. For example, although the Maple statistical procedure `skewness` can compute the skewness of the data set $\{1, 2, 3, 3, 6, 7\}$, it cannot determine the skewness of a normal random variable with mean $\mu = 2$ and standard deviation $\sigma = 4$. Since the Maple statistical procedures cannot be applied to probability distribution functions, solving probability problems with these procedures is impossible.

Further, Karian and Tanis (1999, preface) have developed procedures in Maple to serve as a supplement for “statistical analysis and also explorations within a rich mathematical environment.” Karian and Tanis’s statistics supplement to Maple “consists of about 130 procedures written specifically to promote explorations of probabilistic and statistical concepts.” Their supplement includes procedures for calculating descriptive statistics (e.g., `Mean`, `Median`, and `Variance`), generating random samples from distributions, plotting (e.g., `BoxWhisker`, `PlotEmpPDF`, and `StemLeaf`), working with regression and correlation problems, producing the probability density function (PDF) and cumulative distribution function (CDF) of some distributions, finding percentiles of some distributions, producing confidence intervals, performing an analysis of variance, performing goodness-of-fit and nonparametric tests (e.g., `QQFit`, `ChiSquareFit`, and `KSFit`), and computing the convolution of two random variables. While Karian and Tanis have focused their efforts on building a mainly statistical package powered by Maple, there is a potential for a *probability package* analogous to this statistical package for manipulation of random variables.

The notion of probability software is different from the notion of applied statistical software. An early work by Kendall (1992) made a distinction between packages that are able to *support* investigations and those that aim to *implement structure* “to build

in elements of theory as a preliminary to research investigations.” The latter is the type of software that is not currently available, except for the forthcoming *mathStat-ica*, for processing procedures for random variables in probability theory. Although the text by Hastings (2000), *Introduction to Probability with Mathematica*, also uses Mathematica as a tool for studying probability theory, the book’s on-line description (available at <http://www.crcpress.com/us>) states that “its clever use of simulation to illustrate concepts and motivate important theorems gives it an important and unique place in the library of probability theory.” The software package being developed as part of this dissertation, referred to as “A Probability Programming Language” (APPL), does much more than motivate theorems through simulation. It is a random variable manipulator and is proposed to fill a technology gap that exists in probability theory. Although APPL will more than likely have some similarities with the forthcoming *mathStat-ica* software, its approach to discrete and continuous random variables is unique in data structure, design, and applications. From a preview of Rose and Smith’s materials at the Joint Statistical Meetings in August 2000, the multivariate distribution abilities of their software, which are not currently a part of APPL, were impressive. But the capabilities that APPL possesses (many of which are new and will be presented in this dissertation) and the simplicity in its use and data structure are quite distinct from what they have developed. APPL’s overall ability matches or surpasses much of what has currently been presented as parts of *mathStat-ica*.

The APPL software was begun several years ago by my advisor, Dr. Larry Leemis, and a former William & Mary Ph.D. student and current Army Lieutenant Colonel, Dr. Andrew Glen. Dr. Glen’s dissertation focused on writing algorithms in a computer algebra system for manipulating continuous random variables. My research involves developing similar algorithms, but for *discrete* random variables. Also, be-

fore any algorithms for discrete random variables could be developed for APPL, a data structure that complimented the data structure for continuous random variables was created.

As can be attested to by Parlar's book, *Interactive Operations Research with Maple: Methods and Models* (Parlar, 2000), Vivaldi's discrete mathematics text, *Experimental Mathematics with Maple* (Vivaldi, 2001), Lopez's book, *Advanced Engineering Mathematics* (Lopez, 2001), and Karian and Tanis's 2nd edition of *Probability and Statistics: Explorations with Maple* (Karian & Tanis, 1999), other researchers across the country have incorporated computer algebra systems into mathematical fields, especially those with statistical, probabilistic, and combinatorial applications. By taking advantage of computer algebra systems, software that will derive functions, as opposed to computing numbers, can be developed. Computer algebra systems can be exploited to eliminate repetitive and tedious operations (e.g., calculating moments or finding the distribution of order statistics) associated with random variables. By defining data structures for random variables and writing algorithms for implementing common operations, more interesting and mathematically intractable probability problems can be solved, including those not attempted in undergraduate statistics courses because they were deemed too mechanically arduous. Instructors, students, and researchers can take the time they save in mathematical manipulation and apply it to problem formulation and analysis.

This dissertation contains descriptions of some of the procedures comprising the core of APPL. The APPL tree diagram in Figure 1.1 summarizes the existing procedures in APPL. My specific contributions to APPL include

- devising a data structure for representing the distributions of univariate discrete random variables. The data structure accommodates distributions defined numerically, e.g., $f(x) = 1/4$ for $x = 1$ and $f(x) = 3/4$ for $x = 2$, and formulaically,

e.g., $f(x) = x/6$ for $x = 1, 2, 3$;

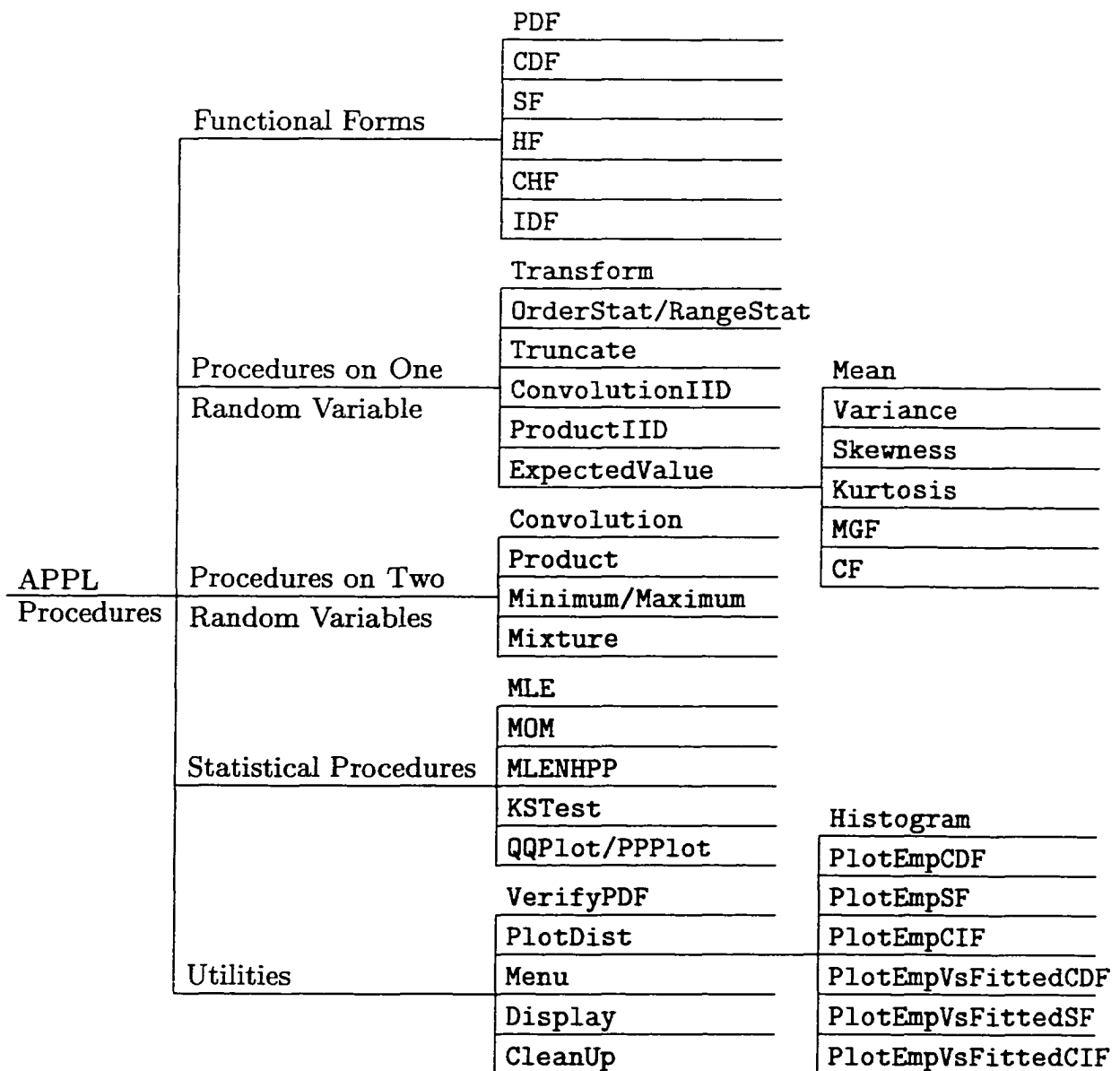


Figure 1.1: APPL tree procedures diagram.

- converting any functional representation of a discrete random variable into any other functional representation using the devised data structure, i.e., allowing conversion amongst the PDF, CDF, survivor function (SF), hazard function (HF), cumulative hazard function (CHF), and inverse distribution function

(IDF);

- providing straightforward instantiation of well-known discrete distributions, such as the binomial, Poisson, or geometric, with either numeric or symbolic parameters;
- providing straightforward instantiation of non-standard discrete distributions, such as the “matching birthday” or “bingo cover” distribution;
- handling discrete distributions of all types, including those that may have never been previously created or explored;
- calculating summary characteristics for discrete random variables, such as the mean, variance, or moment generating function (mgf);
- plotting any of the six functional forms of a discrete distribution with fixed parameters [e.g., the PDF of a binomial(6, 0.4) random variable or the CDF of a Zipf(5) random variable];
- developing algorithms that calculate the PDF of
 - * the r th order statistic from a sample of n independent and identically distributed (iid) discrete random variables, where sampling can occur either with or without replacement;
 - * the sum of independent discrete random variables, i.e., $Z = X + Y$;
 - * the product of independent discrete random variables, i.e., $Z = XY$;
 - * a transformation of a discrete random variable, $Y = g(X)$;
 - * the minimum and maximum of independent discrete random variables, i.e., $Z = \min \{X, Y\}$ and $Z = \max \{X, Y\}$;

- providing maximum likelihood estimation (MLE) for complete and right-censored data for continuous and discrete distributions defined on a single interval of support;
- providing method of moments (MOM) estimation for discrete and continuous distributions defined on a single interval of support;
- providing maximum likelihood estimation for non-homogeneous Poisson processes (NHPP);
- verifying a continuous random variable X has a legitimate PDF in the sense that $f(x) \geq 0$ for all x and $\int_{-\infty}^{\infty} f(x) dx = 1$;
- calculating the PDF of the range of a random sample of size n drawn from a continuous population;
- calculating the PDF of the mixture of independent continuous random variables;
- calculating the PDF of a truncated continuous random variable;
- verifying whether a continuous distribution satisfies Benford's law;
- providing goodness-of-fit testing by calculating the Kolmogorov–Smirnov test statistic;
- providing plots for testing model adequacy, such as an empirical versus fitted CDF plot, $Q-Q$ plots, $P-P$ plots, and empirical versus fitted cumulative intensity function plots for data that can be approximated by the power-law process;
- providing utilities for simplifying functional forms of distributions. For example, the utility procedure `CleanUp` puts a random variable in its simplest form before returning it to the user. If $X \sim \text{Normal}(0, 1)$ and $Y \sim \text{Uniform}(0, 1)$, for

example, then the APPL procedure `Product` returns the PDF of $V = XY$ as

$$f_V(v) = \begin{cases} \frac{\sqrt{2}Ei(1,1/2v^2)}{4\sqrt{\pi}} & -\infty < v < 0 \\ \frac{\sqrt{2}Ei(1,1/2v^2)}{4\sqrt{\pi}} & 0 \leq v < \infty, \end{cases}$$

where Ei is an exponential integral defined for $Re(x) > 0$ by

$$Ei(n, x) = \int_1^\infty \frac{e^{-xt}}{t^n} dt,$$

where n is a non-negative integer. The procedure `CleanUp` puts the identical pieces of the PDF of V together and returns it as

$$f_V(v) = \frac{\sqrt{2}Ei(1, 1/2v^2)}{4\sqrt{\pi}} \quad -\infty < v < \infty;$$

- deriving the distribution of the Kolmogorov–Smirnov test statistic for sampling from an exponential population with the parameter estimated from data for $n = 1, 2$;
- supplementing the structured programming language that hosts the software, in this case Maple, so that all of the above bullets may be used in mathematical and computer programming in Maple.

In addition, I have updated APPL to be compatible with newer versions of Maple, the latest being Version 6.

The following chapters highlight my specific contributions to APPL. Chapter 2 outlines the data structure, functional forms, and core procedures for discrete distributions. Chapter 3 presents algorithms for computing the PDF of order statistics drawn from discrete parent populations, along with an implementation of the algorithms in a computer algebra system. Some examples illustrate the utility of these algorithms.

Chapter 4 introduces algorithms for computing the PDF of the convolution and product of the PDFs of two independent discrete random variables. Chapter 5 introduces algorithms for determining the distribution of the transformation of a discrete random variable. Chapter 6 presents algorithms for determining the PDF of the minimum and maximum of random variables. Some of the algorithms in these chapters involved implementing known results, while others involved the development of original algorithms. Chapter 7 overviews several APPL procedures concerning continuous random variables that have either been extended or newly constructed as separate research areas of my dissertation. Chapter 8 considers an application in APPL that identifies certain survival distributions that satisfy Benford's law. Chapter 9 overviews procedures written in APPL specifically to perform input modeling. Chapter 10 illustrates additional applications of the procedures that have been developed in APPL. Chapter 11 contains suggestions for future work.

1.1 Notation and Nomenclature

The following is a list of comments about the notation, names, abbreviations, and APPL syntax that will be used throughout this dissertation:

- the abbreviations “PDF,” “CDF,” “SF,” “HF,” “CHF,” and “IDF” represent probability density function $f(x)$, cumulative distribution function $F(x)$, survivor function $S(x)$, hazard function $h(x)$, cumulative hazard function $H(x)$, and inverse distribution function $F^{-1}(x)$, respectively;
- the abbreviation “iid” denotes independent and identically distributed;
- parentheses on subscripts denote order statistics, e.g., the r th order statistic associated with a random sample X_1, X_2, \dots, X_n is denoted by $X_{(r)}$;

- “Pr” abbreviates probability. When “Pr” is used in an expression such as $\text{Pr}(X = x)$, it is read as “probability that X is equal to x ;
- “MLE” and “MOM” abbreviate maximum likelihood estimation and method of moments estimation, respectively;
- “NHPP” abbreviates non-homogeneous Poisson process;
- typewriter font is used for APPL statements. The Maple input prompt “>” is included in the examples;
- in an APPL procedure, the use of square brackets around an argument indicates that the argument is optional. For example, `PlotDist(X, [low], [high])` is an APPL procedure that plots the distribution of X from the value `low` to the value `high`. If these two arguments are not included in the procedure call, Maple automatically determines the plot range;
- “MUG” refers to the Maple Users’ Group, which is an on-line Maple newsgroup that provides suggestions and help for Maple related issues;
- “log” is the natural logarithm (log base e);
- in Maple plots, \ominus represents a filled (or solid) circle;
- for clarity, all sentence punctuation has been omitted from APPL statements;
- the pronoun “we” refers to those who have developed APPL.

1.2 Introductory Examples

I close the introduction with three examples that display three different APPL procedures (`OrderStat`, `ConvolutionIID`, and `MOM`) presented in Figure 1.1 and discussed in this dissertation.

Example 1.1. (Hogg & Craig, 1995, page 230) A fair die is cast eight times. Find the PDF of the smallest of the eight numbers obtained, $X_{(1)}$.

Solution: To compute the numeric PDF by hand, we calculate the value

$$f_{X_{(1)}}(x) = \sum_{w=0}^7 \binom{8}{w} \left(\frac{1}{6}\right)^{8-w} \left(1 - \frac{x}{6}\right)^w$$

for $x = 1, 2, \dots, 6$. (Maple incorrectly calculates 0^0 as 1. While mathematically incorrect, it allows the proper calculation.) To determine the probability that the first order statistic assumes the value $x = 4$, for example, we calculate

$$\begin{aligned} f_{X_{(1)}}(4) &= \sum_{w=0}^7 \binom{8}{w} \left(\frac{1}{6}\right)^{8-w} \left(\frac{1}{3}\right)^w \\ &= \frac{1}{1679616} + \frac{1}{104976} + \frac{7}{104976} + \frac{7}{26244} + \frac{35}{52488} + \frac{7}{6561} + \frac{7}{6561} + \frac{4}{6561} \\ &= \frac{6305}{1679616} \\ &\cong 0.0038. \end{aligned}$$

Similar calculations for $x = 1, 2, \dots, 6$ yield the PDF of the first order statistic as

$$f_{X_{(1)}}(x) = \begin{cases} \frac{1288991}{1679616} & x = 1 \\ \frac{36121}{186624} & x = 2 \\ \frac{58975}{1679616} & x = 3 \\ \frac{6305}{1679616} & x = 4 \\ \frac{85}{559872} & x = 5 \\ \frac{1}{1679616} & x = 6. \end{cases}$$

A uniform discrete random variable X with minimum support a and maximum support b is a pre-defined random variable in APPL. Thus, we can obtain the above PDF for the first order statistic, $X_{(1)}$, with the statements

```
> X := UniformDiscreteRV(1, 6);
```

> OrderStat(X, 8, 1);

□

In the next example, APPL is able to find the convolution of a large number (150) of discrete random variables. While not impossible, computing the actual distribution by hand is tremendously tedious and time-consuming.

Example 1.2. (Thompson, 2000, page 54) Let $S = X_1 + X_2 + \cdots + X_{150}$, where the X_i 's are independent, $\Pr(X_i = -1) = \Pr(X_i = 0) = \Pr(X_i = 1) = 1/3$, $i = 1, 2, \dots, 150$. Compute $\Pr(S = 5)$.

Solution: Since the mass values of the parent populations are adjacent, $\Pr(S = 5)$ can be computed using a combinatorics approach:

$$\Pr(S = 5) = \sum_{\substack{\{(p,q,r)\} \\ p+q+r=150, \\ 0 \leq p \leq 150, \\ 0 \leq q \leq 150, \\ 0 \leq r \leq 150, \\ -p+r=5}} \binom{150}{p, q, r} \left(\frac{1}{3}\right)^p \left(\frac{1}{3}\right)^q \left(\frac{1}{3}\right)^r$$

or equivalently

$$\Pr(S = 5) = \sum_{p=0}^{72} \binom{150}{p, 145 - 2p, 5 + p} \left(\frac{1}{3}\right)^{150},$$

yielding the result

$$\Pr(S = 5) = \frac{160709987007649212790999852367465829596098558279031212787052332840770}{4567759074507740406477787437675267212178680251724974985372646979033929},$$

which is approximately 0.03518.

The APPL statements

```
> X := [[1 / 3, 1 / 3, 1 / 3], [-1, 0, 1], ["Discrete", "PDF"]];
> S := ConvolutionIID(X, 150);
> PDF(S, 5);
```

yield the exact PDF for S . The statement `PDF(S, 5)` returns the same value computed by the combinatorics method.

The true utility of APPL is not demonstrated in this particular example because of the adjacent mass values of the parent populations. The APPL approach allows for unequal mass values and unequally spaced support values. Also, more than three mass values can be used in the APPL approach. \square

Example 1.3. (Larsen & Marx, 2001, page 258) During the latter part of the nineteenth century, Prussian officials gathered information on the hazards that horses posed to cavalry soldiers. A total of 10 cavalry corps were monitored over a period of 20 years (Bortkiewicz, 1898). The number of fatalities due to kicks, X , was recorded for each year and each corps. Table 1.1 shows the empirical distribution of X for these 200 “corps-years.”

Table 1.1: Observed horse kick fatalities.

Number of Deaths x	Observed Number of Corps-Years in Which x Fatalities Occurred
0	109
1	65
2	22
3	3
4	1
	200

Among several other phenomena that Bortkiewicz successfully “fit” with the Poisson model, the one best remembered is the Prussian cavalry data described above. The Poisson distribution has PDF

$$f_X(x) = \frac{e^{-\lambda} \lambda^x}{x!} \quad x = 0, 1, 2, \dots; \lambda > 0.$$

Find the method of moments estimate for the parameter λ .

Solution: The first APPL statement defines X as a Poisson random variable with

mean λ . The list `HorseKickFatalities` is a pre-defined list in APPL containing the horse kick data in Table 1.1. The statement `MOM(X, HorseKickFatalities, [lambda])` computes the method of moments estimate for the parameter λ .

```
> X := PoissonRV(lambda);
> MOM(X, HorseKickFatalities, [lambda]);
```

The resulting estimate for the parameter is $\hat{\lambda} = \frac{61}{100}$, which is the method of moments estimator 0.61 fatalities per corps-year. Figure 1.2 displays a plot of the actual and estimated PDF for the `HorseKickFatalities` data. \square

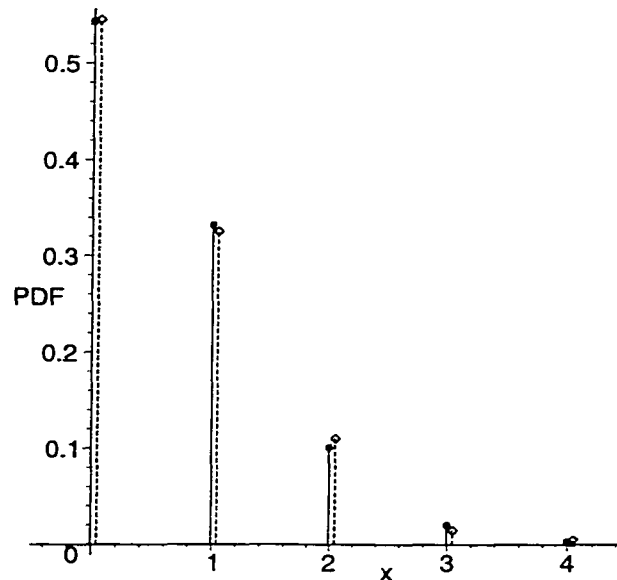


Figure 1.2: Actual and estimated PDF for the `HorseKickFatalities` data. The solid lines represent the PDF values of the `Poisson(61/100)` random variable at $x = 0, 1, \dots, 4$. The dashed lines represent the actual PDF values of the `HorseKickFatalities` data at $x = 0, 1, \dots, 4$.

Chapter 2

Data Structure

APPL was originally written for continuous random variables and algorithmic procedures that applied to them. Before adding discrete random variable capabilities, a data structure that paralleled the continuous case needed to be constructed for discrete distributions. The data structure for a continuous random variable with PDF $f(x)$ is a Maple list consisting of three sublists with the following general format:

$$[[f(x)], [support], ["Continuous", "XXX"]],$$

where **XXX** is either PDF, CDF, SF, HF, CHF, or IDF. The acronyms represent the following for a random variable X , where extensions for discrete random variables have been included:

- probability density function (PDF). For discrete random variables, the probability mass function $f(x) = \Pr(X = x)$ will also be referred to as a probability density function;
- cumulative distribution function (CDF) $F(x) = \int_{-\infty}^x f(w) dw$ for a continuous random variable or $F(x) = \sum_{w \leq x} f(w)$ for a discrete random variable;

- survivor function (SF) $S(x) = \int_x^\infty f(w) dw$ for a continuous random variable or $S(x) = \sum_{w \geq x} f(w)$ for a discrete random variable;
- hazard function (HF) $h(x) = \frac{f(x)}{S(x)}$ for a continuous or discrete random variable;
- cumulative hazard function (CHF) $H(x) = \int_{-\infty}^x h(w) dw$ for a continuous random variable or $H(x) = -\log(S(x))$ for a discrete random variable; and
- inverse distribution function (IDF) $F^{-1}(x)$ for a continuous or discrete random variable.

The common data structure used in this software is referred to as the “list-of-sublists.” All APPL random variables are input in a list that contains three sublists, each with a specific purpose. The first sublist contains either a formula or a numeric list that defines the functional representation of the distribution. For example, the PDF representation of the Poisson distribution with mean λ and support $x = 0, 1, 2, \dots$ has as its first sublist

$$\left[x \rightarrow \frac{\lambda^x e^{-\lambda}}{x!} \right].$$

The CDF representation of the geometric($\frac{1}{4}$) distribution with support $x = 1, 2, \dots$, has as its first sublist

$$\left[x \rightarrow 1 - \left(\frac{3}{4} \right)^x \right].$$

The SF representation (in a numeric Maple list) of the probability of obtaining an x or higher (where $x = 1, 2, \dots, 6$) on single roll of a fair 6-sided die is

$$\left[1, \frac{5}{6}, \frac{2}{3}, \frac{1}{2}, \frac{1}{3}, \frac{1}{6} \right].$$

Since the third sublist is less complicated than the second, the third sublist will be examined next. The third sublist indicates the distribution form of the function in the first sublist. The first element of the third sublist is either the Maple string "Continuous" for a continuous random variable or "Discrete" for a discrete random variable. The second element of the third sublist indicates which of the six functional representations is used in the first sublist. Again, the choices for this second element are PDF, CDF, SF, HF, CHF, or IDF. For the Poisson, geometric, and uniform discrete distributions described in the previous paragraphs, their third sublists are ["Discrete", "PDF"], ["Discrete", "CDF"], and ["Discrete", "SF"], respectively.

The second sublist contains the random variable's support. For a continuous random variable, this second sublist contains an ordered list of real numbers that delineate the end points of the intervals for the functions in the first sublist. The end point of each interval is automatically the start point of the subsequent interval. The triangular(1, 2, 3) CDF, for example, is defined by a piecewise function in the first sublist, specifically $[x \rightarrow \frac{1}{2}x^2 - x + \frac{1}{2}, x \rightarrow -\frac{1}{2}x^2 + 3x - \frac{7}{2}]$. Its second sublist, [1, 2, 3], defines the support interval for each piece of the function. Thus the CDF is

$$F(x) = \begin{cases} 0 & x \leq 1 \\ \frac{1}{2}x^2 - x + \frac{1}{2} & 1 < x \leq 2 \\ -\frac{1}{2}x^2 + 3x - \frac{7}{2} & 2 < x \leq 3 \\ 1 & x > 3. \end{cases}$$

Putting the three sublists together, the following APPL statement defines a triangular(1, 2, 3) random variable X as a list-of-sublists:

```
> X := [[x -> x ^ 2 / 2 - x + 1 / 2, x -> -x ^ 2 / 2 + 3 * x - 7 / 2],
        [1, 2, 3], ["Continuous", "CDF"]]
```


Standard continuous and discrete distributions, such as the normal, binomial, and Poisson distributions, are pre-defined in APPL.

A discrete random variable's support can be input in one of several different formats. This variation in formats presents greater difficulty than in the continuous case for determining a structure for the second sublist. For example, the Poisson distribution with PDF $f(x) = \frac{\lambda^x e^{-\lambda}}{x!}$ for $x = 0, 1, 2, \dots$ has as its support the set of nonnegative integers. This support has a pattern to it; the first value of x at which the PDF is defined is zero and the rest of the support consists of subsequent integers. Many discrete random variables do not have a patterned support. When designing the second sublist structure for discrete random variables, we first had to distinguish between random variables that had some type of pattern to their support versus those that did not. For example, let X be a binomial random variable with parameters $n = 5$ and $p = 0.2$, and PDF

$$f(x) = \binom{5}{x} (0.2)^x (0.8)^{5-x} \quad x = 0, 1, \dots, 5.$$

Let Y be the random variable with PDF

$$f(y) = \begin{cases} 0.2 & y = 1 \\ 0.5 & y = \frac{7}{2} \\ 0.3 & y = 11. \end{cases}$$

The support of the random variable X consists of adjacent integers. For random variables whose support Ω is incremented by one, only the first and last values of the support are needed to generate the entire support list. This support case is called the *Dot* case, since we can write it in Maple's range (also called type '..') format: $\min\{\Omega\} .. \max\{\Omega\}$. Thus, the second sublist for the random variable X is input as $[0 .. 5]$.

Random variables with supports that display no pattern, such as the support of Y , must be entered as a Maple list, where the list values are separated by commas. This support case is called the *NoDot* case. The support values listed in sublist two correspond to the distribution's function values in the first sublist. As an example, we would write the first and second sublists of Y as $[0.2, 0.5, 0.3]$ and $[1, 7 / 2, 11]$, respectively.

After distinguishing between the *Dot* and *NoDot* support cases, there are subcases of these general two cases to consider. First, in the *NoDot* case, the function in the first sublist can be written as a formula or a list of numeric elements. The random variable Y with PDF values $[0.2, 0.5, 0.3]$ is a Maple list of numeric elements (separated by commas). On the other hand, the random variable X with PDF

$$f(x) = \frac{x}{27} \quad x = 1, 3, 7, 16,$$

is a valid discrete probability mass function whose PDF can be written as a formula in Maple's function notation as $x \rightarrow x/27$. Its first two sublists are input in Maple as $[x \rightarrow x / 27]$ and $[1, 3, 7, 16]$. APPL allows the user to enter a discrete random variable represented in the *NoDot* case in either format, numeric or formulaic. Converting sublists one and two to a "standard" *NoDot* format (where the first sublist is *not* written as a formula) is handled in a procedure called `Convert` (see Section 2.1.1). If we enter X as displayed above and apply the `Convert` procedure to X with the APPL statements

```
> X := [[x -> x / 27], [1, 3, 7, 16], ["Discrete", "PDF"]];
> X := Convert(X);
```

X is returned as $[[\frac{1}{27}, \frac{1}{9}, \frac{7}{27}, \frac{16}{27}], [1, 3, 7, 16], ["Discrete", "PDF"]]$. The "standard" *NoDot* and *Dot* formats, along with the `Convert` procedure, are discussed in more detail in Section 2.1.

The *Dot* case also has two general subcases—either the random variable has finite or infinite support. Let Ω represent the support of a random variable X . The “standard” *Dot* format has the general form:

[$\min\{\Omega\} \dots \max\{\Omega\}$, Support incremented by k , Support transformed by $g(x)$],

where the default value of k (if not entered) is 1 and the default value of $g(x)$ (if not entered) is the identity function. If X has infinite support, it is understood that $\max\{\Omega\} = \infty$.

To introduce the different variations in format, let $X_1 \sim \text{geometric}(1/4)$ with PDF

$$f_{X_1}(x) = \frac{1}{4} \left(\frac{3}{4} \right)^{x-1} \quad x = 1, 2, \dots$$

Let $X_2 = 2Y$, where $Y \sim \text{binomial}(5, 0.2)$. The PDF of X_2 is

$$f_{X_2}(x) = \binom{5}{x/2} (0.2)^{x/2} (0.8)^{5-x/2} \quad x = 0, 2, 4, \dots, 10.$$

Let X_3 be a discrete random variable with PDF

$$f_{X_3}(x) = \frac{x+6}{54} \quad x = 1, 4, 9, 16,$$

and let X_4 be a discrete random variable with PDF

$$f_{X_4}(x) = \frac{\sqrt{x}}{15} \quad x = 9, 25, 49.$$

The support of each of these random variables has a pattern that can be accounted for in the *Dot* data structure. X_1 's support is incremented by ones, while X_2 's support is incremented by twos. The support of X_3 is not only incremented by ones, but also is transformed by the function $g(x) = x^2$. Going a step further,

X_4 's support is incremented by twos starting with $x = 3$ and then transformed by $g(x) = x^2$. In each of these cases, the minimum and maximum values of the support, the increment of the support, and the type of transformation applied to the support is pertinent information. Thus, in the *Dot* case, the second sublist will consist of either 1, 2, or 3 elements separated by commas containing this information. This general capability has been included in the data structure to accommodate discrete algorithmic procedures, such as `Transform`.

For the random variables X_1 , X_2 , X_3 , and X_4 , their supports are input in the second sublist in the APPL list-of-sublists as

$$\begin{aligned} X_1 &: [1 \dots \infty] \\ X_2 &: [0 \dots 10, 2] \\ X_3 &: [1 \dots 4, x \rightarrow x^2] \\ X_4 &: [3 \dots 7, 2, x \rightarrow x^2]. \end{aligned}$$

In summary, if the random variable X is discrete, its support will match one of the cases displayed in Table 2.1. An example of the APPL list-of-sublists format of a random variable from each individual category follows.

1. *NoDot* SUPPORT FORMAT: The random variable's support Ω is a Maple numeric list of elements.

- (a) NUMERIC PDF: The random variable's PDF is a Maple numeric list of elements.

Example 2.1. Let X be the random variable with PDF

$$f(x) = \begin{cases} 0.6 & x = 2.5 \\ 0.3 & x = 3 \\ 0.1 & x = 15. \end{cases}$$

Table 2.1: Discrete random variable support categories.

Support Ω	Cases	Subcases		Examples
<i>NoDot</i> $\Omega = [x_1, x_2, \dots, x_n]$	Numeric PDF	—		2.1
	Formulaic PDF			2.2
<i>Dot</i> $\Omega = [\min\{\Omega\} .. \max\{\Omega\},$ Incremented by $k,$ Transformed by $g(x)]$	Finite Support	$k = 1$	$g(x) = x$	2.3
			$g(x) \neq x$	2.4
		$k \neq 1$	$g(x) = x$	2.5
			$g(x) \neq x$	2.6
	Infinite Support	$k = 1$	$g(x) = x$	2.7
			$g(x) \neq x$	2.8
		$k \neq 1$	$g(x) = x$	2.9
			$g(x) \neq x$	2.10

This random variable X is input in APPL as

```
> X := [[0.6, 0.3, 0.1], [2.5, 3, 15], ["Discrete", "PDF"]];
```

(b) FORMULAIC PDF: The random variable's PDF is formulaic.

Example 2.2. Let X be the random variable with PDF $f(x) = x/8$ for $x = 1, 3, 4$. This random variable X is input in APPL as

```
> X := [[x -> x / 8], [1, 3, 4], ["Discrete", "PDF"]];
```

2. *Dot* SUPPORT FORMAT: The random variable's support Ω is of Maple type '..', i.e., $\min\{\Omega\} .. \max\{\Omega\}$.

(a) FINITE SUPPORT: The random variable's support is finite.

- i. The random variable's support is incremented by $k = 1$ and transformed by the identity function, i.e., $g(x) = x$ for all x .

Example 2.3. Let X be a Benford random variable with PDF $f(x) = \log_{10}(1 + 1/x)$ for $x = 1, 2, \dots, 9$. The Benford random variable is input in APPL as

```
> X := [[x -> log[10](1 + 1 / x)], [1 .. 9],
        ["Discrete", "PDF"]];
```

- ii. The random variable's support is incremented by $k = 1$ and transformed by a function other than the identity function, i.e., $g(x) \neq x$ for some or all x .

Example 2.4. Let X be a random variable with PDF $f(x) = x/216$ for $x = 27, 64, 125$. This random variable is input in APPL as

```
> X := [[x -> x / 216], [3 .. 5, x -> x ^ 3],
        ["Discrete", "PDF"]];
```

- iii. The random variable's support is incremented by $k \neq 1$ and transformed by $g(x) = x$.

Example 2.5. Let $Y \sim \text{binomial}(5, p)$, and let $X = 2Y$. The random variable X is input in APPL as

```
> X := [[x -> 120 * p ^ (x / 2) * (1 - p) ^ (5 - x / 2) /
        ((5 - x / 2)! * (x / 2)!)],
        [0 .. 10, 2], ["Discrete", "PDF"]];
```

- iv. The random variable's support is incremented by $k \neq 1$ and transformed by $g(x) \neq x$.

Example 2.6. Let X be a random variable with PDF $f(x) = 1600/489x$ for $x = 4, 25, 64$. The random variable is input in APPL as

```
> X := [[x -> (1600 / 489) / x], [2 .. 8, 3, x -> x ^ 2],
        ["Discrete", "PDF"]];
```

(b) INFINITE SUPPORT: The random variable's support is infinite.

- i. The random variable's support is incremented by $k = 1$ and transformed by $g(x) = x$.

Example 2.7. Let X be a negative binomial random variable with parameters $r = 3$ and $p = 1/3$. The simplified PDF of X is

$$f(x) = \frac{(x-2)(x-1)}{16} \left(\frac{2}{3}\right)^x \quad x = 3, 4, \dots$$

This random variable is input in APPL as

```
> X := [[x -> (((x - 2) * (x - 1)) / 16) * (2 / 3) ^ x],
        [3 .. infinity], ["Discrete", "PDF"]];
```

- ii. The random variable's support is incremented by $k = 1$ and transformed by $g(x) \neq x$.

Example 2.8. Let X be the square of a $\text{geometric}(\frac{1}{2})$ random variable; i.e., $f(x) = \frac{1}{2\sqrt{x}}$ for $x = 1, 4, 9, \dots$. This random variable is input in APPL as

```
> X := [[x -> 2 ^ (-sqrt(x))], [1 .. infinity, x -> x ^ 2],
        ["Discrete", "PDF"]];
```

- iii. The random variable's support is incremented by $k \neq 1$ and transformed by $g(x) = x$.

Example 2.9. Let $Y \sim \text{geometric}(\frac{1}{2})$. Let $X = 2Y$ with PDF $f(x) = (\frac{1}{2})^{(x/2)}$ for $x = 2, 4, 6, \dots$. This random variable is input in APPL as

```
> X := [[x -> (1 / 2) ^ (x / 2)], [2 .. infinity, 2],
        ["Discrete", "PDF"]];
```

- iv. The random variable's support is incremented by $k \neq 1$ and transformed by $g(x) \neq x$.

Example 2.10 Let $Z \sim \text{geometric}(\frac{1}{2})$, $Y = 2Z$, and $X = Y^2$. The PDF of X is $f(x) = (\frac{1}{2})^{(\sqrt{x}/2)}$ for $x = 4, 16, 36, \dots$. This random

variable is input in APPL as

```
> X := [[x -> (1 / 2) ^ (sqrt(x) / 2)],
        [2 .. infinity, 2, x -> x ^ 2],
        ["Discrete", "PDF"]];
```

2.1 Standard Discrete Data Structure Formats

Before an operation is performed on a discrete random variable inside an APPL procedure, the random variable is first converted to its standard format inside that procedure. As discussed in the preceding section, there is a standard discrete data structure format for both the *NoDot* and *Dot* cases. The conversion is necessary since every APPL procedure expects to receive and operate on random variables in these standard formats.

Let X , for example, be the random variable discussed earlier in this chapter with PDF

$$f(x) = \frac{x}{27} \quad x = 1, 3, 7, 16.$$

Suppose the random variable X is input in APPL as

```
> X := [[x -> x / 27], [1, 3, 7, 16], ["Discrete", "PDF"]];
```

When an APPL procedure, such as *Mean*, receives the random variable X as an argument, it first determines if X has a *Dot* or *NoDot* support format. Since the support $\Omega = [1, 3, 7, 16]$ has a *NoDot* support format, then the procedure's formula for computing the mean expects that it is receiving the random variable X in its converted format, which is

$$\left[\left[\frac{1}{27}, \frac{1}{9}, \frac{7}{27}, \frac{16}{27} \right], [1, 3, 7, 16], ["Discrete", "PDF"] \right].$$

In this format, the mean of a random variable with mass values x_1, x_2, \dots with a

NoDot support format is computed as

$$\begin{aligned}
 E[X] &= \sum_{i=1}^{|\Omega|} x_i \cdot f(x_i) \\
 &= \sum_{i=1}^4 X[2][i] \cdot X[1][i] \\
 &= 1 \cdot \frac{1}{27} + 3 \cdot \frac{1}{9} + 7 \cdot \frac{7}{27} + 16 \cdot \frac{16}{27} \\
 &= \frac{35}{3}.
 \end{aligned}$$

where $X[2][i]$ is the i th element in sublist two (X 's support) and $X[1][i]$ is the i th element in sublist one (X 's probability values).

Again, each APPL procedure operates on random variables only in the standard *NoDot* and *Dot* formats. Expecting arguments in a pre-defined format allows algorithms to be developed that exploit these formats. Without these standard formats, an APPL procedure would be forced to diagnose the exact form of each random variable it was operating on before making any computations. The computations required in a procedure would then depend on the unique structure of each random variable, and as indicated in Table 2.1, there are ten acceptable formats for a discrete random variable in APPL. The `Convert` procedure provides each APPL procedure with a standard structural format for a discrete random variable.

Some random variables with finite support, such as the random variable Y with PDF

$$f(y) = \frac{y}{15} \quad y = 3, 5, 7,$$

can be input in APPL in several different formats, which include both a *NoDot* and *Dot* format. The APPL statements

```

> Y := [[1 / 5, 1 / 3, 7 / 15], [3, 5, 7], ["Discrete", "PDF"]];
> Y := [[y -> y / 15], [3, 5, 7], ["Discrete", "PDF"]];

```

```
> Y := [[y -> y / 15], [3 .. 7, 2], ["Discrete", "PDF"]];
```

define the same random variable Y in APPL. The first two lines define Y in its *NoDot* formats, while the third line displays Y in one possible *Dot* format. Although a different format of a certain algorithm or formula may be applied to Y in any given procedure, the outcome will be the same. Conversely, a discrete random variable with countably infinite support can only be input using a *Dot* support format. Since it is impossible to physically list each and every element of a countably infinite set, the *NoDot* support format for this type of random variable cannot be used.

2.1.1 Convert

The **Convert** procedure acts on discrete random variables in both the *Dot* and *NoDot* cases. It converts a discrete random variable X with a

- *Dot* support format to the standard APPL *Dot* support format. If Ω is the support of X , then the standard *Dot* support second sublist format is

$$[\min\{\Omega\} .. \max\{\Omega\}, \text{Support incremented by } k, \text{Support transformed by } g(x)].$$
- *NoDot* support format to the standard APPL *NoDot* support format with corresponding PDF. If $f(x)$ is the PDF and the support is $x = x_1, x_2, \dots, x_n$, then the standard *NoDot* support second sublist format and corresponding PDF first sublist are

$$[f(x_1), f(x_2), \dots, f(x_n)], [x_1, x_2, \dots, x_n].$$

The **Convert** procedure requires one argument, a discrete random variable X in its list-of-sublists format. The procedure does the following:

1. Converts X to its PDF representation (if not already in that representation) using the APPL PDF procedure which is described later in this chapter;

2. Checks to see if the random variable X is in the discrete *Dot* case. It does this by checking whether the first element in the second sublist is of Maple type “range.” An expression of type range (also called type ‘..’) has two operands, the left-hand side expression and the right-hand side expression. For example, a $\text{geometric}(1/2)$ random variable defined in APPL has the structure

$$[[x \rightarrow 1/2^x], [1 .. \infty], ["Discrete", "PDF"]].$$

The first element in the second sublist, $1 .. \infty$, is of type range, where the left-hand side expression is 1 and the right-hand side expression is ∞ .

3. If X with support Ω is in the *Dot* case, then its support sublist contains either one, two, or three elements. The structure of the support sublist is either:

- $[\min\{\Omega\} .. \max\{\Omega\}]$. This support relays the following information about the support of the random variable X :
 - The first value of its support is $\min\{\Omega\}$,
 - The last value of its support is $\max\{\Omega\}$,
 - The support values are incremented by $k = 1$, and
 - The transformation on the support values is $g(x) = x$.

A random variable with this support format is converted to the standard *Dot* format as

$$[\min\{\Omega\} .. \max\{\Omega\}, 1, x \rightarrow x].$$

- $[\min\{\Omega\} .. \max\{\Omega\}, g(x)]$, where $g(x)$ is some function other than the identity function, such as $g(x) = x^2$. This support relays the following information about the support of the random variable X :
 - The first value of its support is $\min\{\Omega\}$,

- The last value of its support is $\max\{\Omega\}$,
- The support values are incremented by $k = 1$, and
- The transformation on the support values is the function $g(x)$, where $g(x) \neq x$.

A random variable with this support format is converted to the standard *Dot* format as

$$[\min\{\Omega\} .. \max\{\Omega\}, 1, g(x)].$$

- $[\min\{\Omega\} .. \max\{\Omega\}, k]$, where k is a positive real number, most likely an integer. This support relays the following information about the support of the random variable X :
 - The first value of its support is $\min\{\Omega\}$,
 - The last value of its support is $\max\{\Omega\}$,
 - The support values are incremented by k , where $k \neq 1$, and
 - The transformation on the support values is $g(x) = x$.

A random variable with this support format is converted to the standard *Dot* format as

$$[\min\{\Omega\} .. \max\{\Omega\}, k, x \rightarrow x].$$

- $[\min\{\Omega\} .. \max\{\Omega\}, k, g(x)]$, where k is a positive real number, most likely an integer, and $g(x)$ is some function other than $g(x) = x$. This support relays the following information about the support of the random variable X :
 - The first value of its support is $\min\{\Omega\}$,
 - The last value of its support is $\max\{\Omega\}$,
 - The support values are incremented by k , where $k \neq 1$, and

- The transformation on the support values is $g(x)$, where $g(x) \neq x$.

A random variable with this support format is already in the standard *Dot* format.

4. If X is in the *NoDot* case, then determine if its PDF in the first sublist contains a list of elements or a formula.

- If the PDF is a Maple list of elements, then the random variable is already in the standard *NoDot* format. For example, the APPL random variable

$$[[0.5, 0.3, 0.2], [1, 14, 37], ["Discrete", "PDF"]]$$

is already in the standard *NoDot* format.

- If the PDF is a formula, which means that the element in the first sublist is of type “procedure,” the elements in the second sublist are substituted into the formula in the first sublist to obtain the probability values that correspond to the support values. For example, returning to the random variable X with PDF $f(x) = x/27$ for $x = 1, 3, 7,$ and 16 , the support values $[1, 3, 7, 16]$ are substituted into the formula $x \rightarrow x/27$ to determine the corresponding probability values $[1/27, 1/9, 7/27, 16/27]$. Symbolic (e.g., $[a, b, c]$) or infinite support values are not permitted in the *NoDot* with *formulaic PDF* case.

5. The converted discrete random variable is returned (to the procedure that is using it) in its standard *Dot* or *NoDot* format.

Two examples of the **Convert** procedure for the *NoDot* and *Dot* formats are included for further clarification.

Example 2.11. Use the APPL `Convert` procedure to convert a Zipf random variable with parameter $\alpha = 1$ to the standard *Dot* format.

Solution: The APPL statements below define X as the desired Zipf random variable and convert it to its standard *Dot* format.

```
> X := [[x -> 6 / (Pi * x) ^ 2], [1 .. infinity], ["Discrete", "PDF"]];
> Convert(X);
```

The converted Zipf random variable in the standard APPL *Dot* format is

$$\left[\left[x \rightarrow \frac{6}{(\pi x)^2} \right], [1 .. \infty, 1, x \rightarrow x], ["Discrete", "PDF"] \right].$$

□

Example 2.12. Use the APPL `Convert` procedure to convert a Benford random variable to the standard *NoDot* format.

Solution: The APPL statements below define X as a Benford random variable and convert it to its standard *NoDot* format.

```
> X := [[x -> log[10](1 + 1 / x)], [1, 2, 3, 4, 5, 6, 7, 8, 9],
        ["Discrete", "PDF"]];
> Convert(X);
```

The converted Benford random variable in the standard APPL *NoDot* format is

$$\left[\left[\frac{\ln(2)}{\ln(10)}, \frac{\ln(\frac{3}{2})}{\ln(10)}, \frac{\ln(\frac{4}{3})}{\ln(10)}, \frac{\ln(\frac{5}{4})}{\ln(10)}, \frac{\ln(\frac{6}{5})}{\ln(10)}, \frac{\ln(\frac{7}{6})}{\ln(10)}, \frac{\ln(\frac{8}{7})}{\ln(10)}, \frac{\ln(\frac{9}{8})}{\ln(10)}, \frac{\ln(\frac{10}{9})}{\ln(10)} \right], [1, 2, 3, 4, 5, 6, 7, 8, 9], ["Discrete", "PDF"] \right].$$

□

2.2 The Six Functional Representations

For this subsection and the rest of Chapter 2, let X be a discrete random variable with support $\Omega = \{x_1, x_2, \dots\}$. Where further explanation and examples are necessary, X

is a random variable with PDF $f(x) = x/6$ for $x = 1, 2, 3$. Its formulaic CDF, SF, HF, CHF, and IDF are defined in Table 2.2.

Table 2.2: The six functional representations of the random variable X with PDF $f(x) = x/6$ for $x = 1, 2, 3$. The IDF is defined for $x = 1/6, 1/2$, and 1.

PDF	CDF	SF
$f(x) = \frac{x}{6}$	$F(x) = \frac{x^2 + x}{12}$	$S(x) = \frac{-x^2 + x + 12}{12}$
$h(x) = \frac{2x}{-x^2 + x + 12}$	$H(x) = -\log\left(\frac{-x^2 + x + 12}{12}\right)$	$F^{-1}(x) = \left[-\frac{1}{2} + \frac{1}{2}\sqrt{1 + 48x}\right]$
HF	CHF	IDF

The matrix in Table 2.3 shows how the PDF, CDF, SF, HF, and CHF distribution representations (given in the columns) can be found if one of the representations (given by the rows) is known. For example, if the CDF of a distribution is given, then its CHF can be determined for $x_i \in \Omega$ by

$$H(x_i) = -\log(S(x_i)) = -\log(1 - F(x_{i-1})),$$

where \log is the natural logarithm (\log base e) and $F(x_0) \equiv 0$. Some facts used to compute the entries in the matrix for $x_i \in \Omega$ are (Leemis, 1995, pages 56–57, 73):

- $h(x_i) = \frac{f(x_i)}{S(x_i)}$; (Definition)
- $H(x_i) = -\log S(x_i)$; (Definition)
- $S(x_i) = \prod_{j|x_j < x_i} (1 - h(x_j))$; and
- $H(x_j) = -\sum_{j|x_j < x_i} \log(1 - h(x_j))$.

Table 2.3: A 5×5 transition matrix for determining $f(x_i)$, $F(x_i)$, $S(x_i)$, $h(x_i)$, or $H(x_i)$ from any of the others for discrete distributions, where $F(x_0) \equiv 0$, $h(x_0) \equiv 0$, and when $|\Omega|$ is finite, $S(x_{|\Omega|+1}) \equiv 0$ and $H(x_{|\Omega|+1}) \equiv 0$.

	$f(x_i)$	$F(x_i)$	$S(x_i)$	$h(x_i)$	$H(x_i)$
$f(x)$.	$\sum_{j x_j \leq x_i} f(x_j)$	$\sum_{j x_j \geq x_i} f(x_j)$	$\frac{f(x_i)}{\sum_{j x_j \geq x_i} f(x_j)}$	$-\log \left(\sum_{j x_j \geq x_i} f(x_j) \right)$
$F(x)$	$F(x_i) - F(x_{i-1})$.	$1 - F(x_{i-1})$	$\frac{F(x_i) - F(x_{i-1})}{1 - F(x_{i-1})}$	$-\log(1 - F(x_{i-1}))$
$S(x)$	$S(x_i) - S(x_{i+1})$	$1 - S(x_{i+1})$.	$\frac{S(x_i) - S(x_{i+1})}{S(x_i)}$	$-\log S(x_i)$
$h(x)$	$h(x_i) \prod_{j x_j < x_i} [1 - h(x_j)]$	$1 - \prod_{j x_j \leq x_i} [1 - h(x_j)]$	$\prod_{j x_j < x_i} [1 - h(x_j)]$.	$-\sum_{j x_j < x_i} \log(1 - h(x_j))$
$H(x)$	$e^{-H(x_i)} - e^{-H(x_{i+1})}$	$1 - e^{-H(x_{i+1})}$	$e^{-H(x_i)}$	$\frac{e^{-H(x_i)} - e^{-H(x_{i+1})}}{e^{-H(x_i)}}$.

Two specific examples of how to derive one distribution representation when another is known (using the 5×5 matrix in Table 2.3) follow.

Example 2.13. Given $h(x) = 1/4$ for $x = 1, 2, \dots$, find $f(x_i)$ for $x_i = i$, $i \in \mathbf{Z}^+$.

Solution: Using the $(h(x), f(x_i))$ matrix element in Table 2.3,

$$f(x_i) = h(x_i) \prod_{j|x_j < x_i} [1 - h(x_j)] = \frac{1}{4} \prod_{j=1}^{x_i-1} \left[1 - \frac{1}{4} \right] = \frac{1}{4} \left(\frac{3}{4} \right)^{x_i-1} \quad x_i = i; i \in \mathbf{Z}^+,$$

which is a geometric distribution with $p = 1/4$. The geometric distribution is the only

discrete distribution with a constant hazard function (and memoryless property). \square

Example 2.14. Determine the SF (in numeric form) corresponding to the PDF shown in Figure 2.1.

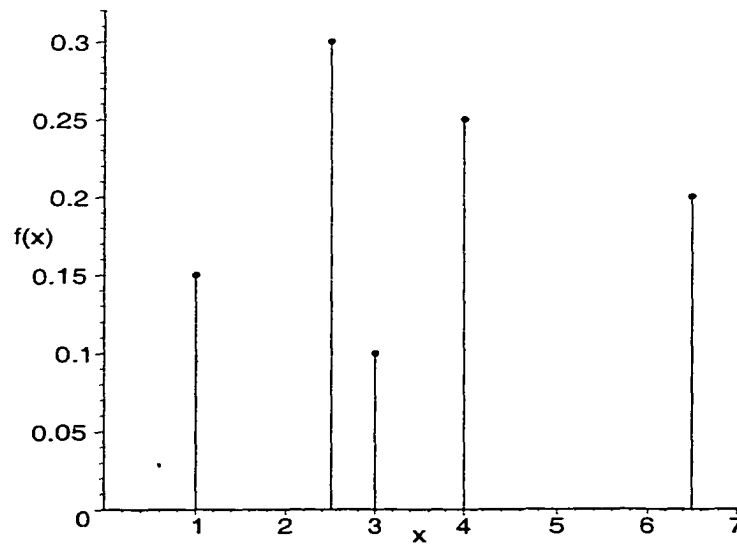


Figure 2.1: PDF for Example 2.14.

Solution: The numeric form of the PDF in Figure 2.1 is

$$f(x) = \begin{cases} 0.15 & x_1 = 1 \\ 0.3 & x_2 = 2.5 \\ 0.1 & x_3 = 3 \\ 0.25 & x_4 = 4 \\ 0.2 & x_5 = 6.5. \end{cases}$$

Using the $(f(x), S(x_i))$ matrix element in Table 2.3,

$$S(x_1) = \sum_{j=1}^5 f(x_j) = 1, \quad S(x_2) = \sum_{j=2}^5 f(x_j) = 0.85,$$

$$S(x_3) = \sum_{j=3}^5 f(x_j) = 0.55, \quad S(x_4) = \sum_{j=4}^5 f(x_j) = 0.45, \quad S(x_5) = 0.2. \quad \square$$

The algorithms for the PDF, CDF, SF, HF, and CHF procedures are similar and utilize the formulas in Table 2.2. When changing the “current representation” of a random variable to a “new representation,” the algorithm uses the Table 2.2 matrix element (“current representation”, “new representation”). Section 2.2.1 outlines the algorithm for just the PDF procedure. Section 2.2.2 briefly discusses some of the differences encountered in the algorithms for the other procedures, including IDF.

2.2.1 PDF

Let Ω be the support of the discrete random variable X . The PDF of X is defined as $f(x) = \Pr(X = x)$ for all $x \in \Omega$. The APPL `PDF(X, [x])` procedure has two arguments—the random variable X and an optional real number argument x . (An optional argument in a procedure is denoted by square brackets, i.e., `[x]`.) The `PDF(X, [x])` procedure returns either the

1. PDF of the random variable X in its APPL list-of-sublists format if only one argument is provided in the procedure call; i.e., `PDF(X)`,
2. probability value $\Pr(X = x)$ if both arguments are provided in the procedure call; i.e., `PDF(X, x)`.

For explanation purposes, let X be the random variable with PDF $f(x) = x/6$ for $x = 1, 2, 3$ whose PDF, CDF, SF, HF, CHF, and IDF are in Table 2.2. If we assign the variable `FX` to the CDF of X (in its *Dot* format), then the statement `PDF(FX)` returns the PDF of X in its APPL list-of-sublists format. The lines

```
> FX := [[x -> (x ^ 2 + x) / 12], [1 .. 3], ["Discrete", "CDF"]];
> PDF(FX);
```

produce the PDF

$$\left[\left[x \rightarrow \frac{x}{6} \right], [1 \dots 3], ["Discrete", "PDF"] \right].$$

Conversely, the statements

```
> FX := [[x -> (x ^ 2 + x) / 12], [1 .. 3], ["Discrete", "CDF"]];
> PDF(FX, 2);
```

compute $\Pr(X = 2)$, which is $1/3$.

The PDF algorithm first checks a discrete random variable X to determine if it has a *Dot* or *NoDot* support format. Although two distinct branches in the algorithm have been constructed to process random variables of each format separately, the basic functional relationships displayed in Table 2.3 are used in each branch. Whether X has a *Dot* or *NoDot* support format, one of the following bullets will be executed depending on X 's distribution representation (and support format in the case of IDF). Without loss of generality, we can assume $\Omega = \{x_1, x_2, \dots\}$, where $x_1 < x_2 < \dots$.

- If X has a PDF representation, do nothing to X .
- If X has a CDF representation, determine the PDF of X with the $(F(x), f(x_i))$ matrix element in Table 2.3:

$$f(x_i) = F(x_i) - F(x_{i-1}) \quad x_i \in \Omega,$$

where $i = 1, 2, \dots$ and $F(x_0) \equiv 0$.

- If X has a SF representation, determine the PDF of X with the $(S(x), f(x_i))$ matrix element in Table 2.3:

$$f(x_i) = S(x_i) - S(x_{i+1}) \quad x_i \in \Omega,$$

where $i = 1, 2, \dots$ and $S(x_{|\Omega|+1}) \equiv 0$ when $|\Omega|$ is finite.

- If X has a HF representation, determine the PDF of X with the $(h(x), f(x_i))$ matrix element in Table 2.3:

$$f(x_i) = h(x_i) \prod_{j|x_j < x_i} [1 - h(x_j)] \quad x_i \in \Omega.$$

where $i = 1, 2, \dots$ and $h(x_0) \equiv 0$.

- If X has a CHF representation, determine the PDF of X with the $(H(x), f(x_i))$ matrix element in Table 2.3:

$$f(x_i) = e^{-H(x_i)} - e^{-H(x_{i+1})} \quad x_i \in \Omega,$$

where $i = 1, 2, \dots$ and $H(x_{|\Omega|+1}) \equiv 0$ when $|\Omega|$ is finite.

- If X has an IDF representation with *Dot* support, then
 1. If there is a ceiling term as part of the inverse function (discussed in more detail in Section 2.2.2), extract the expression under the ceiling in order to solve $F^{-1}(x) = y$. Referring back to X with PDF $f(x) = x/6$ for $x = 1, 2, 3$, the IDF of X (provided in Table 2.2) is

$$F^{-1}(x) = \left\lceil -\frac{1}{2} + \frac{1}{2}\sqrt{1 + 48x} \right\rceil,$$

for $x = 1/6, 1/2, 1$. Rewrite $F^{-1}(x)$ as

$$F^{-1}(x) = -\frac{1}{2} + \frac{1}{2}\sqrt{1 + 48x}$$

This removal can be done in Maple using the `eval` command (McCarron, 2001).

2. Solve the equation

$$F^{-1}(x) = y$$

for x , yielding $F(y)$. For our particular X , solving $-\frac{1}{2} + \frac{1}{2}\sqrt{1+48x} = y$ for x yields $F(x) = \frac{x^2+x}{12}$, replacing y with x .

3. Find the *appropriate* inverse. If there is more than one solution to the equation $F^{-1}(x) = y$, determine the correct inverse by testing which solution $F(x)$ correctly calculates $F(F^{-1}(s)) = s$ for s equal to one of the following. The choices for s are considered in the bulleted order.

- $s = 1$ if $\min\{\Omega\} = -\infty$ and $\max\{\Omega\} = \infty$;
- $s = \max\{\Omega\}$, if $\max\{\Omega\} \neq \infty$;
- $s = \min\{\Omega\}$, if $\min\{\Omega\} \neq -\infty$;
- $s = \frac{\min\{\Omega\} + \max\{\Omega\}}{2}$.

4. Convert the CDF of X to its PDF representation by the using the $(F(x), f(x_i))$ matrix element in Table 2.3.

5. Determine the range of the PDF of X . The minimum support value is $F^{-1}(x_1)$ (where x_1 is the minimum support value of the IDF here) and the maximum support value is $F^{-1}(1)$. Since the minimum support value of the IDF for the example random variable X is $x_1 = 1/6$, then $F^{-1}(1/6) = 1$. Also, $F^{-1}(1) = 3$.

- If X has an IDF representation and a *NoDot* support, then the second sublist of the APPL list-of-sublists contains the CDF values of X and the first sublist contains the CDF support values. To compute the PDF of X , the first and second sublists of the IDF are swapped to form the CDF, and then the PDF is computed using the $(F(x), f(x_i))$ matrix element in Table 2.3.

In its numeric form, the IDF of our example random variable X is

$$F^{-1}(x) = \begin{cases} 1 & x = \frac{1}{6} \\ 2 & x = \frac{1}{2} \\ 3 & x = 1. \end{cases}$$

The APPL list-of-sublists is $[[1, 2, 3], [\frac{1}{6}, \frac{1}{2}, 1], [\text{"Discrete"}, \text{"IDF"}]]$. The CDF is found by swapping the first two sublists: $[[\frac{1}{6}, \frac{1}{2}, 1], [1, 2, 3], [\text{"Discrete"}, \text{"IDF"}]]$.

Lastly, the PDF is

$$\begin{aligned} f(1) &= \frac{1}{6} \\ f(2) &= F(2) - F(1) = \frac{1}{3} \\ f(3) &= F(3) - F(2) = \frac{1}{2}. \end{aligned}$$

After the PDF has been computed, the algorithm checks the number of arguments that were entered as part of the procedure, either one or two. If only one argument, X , is provided with the procedure, i.e., $\text{PDF}(X)$, then the PDF of X is returned in its list-of-sublists format. Otherwise, if two arguments, X and x , are provided with the procedure, then the probability value $\text{Pr}(X = x)$ is computed. The algorithm distinguishes between the two support formats when determining this probability value.

- If X has the *Dot* support format and we are computing $\text{Pr}(X = x)$, then:
 1. If the value $x_i \notin \Omega$, then return $\text{Pr}(X = x_i) = 0$. (Determining whether or not $x_i \in \Omega$ is discussed in further detail in Section 2.2.2.)
 2. If $x_i \in \Omega$, then compute and return the probability value $f(x_i)$ using the PDF formula already determined in the earlier part of the procedure.
- If X has the *NoDot* support format, then:
 1. Loop through the support values of the PDF searching for the value x_i in the second sublist. If x_i is found in position j , then its corresponding

probability value is in position j of the first sublist. Return the j th element in the first sublist. If the value x_i is not found in the second sublist, then return $\Pr(X = x_i) = 0$.

Two examples using the PDF procedure follow. Also, Example 1.2 in the introduction uses the PDF procedure to determine a probability value.

Example 2.15. Return to Example 2.13 and use APPL to determine $f(x)$ given $h(x) = 1/4$ for $x = 1, 2, \dots$

Solution: The statements

```
> X := [[x -> 1 / 4], [1 .. infinity], ["Discrete", "HF"]];
> PDF(X);
```

return the PDF $f(x) = \frac{1}{4}(\frac{3}{4})^{x-1}$ for $x = 1, 2, \dots$, as previously determined. \square

Example 2.16. (Adapted from Hogg & Craig, 1995, page 37) Cast a die two independent times and let X equal the absolute value of the difference of the two resulting values (the numbers on the up sides). The CDF of X is:

$$F(x) = \begin{cases} \frac{1}{6} & x = 0 \\ \frac{11x - x^2 + 6}{36} & x = 1, 2, \dots, 5. \end{cases}$$

Determine the PDF of X .

Solution: Define the APPL *NoDot* format of $F(x)$ as the random variable FX. Then use the PDF procedure to determine X 's numeric PDF.

```
> FX := [[1 / 6, 4 / 9, 2 / 3, 5 / 6, 17 / 18, 1], [0, 1, 2, 3, 4, 5],
         ["Discrete", "CDF"]];
> PDF(FX);
```

The APPL *NoDot* format of $f(x)$ is

$$\left[\left[\frac{1}{6}, \frac{5}{18}, \frac{2}{9}, \frac{1}{6}, \frac{1}{9}, \frac{1}{18} \right], [0, 1, 2, 3, 4, 5], ["Discrete", "PDF"] \right],$$

which can be converted by hand to the formulaic PDF:

$$f(x) = \begin{cases} \frac{1}{6} & x = 0 \\ \frac{6-x}{18} & x = 1, 2, \dots, 5. \end{cases} \quad \square$$

2.2.2 CDF, SF, HF, CHF, and IDF

The CDF, SF, HF, and CHF procedures follow the same algorithmic steps as the PDF procedure. The main difference between the procedures is the formulas used for transforming one representation into another from Table 2.3. The PDF procedure uses column one of Table 2.3, while the CDF, SF, HF, and CHF procedures use columns two, three, four, and five, respectively.

Another difference is how the PDF and HF procedures handle two arguments, i.e., $\text{PDF}(X, x)$, compared to how CDF, SF, and CHF handle two arguments. The way a functional representation procedure (e.g., PDF) treats a two argument input stems from the representation's mathematical definition. Returning to the example random variable X with PDF $f(x) = x/6$ for $x = 1, 2, 3$, its PDF, CDF, SF, and CHF are

$$f(x) = \begin{cases} \frac{1}{6} & x = 1 \\ \frac{1}{3} & x = 2 \\ \frac{1}{2} & x = 3 \end{cases} \quad F(x) = \begin{cases} 0 & x < 1 \\ \frac{1}{6} & 1 \leq x < 2 \\ \frac{1}{2} & 2 \leq x < 3 \\ 1 & x \geq 3 \end{cases}$$

$$S(x) = \begin{cases} 1 & x \leq 1 \\ \frac{5}{6} & 1 < x \leq 2 \\ \frac{1}{2} & 2 < x \leq 3 \\ 0 & x > 3 \end{cases} \quad H(x) = \begin{cases} 0 & x \leq 1 \\ -\log\left(\frac{5}{6}\right) & 1 < x \leq 2 \\ -\log\left(\frac{1}{2}\right) & 2 < x \leq 3. \end{cases}$$

Since the CDF and SF of X are defined for $x \in \mathbb{R}$ and the CHF is defined for $x \leq 3$, they could be formulaically defined more generally in Table 2.2 as

$$F(x) = \begin{cases} 0 & x < 1 \\ \frac{[x]^2 + [x]}{12} & 1 \leq x < 3 \\ 1 & x \geq 3, \end{cases}$$

$$S(x) = \begin{cases} 1 & x \leq 1 \\ \frac{-[x]^2 + [x] + 12}{12} & 1 < x \leq 3 \\ 0 & x > 3, \end{cases}$$

$$H(x) = \begin{cases} 0 & x \leq 1 \\ -\log\left(\frac{-[x]^2 + [x] + 12}{12}\right) & 1 < x \leq 3. \end{cases}$$

The CDF, SF, and CHF data structures were developed to display the values of these functions at x_1, x_2, \dots , hence the term “discrete” data structure. Although the CDF, SF, and CHF representations of X are defined on a continuous interval of values, their counterparts, PDF and HF, are not. In APPL, discrete random variables are defined for a discrete set of values. Users of APPL who are familiar with probability theory will know that, for example, if $F(x)$ is displayed as

$$F(x) = \frac{x^2}{12} + \frac{x}{12} \quad x = 1, 2, 3,$$

that when $1 \leq x < 2$, $F(x) = F(1) = 1/6$. Consistency of format between all five functional representations was a priority when the *Dot* and *NoDot* data structures and their distribution representations were constructed.

Accuracy and fact are not jeopardized as a consequence of this consistency. Thus, when a procedure, such as CDF, computes $\text{CDF}(X, 1.5)$ (for the example random variable X), it returns $1/6$. Although the format of the CDF representation of X does not display its continuous nature, i.e.,

$$\left[\left[x \rightarrow \frac{x^2}{12} + \frac{x}{12} \right] [1, 2, 3], ["Discrete", "CDF"] \right]$$

the computation $\text{CDF}(X, 1.5)$ yields the correct value of $1/6$, not 0 or $\frac{1.5^2}{12} + \frac{1.5}{12} = 0.3125$. In order to have the CDF algorithm compute this value correctly in the procedure call $\text{CDF}(X, \mathbf{x}), \lfloor \lfloor x \rfloor \rfloor$, where $\lfloor \lfloor x \rfloor \rfloor$ is defined as the largest support value less than or equal to x , rather than x , is substituted into the formula for $F(x)$. In the SF and CHF procedures, $\lceil \lceil x \rceil \rceil$, where $\lceil \lceil x \rceil \rceil$ is defined as the smallest support value greater than or equal to x , is substituted into the formulas $S(x)$ and $H(x)$, respectively, rather than x .

Although the PDF and HF procedures are less complex than CDF, SF, and CHF from one perspective (since for $x_i \notin \Omega$, $f(x_i) = 0$ and $h(x_i) = 0$), checking whether or not $x_i \in \Omega$ presents its own obstacles. If $|\Omega|$ is finite, which implies that x_1 and $x_{|\Omega|}$ are finite, then it is not difficult (albeit time-consuming in some instances) in APPL procedures to confirm whether or not $x_i \in \Omega$. If $|\Omega|$ is infinite, then $x_1 = -\infty$ and/or $x_{|\Omega|} = \infty$. Although an algorithm for determining whether $x_i \in \Omega$ in this situation has not yet been implemented in APPL procedures, the following paragraph describes one algorithmic method that is under consideration.

Suppose the *Dot* support is $[a..b, k, g(x)]$, which means that $a = \min\{\Omega\}$, $b = \max\{\Omega\}$, and the support is incremented by k and transformed by $g(x)$. In order to

check if $x_i \in \Omega$, do the following

1. Find the inverse function $g^{-1}(x)$ of $g(x)$ (if it exists). If $g(x)$ has more than one inverse, the “correct” inverse needs to be determined.
2. Compute $v = g^{-1}(x_i)$.
3. Determine if $v \equiv a \pmod{k}$, where $v \equiv a \pmod{k}$ provided $k|(v - a)$. If it is, then $x_i \in \Omega$.

Suppose $\Omega = \{27, 125, 343, \dots\}$, for example, and let $x_i = 3375$. The *Dot* support of Ω is $[3 .. \infty, 2, x \rightarrow x^3]$. Since $g^{-1}(x) = \sqrt[3]{x}$ and $\sqrt[3]{3375} = 15 \equiv 3 \pmod{2}$, then $x_i = 3375 \in \Omega$. Conversely, the value $x_i = 2748 \notin \Omega$ since $\sqrt[3]{2748} \notin \mathbf{Z}$ [and thus $\sqrt[3]{2748} \not\equiv 3 \pmod{2}$] and $x_i = 1728 \notin \Omega$ since $\sqrt[3]{1728} = 12 \not\equiv 3 \pmod{2}$. Unfortunately, problems will arise in this algorithm when $g(x)$ either does not have an inverse or has more than one inverse and the appropriate one cannot be determined.

Because row one of Table 2.3, ($f(x)$, “new representation”), relies on Maple’s `sum` procedure, this is another area where `CDF`, `SF`, `HF`, and `CHF` experience some difficulties. A probable candidate for a fractious transformation to a new functional representation is the Poisson random variable. Since the Poisson random variable is a pre-defined random variable in APPL, we can assign the variable `X` as a Poisson random variable with a mean of λ with the statement

```
> X := PoissonRV(lambda);
```

APPL returns the random variable in its list-of-sublists PDF representation as

$$\left[\left[x \rightarrow \frac{\lambda^x e^{-\lambda}}{x!} \right], [0 .. \infty], ["Discrete", "PDF"] \right].$$

Changing `X` to its CDF representation in APPL with the statement

```
> CDF(X);
```

yields

$$\left[\left[x \rightarrow \frac{\Gamma(x+1, \lambda)}{\Gamma(x+1)} \right] [0.. \infty], ["Discrete", "CDF"] \right].$$

The SF, HF, and CHF procedures produce results containing the gamma and incomplete gamma functions also.

In Maple, the gamma function is defined for $\Re(z) > 0$ by $\Gamma(z) = \int_0^\infty e^{-t} t^{z-1} dt$, and is extended to the rest of the complex plane, less the non-positive integers, by analytic continuation. The incomplete gamma function is defined as $\Gamma(a, z) = \Gamma(a) - \frac{z^a}{a} {}_1F_1(a, 1+a, -z)$ where ${}_1F_1$ is the confluent hypergeometric function. In Maple notation, ${}_1F_1(a, 1+a, -z) = \text{hypergeom}([a], [1+a], -z)$. For $\Re(a) > 0$, we also have the integral representation $\Gamma(a, z) = \int_z^\infty e^{-t} t^{a-1} dt$.

Although this is not a tractable representation for the CDF of a Poisson random variable, cumulative probabilities can still be easily computed, as shown in the next example.

Example 2.17. (Ross, 1998, page 155) Consider an experiment that consists of counting the number of α -particles given off in an one-second interval by one gram of a radioactive material. If we know from past experience that, on the average, 3.2 such α -particles are given off, what is a good approximation to the probability that no more than three α -particles will appear?

Solution: If we think of the gram of radioactive material as consisting of a large number n of atoms, each of which has probability $3.2/n$ of disintegrating and sending off an α -particle during the second considered, then we see that, to a very close approximation, the number of α -particles given off will be a Poisson random variable with parameter $\lambda = 3.2$. In order to compute the desired probability, we use the APPL statements

```
> X := PoissonRV(3.2);
> CDF(X, 3);
```

which yield the approximate probability 0.6025. \square

Before discussing the IDF procedure, one last example of transforming from one representation to another is presented in Example 2.18.

Example 2.18. Let n be a positive integer. Let X be a random variable with HF $h(x)$ defined as

$$h(x) = \frac{2x}{n^2 + n - x^2 + x} \quad x = 1, 2, \dots, n.$$

Determine the SF of X for $x_i = 1, 2, \dots, n$.

Solution: Using the $(h(x), S(x_i))$ matrix element in Table 2.3, $S(x_i)$ is

$$S(x_i) = \prod_{j=1}^{x_i-1} (1 - h(x_j)) \quad x_i = i; i = 1, 2, \dots, n.$$

Letting $x_i = x$, we can simplify $S(x)$ in the following manner:

$$\begin{aligned} S(x) &= \prod_{t=1}^{x-1} (1 - h(t)) \\ &= \prod_{t=1}^{x-1} \left(1 - \frac{2t}{n^2 + n - t^2 + t} \right) \\ &= \prod_{t=1}^{x-1} \frac{n^2 + n - t^2 - t}{n^2 + n - t^2 + t} \\ &= \prod_{t=1}^{x-1} \frac{(n-t)(n+t) + (n-t)}{(n-t)(n+t) + (n+t)} \\ &= \prod_{t=1}^{x-1} \frac{(n-t)(n+t+1)}{(n+t)(n-t+1)} \quad x = 1, 2, \dots, n. \end{aligned}$$

Claim:

$$\prod_{t=1}^{x-1} \frac{(n-t)(n+t+1)}{(n+t)(n-t+1)} = \frac{(n-x+1)(n+x)}{n(n+1)} \quad (2.1)$$

for $x = 1, 2, \dots, n$.

Proof of Claim: The claim can be proved by induction on x . Since the left hand side of equation (2.1) is an empty product for $x = 1$, it follows that $S(1) = \prod_{t=1}^0 \frac{(n-t)(n+t+1)}{(n+t)(n-t+1)} = 1$. The right hand side of equation (2.1) for $x = 1$ is $\frac{(n-1+1)(n+1)}{n(n+1)} = 1$. Thus, equation (2.1) is true for $x = 1$.

Now suppose equation (2.1) holds for a certain integer x such that $1 \leq x \leq n-1$. Then using this assumption and starting with the left hand side of equation (2.1) with x replaced by $x+1$, we find:

$$\begin{aligned} \prod_{t=1}^{x+1} \frac{(n-t)(n+t+1)}{(n+t)(n-t+1)} &= \prod_{t=1}^{x-1} \frac{(n-t)(n+t+1)}{(n+t)(n-t+1)} \cdot \frac{(n-x)(n+x+1)}{(n+x)(n-x+1)} \\ &= \frac{(n-x+1)(n+x)}{n(n+1)} \cdot \frac{(n-x)(n+x+1)}{(n+x)(n-x+1)} \quad \text{by assumption} \\ &= \frac{(n-x)(n+x+1)}{n(n+1)}. \end{aligned}$$

The result is the right hand side of equation (2.1) with $x+1$ replacing x . Hence, by induction, equation (2.1) holds for $x = 1, 2, \dots, n$.

Using the claim, we can write $S(x)$ as:

$$S(x) = \frac{(n+x)(n-x+1)}{(n+1)n} \quad x = 1, 2, \dots, n.$$

In APPL, the statements

```
> unassign('n');
> hX := [[x -> 2 * x / (n ^ 2 + n - x ^ 2 + x)], [1 .. n],
        ["Discrete", "HF"]];
> SX := SF(hX);
```

produce the desired SF that we previously computed by hand. \square

Computing the IDF in the *Dot* format

Let X have a CDF F_X in which $F_X(y) = x$. For $x \in [0, 1]$, the inverse distribution function (IDF) $F_X^{-1}(x)$ performs the inverse mapping of x to y , i.e., $F^{-1}(x) = y$. Like the PDF, CDF, SF, HF, and CHF procedures, the $\text{IDF}(X, [\mathbf{x}])$ procedure returns either the

1. IDF of the random variable X in the APPL list-of-sublists format if only the argument X is provided in the procedure call, i.e., $\text{IDF}(X)$;
2. quantile value y such that $\Pr(X \leq y) = x$ if both arguments are provided in the procedure call, i.e., $\text{IDF}(X, \mathbf{x})$.

Unlike the CDF, SF, and CHF procedures, the IDF functional representation in sublist one is displayed as a continuous function. The IDF of a random variable X is defined for all real numbers $x \in [0, 1]$. For the example random variable X with PDF $f(x) = x/6$ for $x = 1, 2, 3$, its APPL IDF in *Dot* format is

$$\left[\left[x \rightarrow \left[-\frac{1}{2} + \frac{1}{2}\sqrt{1 + 48x} \right] \right], \left[1 .. 3, x \rightarrow \frac{x^2 + x}{12} \right], ["Discrete", "IDF"] \right],$$

where the second sublist $\left[1 .. 3, x \rightarrow \frac{x^2 + x}{12} \right]$ indicates IDF support values at $\frac{1}{6}$, $\frac{1}{2}$, and

1. Although the first support value is $x_1 = 1/6$, APPL understands the IDF is still defined for $0 \leq x \leq 1/6$. When another functional representation procedure, such as PDF, receives an IDF with a ceiling expression, it only takes one additional line in that procedure to remove the ceiling from the expression before computing an inverse. In the case of IDF, the ceiling definition was believed to be the appropriate way to define this function in its first sublist.

In a much easier scenario, if a “non-IDF” functional representation of X is reported to IDF in its *NoDot* format, such as

```
> X := [[1 / 6, 1 / 3, 1 / 2], [1, 2, 3], ["Discrete", "PDF]];
> IDF(X);
```

then the IDF algorithm simply computes the CDF of X and then switches the first two sublists to form the IDF.

2.3 Algorithms for Fundamental Procedures

Before attempting algorithm construction for non-standard discrete random variable processes (e.g., `OrderStat`), some of the straightforward algorithms were written to handle fundamental random variable manipulation (e.g., computing the mean of a random variable). The several procedures discussed in this section had already been written for continuous distributions. All APPL procedures that handle discrete random variables are equipped to process random variables in both the *Dot* and *NoDot* formats.

2.3.1 Verifying the Validity of a PDF

Let X be a discrete random variable with support $\Omega = \{x_1, x_2, \dots\}$. A discrete probability density (mass) function (PDF) must satisfy the following two conditions:

- $\Pr(X = x_i) \geq 0$ for each $x_i \in \Omega$;
- $\sum_{\text{all } x_i \in \Omega} \Pr(X = x_i) = 1$.

The APPL `VerifyPDF` procedure has one argument, a random variable X , and the procedure's purpose is to verify that X satisfies these two conditions. When making the procedure call `VerifyPDF(X)`, either a message is returned stating that X has a valid PDF or that it does not. The following example illustrates the usage of this procedure.

Example 2.19. (Karian & Tanis, 1999, page 62) Verify that the probabilities of a geometric random variable with parameter p sum to 1.

Solution: X can be defined as a geometric random variable with parameter p using the pre-defined APPL random variable `GeometricRV(p)`. When the `GeometricRV` procedure receives a symbolic argument (parameter), it uses the Maple `assume` command to correctly assume that the symbolic parameter is inclusively between 0 and 1. Thus, the statements

```
> X := GeometricRV(p);
> VerifyPDF(X);
```

indicate that X has a valid PDF. □

2.3.2 Calculating Summary Characteristics

Moments describe certain behaviors of random variables. Measures of *central tendency*, for example, such as the mean or median, refer to the “average” or “central” values of a random variable. Measures of *dispersion*, such as the standard deviation, are used to measure the spread of a random variable’s distribution. Skewness quantifies a random variable’s symmetry about its mean, while kurtosis measures the flatness (or “peakedness”) of a random variable.

If X is a discrete random variable with support $\Omega = \{x_1, x_2, \dots\}$ and PDF $f(x)$, then the *mean* (or *expected value*) of X is defined by

$$E[X] = \mu = \sum_{\text{all } x_i \in \Omega} x_i \cdot f(x_i).$$

More generally, if $g(X)$ is any function of X , such as $g(X) = X^2$, then

$$E[g(X)] = \sum_{\text{all } x_i \in \Omega} g(x_i) \cdot f(x_i).$$

The `ExpectedValue` procedure requires one argument, a random variable X , and can be given an additional argument, the function $g(X)$. When only one argument is provided for the `ExpectedValue` procedure, i.e., `ExpectedValue(X)`, the mean of the random variable X is returned. By default, the procedure assumes $g(X) = X$ when no second argument is given. The `Mean`, `Variance`, `Kurtosis`, and `Skewness` procedures all make use of the `ExpectedValue` procedure. This is efficient since variance, the coefficient of skewness, γ_1 , and the coefficient of kurtosis, γ_2 , for a random variable X can be calculated as

$$\begin{aligned}\text{Var}(X) &= E[(X - \mu)^2] = E[X^2] - \mu^2, \\ \gamma_1 &= E\left[\left(\frac{X - \mu}{\sigma}\right)^3\right] = \frac{E[X^3] - 3\mu E[X^2] + 2\mu^3}{[\text{Var}(X)]^{3/2}}, \text{ and} \\ \gamma_2 &= E\left[\left(\frac{X - \mu}{\sigma}\right)^4\right] = \frac{E[X^4] - 4\mu E[X^3] + 6\mu^2 E[X^2] - 3\mu^4}{[\text{Var}(X)]^2}.\end{aligned}$$

Two examples of the usage of the `ExpectedValue` procedure follow. In the second example, `Example 2.21`, notice that $g(x)$ is non-linear.

Example 2.20. (Ross, 1998, page 167) Find the expected value and the variance of the number of times one must throw a die until the outcome one has occurred four times.

Solution: Since the random variable of interest is a negative binomial with parameters $r = 4$ and $p = 1/6$, then the statements

```
> r := 4;
> p := 1 / 6;
> X := NegativeBinomialRV(r, p);
> ExpectedValue(X);
> Variance(X);
```

yield the correct results: $E[X] = 24$ and $\text{Var}(X) = 120$. □

Example 2.21. (Ross, 1998, page 185) Let X be a binomial random variable with parameters n and p . Show that

$$E \left[\frac{1}{X+1} \right] = \frac{1 - (1-p)^{n+1}}{(n+1)p}.$$

Solution: Since we want to compute $E \left[\frac{1}{X+1} \right]$, let $g(X) = 1/(X+1)$. The APPL statements

```
> X := BinomialRV(n, p);
> ExpectedValue(X, x -> 1 / (x + 1));
```

produce the desired expected value. □

2.3.3 Discrete Plotting Functions

Especially for newcomers to probability theory, plots of the various representations of a random variable can provide much insight into the definition and structure of that random variable. The `PlotDist` procedure requires only one argument, a random variable X , but two additional arguments can be provided to indicate the plotting range, `low` and `high`. The procedure can plot a random variable in any of its six functional representations. The algorithm employed by the `PlotDist` procedure utilizes the list-of-sublists data structure to glean the necessary information for a plot. Graphs for CDF's, SF's, CHF's, and IDF's, for instance, are plots of step functions with jumps at the support values specified in sublist two. Open and closed circles at the ends of the steps indicate exclusion and inclusion, respectively, of support values. One example of a CDF plot follows.

Example 2.22. (Karian & Tanis, 1999, page 40) Let $f(x) = (5 - x)/10$ for $x =$

1, 2, 3, 4 be a PDF of the discrete random variable X . Use `PlotDist` to depict the CDF of X .

Solution: Instead of first converting X 's PDF representation into a CDF representation, one can directly define the PDF and compose the CDF conversion with `PlotDist`. The plot of the desired CDF can be executed with the statements

```
> X := [[x -> (5 - x) / 10], [1 .. 4], ["Discrete", "PDF"]];  
> PlotDist(CDF(X));
```

The corresponding plot is displayed in Figure 2.2.

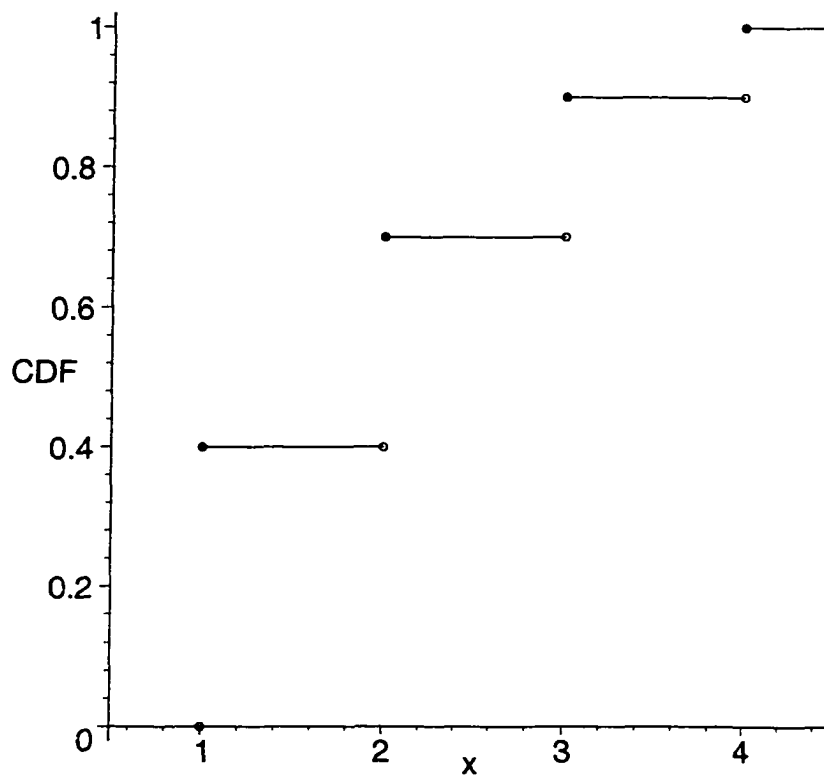


Figure 2.2: CDF for Example 2.22.

Chapter 3

Order Statistics

As evidenced by over six hundred references cited in the book “Order Statistics” by David (1970), the theory and applications of order statistics appear in many areas of statistical theory and practice. Most authors of introductory textbooks only address order statistics drawn from continuous parent populations due to the mathematical intractability in the discrete case. Results for order statistics drawn from discrete parent populations are sparse and usually specialized to fit one particular discrete population. The purpose of this chapter is to present algorithms for determining distributions of order statistics drawn from standard (e.g., binomial, geometric) and non-standard discrete parent populations. The algorithms handle discrete parent populations with finite or infinite support, and sampling with or without replacement. Computer algebra systems make it feasible to derive (or compute) distributions of order statistics from parent populations with formulaic (e.g., $f(x) = \frac{x}{2!}$, $x = 1, 2, \dots, 6$) or numeric (e.g., $f(1) = \frac{1}{2}$, $f(3) = \frac{1}{8}$, $f(7) = \frac{3}{8}$) PDF representations. The development of these algorithms provides the general scientific community easy access to many discrete order statistic distributions.

Let X_1, X_2, \dots, X_n be n independent and identically distributed (iid) random variables defined on Ω , each with CDF $F(x)$ and PDF $f(x)$. Let $X_{(1)} \leq X_{(2)} \leq \dots \leq$

$X_{(n)}$ denote these random variables rearranged in non-descending order of magnitude. Thus, $X_{(r)}$ is the r th smallest order statistic of the sample, $r = 1, 2, \dots, n$. Since the order statistics are random variables, it is possible to compute the probabilities that they take on various values in their support.

When the population is continuous, the PDF of the r th order statistic can be expressed easily since the probability that any two $X_{(j)}$'s are the same is 0. As is well known (e.g., Casella & Berger, 1990, page 232), the PDF of $X_{(r)}$ is

$$f_{X_{(r)}}(x) = \frac{n!}{(r-1)!(n-r)!} f(x) [F(x)]^{r-1} [1-F(x)]^{n-r} \quad x \in \Omega$$

for $r = 1, 2, \dots, n$. Two examples of order statistics with continuous parent populations follow.

Example 3.1. (Adapted from Feller, 1971, pages 20–21) Let X_1, X_2, \dots, X_n be iid exponential random variables with rate parameter λ . Find the PDF of $X_{(r)}$.

Solution: Using the above equation, the PDF of the r th order statistic is

$$f_{X_{(r)}}(x) = \frac{n!}{(r-1)!(n-r)!} \cdot \lambda e^{-\lambda x} \cdot (1 - e^{-\lambda x})^{r-1} \cdot (e^{-\lambda x})^{n-r} \quad x > 0$$

for $r = 1, 2, \dots, n$. □

Example 3.2. (Adapted from Ross, 1998, pages 278–279) Consider a sample of size $n = 5$ from a beta distribution with parameters $\alpha = \frac{1}{2}$ and $\beta = 2$. Compute the probability that the median is in the interval $(0, \frac{1}{4})$.

Solution: Again, using the above equation, the PDF of the median $r = 3$ is

$$f_{X_{(3)}}(x) = \frac{45 \left(-\frac{1}{2}x^{3/2} + \frac{3}{2}\sqrt{x}\right)^2 \left(1 + \frac{1}{2}x^{3/2} - \frac{3}{2}\sqrt{x}\right)^2 (1-x)}{2\sqrt{x}} \quad 0 < x < 1.$$

Thus

$$\begin{aligned}
\Pr(0 < X_{(3)} < \frac{1}{4}) &= \int_0^{1/4} f_{X_{(3)}}(x) dx \\
&= \left(-\frac{3}{16}x^{15/2} + \frac{45}{16}x^{13/2} - \frac{15}{16}x^6 - \frac{135}{8}x^{11/2} + \frac{45}{4}x^5 + \frac{395}{8}x^{9/2} - \frac{405}{8}x^4 \right. \\
&\quad \left. - \frac{1035}{16}x^{7/2} + \frac{405}{4}x^3 + \frac{189}{16}x^{5/2} - \frac{1215}{16}x^2 + \frac{135}{4}x^{3/2} \right) \Bigg|_0^{1/4} \\
&= \frac{429913}{524288} \\
&\cong 0.81999. \quad \square
\end{aligned}$$

If X_1, X_2, \dots, X_n is a random sample from a discrete population, then the PDF of the r th order statistic cannot always be expressed as a single formula, as displayed in the continuous case examples. When working with discrete random variables, the computation of the PDF of the r th order statistic will fall into one of several categories, depending on the sampling convention (with or without replacement), the random variable's PDF representation (formulaic or numeric), the random variable's support (finite or infinite), and the random variable's distribution (equally likely or non-equally likely). A taxonomy of these categories appears in Figure 3.1.

3.1 Implementation for Discrete Populations

This section presents an algorithm, `OrderStat`, for computing the PDF of order statistics sampled from discrete parent populations. Additionally, the algorithm computes the PDF of order statistics from continuous populations, as in the examples provided previously. The algorithm, included in Appendix A, is implemented in Maple.

After a discrete random variable X is defined in its list-of-sublists format, the `OrderStat` algorithm computes the PDF of the r th order statistic given that n items are sampled either with or without replacement from the discrete parent population.

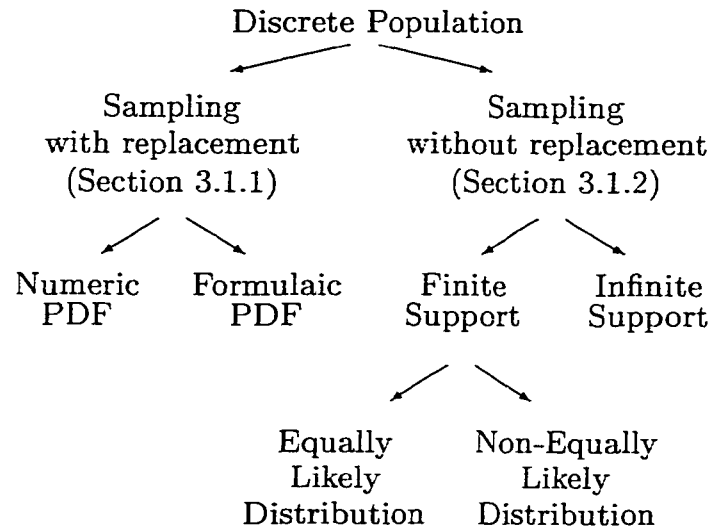


Figure 3.1: Categorization of discrete order statistics by sampling convention, PDF representation, support type, and population probability distribution.

If the random variable X has finite support in the *Dot* case and sampling is to occur *without* replacement, the `ConvertToNumeric` procedure modifies the first two sublists of the random variable to the standard *NoDot* format. For example, if the first two sublists of the random variable are entered in the *Dot* format as $[x \rightarrow \frac{x}{16}], [1 .. 7, 2]$ (where $k = 2$ is the increment value for the range), then `ConvertToNumeric` rewrites the sublists in the standard *NoDot* format as $[\frac{1}{16}, \frac{3}{16}, \frac{5}{16}, \frac{7}{16}], [1, 3, 5, 7]$. The conversion is necessary since the *without* replacement part of the algorithm requires a *list* of support values for the construction of permutations and combinations. The implementation steps of the `OrderStat` algorithm are explained in more detail in the “Sampling With Replacement” and “Sampling Without Replacement” subsections.

3.1.1 Sampling With Replacement

If only three arguments, X , n , and r , are provided as `OrderStat` arguments, then the items are assumed to be sampled from the population with replacement by default. The APPL `CDF` and `SF` procedures determine the cumulative distribution function,

$F(x) = \Pr(X \leq x)$, and survivor function, $S(x) = \Pr(X \geq x)$, respectively, of X in the list-of-sublists format. A branch in the with replacement portion of the code then occurs based on the form of the population's PDF, numeric or formulaic. The "Numeric PDF" and "Formulaic PDF" subsections detail the subsequent steps for each PDF representation. A random variable with a *Dot* support format *always* has a formulaic PDF. A random variable with a *NoDot* support format, although it may initially have a formulaic PDF representation, is converted to the standard *NoDot* format with a numeric PDF at the onset of the `OrderStat` procedure.

Numeric PDF

If the random variable X (in its APPL list-of-sublists) has a *NoDot* format, then its PDF is given as or converted to a Maple numeric list. For example, the random variable X with PDF

$$f(x) = \begin{cases} \frac{1}{4} & x = 5 \\ \frac{1}{2} & x = 6 \\ \frac{1}{4} & x = 10, \end{cases}$$

is entered as a random variable in APPL as

```
> X := [[1 / 4, 1 / 2, 1 / 4], [5, 6, 10], ["Discrete", "PDF"]];
```

If X has a *NoDot* format, then its support, Ω , must be finite. For this particular example, $\Omega = \{5, 6, 10\}$. Assuming (without loss of generality) that $\Omega = \{1, 2, \dots, N\}$, the PDF of $X_{(r)}$ when n items are sampled with replacement is given by:

$$f_{X_{(r)}}(x) = \begin{cases} \sum_{w=0}^{n-r} \binom{n}{w} [f(1)]^{n-w} [S(2)]^w & x = 1 \\ \sum_{u=0}^{r-1} \sum_{w=0}^{n-r} \binom{n}{u, n-u-w, w} [F(x-1)]^u [f(x)]^{n-u-w} [S(x+1)]^w & x = 2, 3, \dots, N-1 \\ \sum_{u=0}^{r-1} \binom{n}{u} [F(N-1)]^u [f(N)]^{n-u} & x = N. \end{cases}$$

The formula above is not valid in the special case when the discrete population consists of only one item, i.e., $N = 1$. The PDF of the r th order statistic with $N = 1$ is simply $f_{X_{(r)}}(1) = 1$.

The formula for calculating the PDF of $X_{(r)}$ is a direct result of the following observation. In order for the r th order statistic to take on the value x , the $r - 1$ order statistics preceding the r th index position must be less than or equal to x and the $n - r$ order statistics following the r th index position must be greater than or equal to x . More specifically, there must be *between* 0 and $r - 1$ values less than x , and *between* 0 and $n - r$ values greater than x , with all other values equal to x . The general formula for $x = 2, 3, \dots, N - 1$ is obtained by using the CDF, $F(x - 1)$, to determine the probability of obtaining a value less than or equal to $x - 1$, the PDF, $f(x)$, to determine the probability of obtaining x , and the SF, $S(x + 1)$, to determine the probability of obtaining a value greater than or equal to $x + 1$. The multinomial coefficient calculates the number of combinations that yield a specific ordering.

As a brief illustration of this formula, let X be a discrete random variable that can assume $N = 4$ values with PDF

$$f(x) = \begin{cases} 0.2 & x = 1 \\ 0.4 & x = 2 \\ 0.3 & x = 3 \\ 0.1 & x = 4. \end{cases}$$

Then the CDF and SF of X , respectively, are

$$F(x) = \begin{cases} 0 & x < 1 \\ 0.2 & 1 \leq x < 2 \\ 0.6 & 2 \leq x < 3 \\ 0.9 & 3 \leq x < 4 \\ 1 & x \geq 4 \end{cases} \quad S(x) = \begin{cases} 1 & x \leq 1 \\ 0.8 & 1 < x \leq 2 \\ 0.4 & 2 < x \leq 3 \\ 0.1 & 3 < x \leq 4 \\ 0 & x > 4. \end{cases}$$

Sample $n = 3$ values with replacement from the population. In order to calculate $f_{X_{(2)}}(3)$, i.e., the probability that the second order statistic (the median, $r = 2$) takes on the value $x = 3$, simplify the sum

$$f_{X_{(2)}}(3) = \sum_{u=0}^1 \sum_{w=0}^1 \binom{3}{u, 3-u-w, w} [F(2)]^u [f(3)]^{3-u-w} [S(4)]^w.$$

The first term in the summation

$$\binom{3}{0, 3, 0} \cdot (0.6)^0 \cdot (0.3)^3 \cdot (0.1)^0 = 0.3^3,$$

is the probability of drawing all threes. The second term

$$\binom{3}{0, 2, 1} \cdot (0.6)^0 \cdot (0.3)^2 \cdot (0.1)^1 = 3 \cdot (0.3)^2 \cdot 0.1,$$

is the probability of drawing two threes and a value greater than or equal to four (which can only be the value four in this example). This sample can be drawn three different ways: 3-3-4, 3-4-3, or 4-3-3. The third term

$$\binom{3}{1, 2, 0} \cdot (0.6)^1 \cdot (0.3)^2 \cdot (0.1)^0 = 3 \cdot 0.6 \cdot (0.3)^2,$$

is the probability of drawing two threes and one value less than or equal to two.

Letting L represent the value one or two, the three orderings are: $L-3-3$, $3-L-3$, $3-3-L$. Finally, the fourth term is the probability of drawing one three, one value less than or equal to two, and one value greater than or equal to four. This fourth term is

$$\binom{3}{1, 1, 1} \cdot (0.6)^1 \cdot (0.3)^1 \cdot (0.1)^1 = 6 \cdot 0.6 \cdot 0.3 \cdot 0.1.$$

Using the previous notation, the six orderings are: $L-3-4$, $L-4-3$, $3-L-4$, $3-4-L$, $4-L-3$, $4-3-L$. Thus, $f_{X(2)}(3) = 0.3^3 + 3 \cdot (0.3)^2 \cdot 0.1 + 3 \cdot 0.6 \cdot (0.3)^2 + 6 \cdot 0.6 \cdot 0.3 \cdot 0.1 = 0.324$.

Formulaic PDF

If the support of X is countably infinite, i.e., $|\Omega| = \{x_i \mid i = 1, 2, 3, \dots\}$, then the PDF of X must be entered in its APPL list-of-sublists format with a formulaic PDF representation. For example, if $X \sim \text{Poisson}(4)$ with PDF $f(x) = \frac{4^x e^{-4}}{x!}$, $x = 0, 1, 2, \dots$, then X is defined in APPL with the statement

```
> X := [[x -> ((4 ^ x) * exp(-4)) / x!], [0 .. infinity],
        ["Discrete", "PDF"]];
```

It is impossible to write a PDF with infinite support in the APPL standard *NoDot* format. Even if a random variable does not have infinite support, it may still have a formulaic PDF in the APPL list-of-sublists format. A binomial(5, 1/2) random variable with PDF $f(x) = \frac{120}{2^5 (5-x)! x!}$, $x = 0, 1, \dots, 5$, can be defined in APPL with the statement

```
> X := [[x -> 120 / (2 ^ 5 * (5 - x)! * x!)], [0 .. 5],
        ["Discrete", "PDF"]];
```

The calculation of the PDF of $X_{(r)}$ in the formulaic case, whether the support is finite or infinite, is similar to the calculation in the numeric PDF case. The main exception is that the formula used in the numeric PDF case can be used for values of

x arbitrarily large:

$$f_{X_{(r)}}(x) = \begin{cases} \sum_{w=0}^{n-r} \binom{n}{w} [f(1)]^{n-w} [S(2)]^w & x = 1 \\ \sum_{u=0}^{r-1} \sum_{w=0}^{n-r} \binom{n}{u, n-u-w, w} [F(x-1)]^u [f(x)]^{n-u-w} [S(x+1)]^w & x = 2, 3, \dots \end{cases}$$

Since the formula assumes that the support of X is $\Omega = \{1, 2, \dots, N\}$ or $\Omega = \{1, 2, \dots\}$, the implementation accepts distributions with finite supports or infinite *right-hand* tails. Discrete distributions with infinite left-hand tails could be handled in a similar manner. The APPL code presently issues an error message from the `OrderStat` procedure when there is an infinite left-hand tail.

Since it is impossible to execute the above formula for infinitely many values of x if the random variable X has infinite support, a general expression for $f_{X_{(r)}}(x)$ (in terms of x) is obtained by taking advantage of Maple's ability to sum and simplify complicated symbolic expressions. Maple finds and simplifies (in symbolic terms) the double summation of a multinomial coefficient times the product of the CDF, PDF, and SF raised to various powers (see Example 3.2).

3.1.2 Sampling Without Replacement

Providing the string "wo" as the optional fourth argument indicates that items are sampled from the discrete population without replacement. The algorithm first determines whether the random variable has finite or infinite support. The following two subsections explain the steps followed after this determination is made.

Finite Support

If the population distribution has finite support, again denote the population size by N . In order to specify the support of the order statistic in a compact form,

additionally assume (without loss of generality) that the population support is the ordered set $\Omega = \{1, 2, \dots, N\}$. For example, if

$$f(x) = \begin{cases} 0.5 & x = 7 \\ 0.2 & x = 11 \\ 0.3 & x = 15, \end{cases}$$

then $N = 3$, and the support is assumed to be $\{1, 2, 3\}$, instead of $\{7, 11, 15\}$.

If the population has equally likely PDF values, e.g., $f(x) = \frac{1}{6}$ for $x = 1, 2, \dots, 6$, then by combinatorial analysis (Wilks 1962, page 243):

$$f_{X_{(r)}}(x) = \frac{\binom{x-1}{r-1} \binom{N-x}{n-r}}{\binom{N}{n}} \quad x = r, r+1, \dots, r+N-n.$$

If X has finite support and non-equally likely PDF values, there are three cases for calculating the PDF of the r th order statistic:

1. If $n = 1$, i.e., only one item is sampled, then the PDF of the r th order statistic is the same as the population PDF.
2. If $n = N$, i.e., the entire population is sampled, then the PDF of the r th order statistic is

$$f_{X_{(r)}}(x) = \begin{cases} 1 & x = r \\ 0 & \text{otherwise.} \end{cases}$$

3. If $n = 2, 3, \dots, N-1$, then an n -by- N array, *ProbStorage*, is defined that eventually contains the PDF values for all order statistics, i.e., $f_{X_{(r)}}(x)$ for $x = 1, 2, \dots, N$ and $r = 1, 2, \dots, n$. The rows denote the r order statistic values and the columns denote the x values that the order statistic may assume. The array is initialized to contain all zeros. The algorithm's implementation requires the use of two combinatorial procedures, **NextCombination**

and `NextPermutation`, whose Maple codes are contained in Appendix B. The following steps are implemented.

- (a) The first lexicographical combination of n items sampled from the sequence of integers 1 to N is formed. This first combination is the unordered set $\{1, 2, \dots, n\}$.
- (b) Given a combination consisting of n distinct integers, the algorithm generates all corresponding permutations. It generates the first permutation by arranging the integers in increasing order. The probability for the permutation is calculated by substituting the permuted values into the PDF of X . Since the PDF has already been converted to a list of probabilities values in the first sublist of X , the probability of a permutation is computed by selecting the probability values in the first sublist that occupy the positions of the permuted values. For example, let $f(x) = \binom{4}{x-1} \frac{1}{16}$ for $x = 1, 2, \dots, 5$. Then the first two sublists of the PDF of X are (or have been converted to) $[\frac{1}{16}, \frac{1}{4}, \frac{3}{8}, \frac{1}{4}, \frac{1}{16}]$, $[1, 2, 3, 4, 5]$. Suppose the permutation that the algorithm is currently processing is $[5, 3, 2]$. Then the probability of this permutation is

$$f(5) \cdot \frac{f(3)}{1 - f(5)} \cdot \frac{f(2)}{1 - (f(5) + f(3))}$$

or

$$\frac{1}{16} \cdot \frac{\frac{3}{8}}{1 - \frac{1}{16}} \cdot \frac{\frac{1}{4}}{1 - (\frac{1}{16} + \frac{3}{8})} = \frac{1}{90}$$

- (c) After the permutation's probability is computed, the permutation is rearranged in increasing order. The permutation above, for example, becomes $[2, 3, 5]$. The algorithm then adds the computed probability value to the appropriate cells in the *ProbStorage* array. The (r, x) cell accumulates the

probabilities of the various ways that the r th order statistic takes on the value x . For example, the final value of the (1, 2) cell represents the probability that the first order statistic assumes the value two. In the example illustrated above, the *ProbStorage* array cells (1, 2), (2, 3), and (3, 5) are incremented by the permutation [5, 3, 2]'s probability $\frac{1}{90}$. See Figure 3.2.

$r \backslash x$	1	2	3	4	5
1	$\Sigma(1, 1)$	$\Sigma(1, 2) + \frac{1}{90}$	$\Sigma(1, 3)$	—	—
2	—	$\Sigma(2, 2)$	$\Sigma(2, 3) + \frac{1}{90}$	$\Sigma(2, 4)$	—
3	—	—	$\Sigma(3, 3)$	$\Sigma(3, 4)$	$\Sigma(3, 5) + \frac{1}{90}$

Figure 3.2: *ProbStorage* array for X with PDF $f(x) = \binom{4}{x-1} \frac{1}{16}$ for $x = 1, 2, \dots, 5$, in which $n = 3$ items are sampled without replacement. Given the permutation [5, 3, 2], its probability $\frac{1}{90}$ is added to the current probability sums $\Sigma(r, x)$ in cells [1, 2], [2, 3], and [3, 5]. Dashes denote impossible values.

- (d) After the appropriate cells are incremented, the next permutation in lexicographical order is found using the `NextPermutation` procedure. The permutation's probability is then computed and placed in the appropriate *ProbStorage* cells, as discussed above in steps (b) and (c).
- (e) After all $n!$ permutations of a given combination are exhausted, the procedure `NextCombination` finds the next lexicographical combination. Given this new combination, the algorithm repeats steps (b) through (d). This process iterates $\binom{N}{n}$ times, since this is the number of combinations of size n chosen from a population of size N .

Infinite Support

If X has infinite support, then the pattern established for finding the PDF of the r th order statistic in the finite support case does not work because the loops will

be endless. At this time, the `OrderStat` procedure cannot calculate the distribution of order statistics in the without replacement case when the discrete population has infinite support and $n \geq 3$ or $r \geq 3$. Future work with `OrderStat` will begin to incorporate the algorithmic pattern (already identified for $n = 2$ in the next paragraphs) for values of $n \geq 3$.

Assume that n items are sampled without replacement from the support $\Omega = \{1, 2, \dots\}$. The PDF of $X_{(1)}$ when $n = 1$ item is sampled without replacement from Ω is identical to the population PDF $f(x)$.

When $n = 2$, let X_1 be the first variate sampled and X_2 be the second variate sampled. Also, let $X_{(1)} = \min\{X_1, X_2\}$ and $X_{(2)} = \max\{X_1, X_2\}$. Figure 3.3 shows the support values associated with the joint distribution X_1 and X_2 along with the appropriate mass values to sum over in order to calculate $f_{X_{(1)}}(x) = \Pr(X_{(1)} = x)$ (indicated by the dashed rectangles of infinite length) and $f_{X_{(2)}}(x) = \Pr(X_{(2)} = x)$ (indicated by the solid rectangles). If the population PDF is given by $f(x)$, the marginal PDF of X_1 is

$$f_{X_{(1)}}(x_1) = f(x_1) \quad x_1 = 1, 2, \dots$$

The conditional PDF of X_2 given $X_1 = x_1$ is

$$f_{X_2|X_1}(x_2|x_1) = \frac{f(x_2)}{1 - f(x_1)} \quad x_1 = 1, 2, \dots; x_2 = 1, 2, \dots; x_1 \neq x_2.$$

Thus the joint distribution of X_1 and X_2 is

$$f_{X_1, X_2}(x_1, x_2) = f_{X_2|X_1}(x_2|x_1)f_{X_1}(x_1) = \frac{f(x_1)f(x_2)}{1 - f(x_1)} \quad x_1 = 1, 2, \dots; x_2 = 1, 2, \dots; x_1 \neq x_2.$$

Summing as indicated in Figure 3.3 yields the PDFs for $X_{(1)}$ and $X_{(2)}$ as

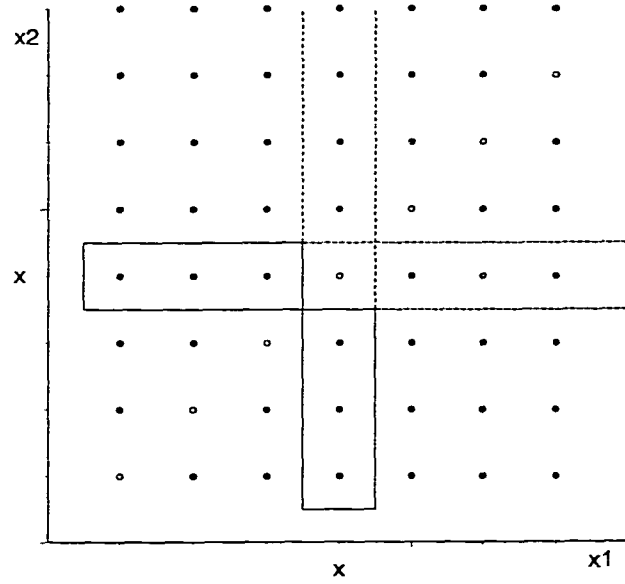


Figure 3.3: Support values associated with the joint distribution of X_1 and X_2 along with the appropriate mass values to sum over to calculate $f_{X_{(1)}}(x)$ (indicated by dashed rectangles of infinite length) and $f_{X_{(2)}}(x)$ (indicated by solid rectangles).

$$\begin{aligned}
 f_{X_{(1)}}(x) &= \Pr(X_{(1)} = x) \\
 &= \sum_{x_1=x+1}^{\infty} f_{X_1, X_2}(x_1, x) + \sum_{x_2=x+1}^{\infty} f_{X_1, X_2}(x, x_2) \\
 &= \sum_{x_1=x+1}^{\infty} \frac{f(x_1)}{1-f(x_1)} \cdot f(x) + \sum_{x_2=x+1}^{\infty} \frac{f(x)}{1-f(x)} \cdot f(x_2) \\
 &= f(x) \sum_{x_1=x+1}^{\infty} \frac{f(x_1)}{1-f(x_1)} + \frac{f(x)}{1-f(x)} \cdot S(x+1),
 \end{aligned}$$

and

$$\begin{aligned}
 f_{X_{(2)}}(x) &= \Pr(X_{(2)} = x) \\
 &= \sum_{x_1=1}^{x-1} f_{X_1, X_2}(x_1, x) + \sum_{x_2=1}^{x-1} f_{X_1, X_2}(x, x_2) \\
 &= \sum_{x_1=1}^{x-1} \frac{f(x_1)}{1-f(x_1)} \cdot f(x) + \sum_{x_2=1}^{x-1} \frac{f(x)}{1-f(x)} \cdot f(x_2) \\
 &= f(x) \sum_{x_1=1}^{x-1} \frac{f(x_1)}{1-f(x_1)} + \frac{f(x)}{1-f(x)} \cdot F(x-1).
 \end{aligned}$$

Thus, the PDF of $X_{(1)}$ and $X_{(2)}$ when $n = 2$ items are sampled without replacement from $\Omega = \{1, 2, \dots\}$ are

$$f_{X_{(1)}}(x) = f(x) \left[\frac{S(x+1)}{1-f(x)} + \sum_{x_1=x+1}^{\infty} \frac{f(x)}{1-f(x)} \right] \quad x_1 = 1, 2, \dots,$$

and

$$f_{X_{(2)}}(x) = f(x) \left[\frac{F(x-1)}{1-f(x)} + \sum_{x_1=1}^{x-1} \frac{f(x)}{1-f(x)} \right] \quad x_1 = 1, 2, \dots$$

3.2 Examples

Returning to the continuous exercises discussed in the introduction of this chapter, the `OrderStat` algorithm can be used to determine their solutions. In the first example, define X to be an `exponential(λ)` random variable (pre-defined in APPL). Then the `OrderStat` algorithm (for continuous random variables) is used to determine the PDF of its r th order statistic. The following statements

```
> X := ExponentialRV(lambda);
> OrderStat(X, n, r);
```

returns the PDF in its APPL list-of-sublists as

$$\left[\left[x \rightarrow \frac{n! \cdot \lambda e^{-\lambda x}}{(r-1)!(n-r)!} \cdot (1 - e^{-\lambda x})^{r-1} \cdot (e^{-\lambda x})^{n-r} \right], [0.. \infty], ["Continuous", "PDF"] \right].$$

For the second example, define X as a `beta($\frac{1}{2}, 2$)` random variable (also pre-defined in APPL) and Y as the PDF of the median order statistic $r = 3$ associated with a sample of size $n = 5$. The CDF for the median order statistic, Y , at the value $1/4$ is computed with the APPL statements

```
> X := BetaRV(1 / 2, 2);
> Y := OrderStat(X, 5, 3);
> CDF(Y, 1 / 4);
```

yielding the exact solution of $\frac{429913}{524288}$.

The following two subsections use the algorithm described in the implementation section to determine the PDF of r th order statistic for discrete populations. Examples for each branch of the tree in Figure 3.1 are provided to illustrate the algorithms used to generate the PDF of the order statistic. Table 3.1 displays the type of sampling and the example associated with it.

Table 3.1: Categorization of discrete order statistics with associated examples.

Sampling convention	PDF representation	Support type	Probability distribution	Examples
With replacement	Numeric PDF	—	—	3.3
	Formulaic PDF	—	—	3.4, 3.5
Without replacement	—	Finite	Equally likely	3.6
			Non-equally likely	3.7
		Infinite	—	3.8

3.2.1 Sampling With Replacement

Example 3.3. (Miller & Miller, 1999, page 296) Find the sampling distribution of $X_{(1)}$ for random samples of size $n = 2$ taken with replacement from the finite population that consists of the first five positive integers. (*Hint:* Enumerate all possibilities.)

Solution: Using the hint, there are twenty-five possible samples of size $n = 2$ when sampling with replacement from the first five positive integers. They are

(1, 1)	(2, 1)	(3, 1)	(4, 1)	(5, 1)
(1, 2)	(2, 2)	(3, 2)	(4, 2)	(5, 2)
(1, 3)	(2, 3)	(3, 3)	(4, 3)	(5, 3)
(1, 4)	(2, 4)	(3, 4)	(4, 4)	(5, 4)
(1, 5)	(2, 5)	(3, 5)	(4, 5)	(5, 5)

After examining the possibilities, it is not hard to determine that the PDF of $X_{(1)}$ is

$$f_{X_{(1)}}(x) = \begin{cases} \frac{9}{25} & x = 1 \\ \frac{7}{25} & x = 2 \\ \frac{1}{5} & x = 3 \\ \frac{3}{25} & x = 4 \\ \frac{1}{25} & x = 5 \end{cases}$$

This PDF is computed in APPL with the statements

```
> X := UniformDiscreteRV(1, 5);
> OrderStat(X, 2, 1);
```

Although examples of this type can be done by enumeration, the benefit of APPL is demonstrated for large populations and/or when sampling more than just $n = 2$ items. □

The next example illustrates a case in which a formulaic PDF representation of the population must be used since it has an infinite support.

Example 3.4. (*Sampling with replacement; Formulaic PDF*) Define a geometric random variable X with parameter p ($0 < p < 1$) to be the number of trials up to

and including the first success, i.e., $f_X(x) = p \cdot q^{x-1}$, where $q = 1 - p$ for $x = 1, 2, \dots$. Margolin and Winokur (1967, pages 924–925) have tabulated values for the mean and variance of the r th order statistic of a geometric distribution in a sample of size n for $n = 1, 5, 10, 15, 20$, $r = 1, 2, \dots, 5, 10, 15, 20$, (where $r \leq n$), and $p = 0.25, 0.5, 0.75$. (The table values provided by Margolin and Winokur are given to two decimal places.) If $X \sim \text{geometric}(\frac{1}{4})$, then determine the exact values of the mean and variance of the third order statistic when $n = 5$ items are sampled with replacement.

Solution: The formulas Margolin and Winokur (1967, page 921) use to compute the first and second moments of the r th order statistic of a geometric distribution (with parameter $p = 1 - q$) when n items are sampled are

$$E[X_{(r)}] = n \binom{n-1}{r-1} \sum_{j=0}^{r-1} \frac{(-1)^j \binom{r-1}{j}}{(n-r+j+1)(1-q^{n-r+j+1})}$$

and

$$E[X_{(r)}^2] = n \binom{n-1}{r-1} \sum_{j=0}^{r-1} \frac{(-1)^j \binom{r-1}{j} (1+q^{n-r+j+1})}{(n-r+j+1)(1-q^{n-r+j+1})^2}.$$

Using the `OrderStat` procedure, the exact value of any of the rounded figures given in their tables can be produced. If $X \sim \text{geometric}(\frac{1}{4})$, then the exact values of the mean and variance of the third ($r = 3$) order statistic when $n = 5$ items are sampled (with replacement) can be found with the statements

```
> X := GeometricRV(1 / 4);
> Y := OrderStat(X, 5, 3);
> Mean(Y);
> Variance(Y);
```

which yields $\frac{3257984}{1011395} \cong 3.22$ and $\frac{13665723739776}{5114599230125} \cong 2.67$, respectively. \square

The `OrderStat` procedure is able to accept a much larger range of arguments than just the numeric values for p , n , and r given in Example 3.4. Further, the procedures

can also accept random variables with symbolic parameters. If $Y \sim \text{geometric}(p)$, for example, then the variance of the minimum order statistic ($r = 1$) when $n = 6$ items are sampled is $\frac{1-6p+15p^2-20p^3+15p^4-6p^5+p^6}{p^2(p^5-6p^4+15p^3-20p^2+15p-6)}$, which is determined with the statement

```
> Variance(OrderStat(GeometricRV(p), 6, 1));
```

Other measures, such as the median of this distribution, can be found with the use of additional APPL procedures. The median of the maximum order statistic when $n = 15$ items are sampled with replacement from a geometric distribution with parameter $p = \frac{2}{5}$ can be found with the statements

```
> X := GeometricRV(2 / 5);
> Y := OrderStat(X, 15, 15);
> IDF(Y, 0.5);
```

which returns the median of this distribution as 7.

Example 3.5. (*Sampling with replacement; Formulaic PDF*) Let $X \sim \text{Poisson}(\lambda)$. Draw $n = 3$ items with replacement from this population and determine the PDF of the largest order statistic.

Solution: In order to find a formula for the PDF of $X_{(3)}$, where $f(x) = \frac{\lambda^x e^{-\lambda}}{x!}$ for $x = 0, 1, 2, \dots$, simplify the following general expressions (in terms of x) to obtain $f_{X_{(3)}}(0) = [f(0)]^3 = e^{-3\lambda}$ and

$$\begin{aligned}
 f_{X_{(3)}}(x) &= \sum_{u=0}^2 \binom{3}{u} [F(x-1)]^u [f(x)]^{3-u} \\
 &= [f(x)]^3 + 3[F(x-1)][f(x)]^2 + 3[F(x-1)]^2[f(x)] \\
 &= \frac{\lambda^{3x} e^{-3\lambda}}{(x!)^3} + 3 \left[\sum_{y=0}^{x-1} \frac{\lambda^y e^{-\lambda}}{y!} \right] \frac{\lambda^{2x} e^{-2\lambda}}{(x!)^2} + 3 \left[\sum_{y=0}^{x-1} \frac{\lambda^y e^{-\lambda}}{y!} \right]^2 \frac{\lambda^x e^{-\lambda}}{x!} \\
 &= \frac{\lambda^{3x} e^{-3\lambda}}{(x!)^3} + 3 \cdot \frac{\Gamma(x, \lambda)}{\Gamma(x)} \cdot \frac{\lambda^{2x} e^{-2\lambda}}{(x!)^2} + 3 \cdot \frac{(\Gamma(x, \lambda))^2}{(\Gamma(x))^2} \cdot \frac{\lambda^x e^{-\lambda}}{x!} \\
 &= \frac{e^{-\lambda} (\lambda^{3x} e^{-2\lambda} + 3\lambda^{2x} e^{-\lambda} \Gamma(x, \lambda)x + 3\lambda^x (\Gamma(x, \lambda))^2 x^2)}{(x!)^3} \quad x = 1, 2, \dots,
 \end{aligned}$$

where $\Gamma(a, z) = \int_z^\infty e^{-t} t^{a-1} dt$ is the incomplete gamma function. The APPL statements

```
> X := PoissonRV(lambda);
> OrderStat(X, 3, 3);
```

yield the above PDF as a single function

$$f_{X_{(3)}}(x) = \frac{e^{-\lambda} (3\lambda^x (\Gamma(x, \lambda))^2 x^2 + 3\lambda^{2x} e^{-\lambda} \Gamma(x, \lambda)x + \lambda^{3x} e^{-2\lambda})}{(x!)^3}$$

for $x = 0, 1, \dots$. APPL procedures, such as `OrderStat`, are written so as to convert expressions involving gamma functions and incomplete gamma functions to their simplest form, which may include rewriting gamma terms as factorials, whenever possible. Especially in the case of problems with Poisson random variables, the user should see the PDF in its well-known form $f(x) = \frac{\lambda^x e^{-\lambda}}{x!}$, rather than $f(x) = \frac{\lambda^x e^{-\lambda}}{\Gamma(x+1)}$ for $x = 0, 1, \dots$. \square

3.2.2 Sampling Without Replacement

Example 3.6. (*Sampling without replacement; Finite support; Equally likely distribution*) (Hogg & Craig, 1995, page 231) Draw 15 cards at random and without replacement from a deck of 25 cards numbered 1, 2, ..., 25. Find the probability that the card numbered 10 is the median of the cards selected.

Solution: Let $f(x) = \frac{1}{25}$ for $x = 1, 2, \dots, 25$. The population size is $N = 25$, the size of the sample drawn is $n = 15$, and the order statistic being considered is $r = 8$. To calculate the probability that the median order statistic is $x = 10$, compute

$$\begin{aligned} \Pr(X_{(8)} = 10) &= \frac{\binom{9}{7} \binom{15}{7}}{\binom{25}{15}} \\ &= \frac{1053}{14858} \\ &\cong 0.0709, \end{aligned}$$

since X has finite support and equally likely probability values.

The APPL statements to solve this problem are

```
> X := UniformDiscreteRV(1, 25);
> Y := OrderStat(X, 15, 8, "wo"); # sampling without ("wo") replacement
> PDF(Y, 10);
```

which yield the exact solution $\frac{1053}{14858}$. □

Of greater significance, the random variable Y computed in Example 3.6 contains more than just the PDF value at $y = 10$ —it contains the distribution of the median, ranging from 8 to 18. The PDF of Y as returned in Example 3.6 is

$$Y := \left[\left[\frac{13}{2185}, \frac{208}{7429}, \frac{1053}{14858}, \frac{936}{7429}, \frac{1287}{7429}, \frac{7128}{37145}, \frac{1287}{7429}, \frac{936}{7429}, \frac{1053}{14858}, \frac{208}{7429}, \frac{13}{2185} \right], \right. \\ \left. [8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18], ["Discrete", "PDF"] \right].$$

Furthermore, APPL can return the formulaic PDF for the r th order statistic of Example 3.6 when $n = 15$ items are drawn from the population without replacement.

The statements

```
> X := UniformDiscreteRV(1, 25);
> Y := OrderStat(X, 15, r, "wo");
```

produce the correct PDF

$$Y := \left[\left[\frac{1}{3268760} \binom{x-1}{r-1} \binom{25-x}{15-r} \right], [x = r .. 10 + r], ["Discrete", "PDF"] \right].$$

Example 3.7. (*Sampling without replacement; Finite support; Non-equally likely distribution*) Let X be a random variable with PDF

$$f(x) = \begin{cases} p_1 & x = 1 \\ p_2 & x = 10 \\ p_3 & x = 100 \\ p_4 & x = 1000, \end{cases}$$

where $p_1 + p_2 + p_3 + p_4 = 1$ and $p_i > 0, i = 1, 2, 3, 4$. Find the distribution of the median order statistic $r = 2$ when $n = 3$ items are sampled without replacement from the population.

Solution: The PDF for the median order statistic $r = 2$ when $n = 3$ items are sampled is

$$f_{X_{(2)}}(x) = \begin{cases} \frac{p_4 p_2 p_1}{(1-p_4)(1-p_2-p_4)} + \frac{p_4 p_2 p_1}{(1-p_4)(1-p_1-p_4)} + \frac{p_4 p_2 p_1}{(1-p_2)(1-p_2-p_4)} + \frac{p_4 p_2 p_1}{(1-p_2)(1-p_1-p_2)} + \\ \frac{p_4 p_2 p_1}{(1-p_1)(1-p_1-p_4)} + \frac{p_4 p_2 p_1}{(1-p_1)(1-p_1-p_2)} + \frac{p_3 p_2 p_1}{(1-p_3)(1-p_2-p_3)} + \frac{p_3 p_2 p_1}{(1-p_3)(1-p_1-p_3)} + \\ \frac{p_3 p_2 p_1}{(1-p_2)(1-p_2-p_3)} + \frac{p_3 p_2 p_1}{(1-p_2)(1-p_1-p_2)} + \frac{p_3 p_2 p_1}{(1-p_1)(1-p_1-p_3)} + \frac{p_3 p_2 p_1}{(1-p_1)(1-p_1-p_2)} & x = 10 \\ \frac{p_4 p_3 p_2}{(1-p_4)(1-p_4-p_3)} + \frac{p_4 p_3 p_2}{(1-p_4)(1-p_2-p_4)} + \frac{p_4 p_3 p_2}{(1-p_3)(1-p_4-p_3)} + \frac{p_4 p_3 p_2}{(1-p_3)(1-p_2-p_3)} + \\ \frac{p_4 p_3 p_2}{(1-p_2)(1-p_2-p_4)} + \frac{p_4 p_3 p_2}{(1-p_2)(1-p_2-p_3)} + \frac{p_4 p_3 p_1}{(1-p_4)(1-p_4-p_3)} + \frac{p_4 p_3 p_1}{(1-p_4)(1-p_1-p_4)} + \\ \frac{p_4 p_3 p_1}{(1-p_3)(1-p_4-p_3)} + \frac{p_4 p_3 p_1}{(1-p_3)(1-p_1-p_3)} + \frac{p_4 p_3 p_1}{(1-p_1)(1-p_1-p_4)} + \frac{p_4 p_3 p_1}{(1-p_1)(1-p_1-p_3)} & x = 100, \end{cases}$$

which is found in APPL using the statements

```
> X := [[p1, p2, p3, p4], [1, 10, 100, 1000], ["Discrete", "PDF"]];
> OrderStat(X, 3, 2, "wo");
```

□

Example 3.8. (*Sampling without replacement; Infinite support*) Let $X \sim \text{geometric}(\frac{1}{2})$.

Find the probability that the maximum order statistic $r = 2$ is five when $n = 2$ items are drawn without replacement from the population.

Solution: Let $X \sim \text{geometric}(p)$ parameterized as

$$f(x) = p(1-p)^{x-1} \quad x = 1, 2, \dots$$

The SF is

$$\begin{aligned}
 S(x) &= \Pr(X \geq x) \\
 &= \sum_{w=x}^{\infty} f(w) \\
 &= \sum_{w=x}^{\infty} p(1-p)^{w-1} \\
 &= (1-p)^{x-1} \quad x = 1, 2, \dots
 \end{aligned}$$

by summing the geometric series. Also, since $F(x) + S(x) - f(x) = 1$ for $x = 1, 2, \dots$, the CDF is

$$\begin{aligned}
 F(x) &= 1 - (1-p)^{x-1} + p(1-p)^{x-1} \\
 &= 1 - (1-p)^x \quad x = 1, 2, \dots
 \end{aligned}$$

Thus, the PDFs of $X_{(1)}$ and $X_{(2)}$ when $n = 2$ items are sampled without replacement are

$$\begin{aligned}
 f_{X_{(1)}}(x) &= p(1-p)^{x-1} \sum_{x_1=x+1}^{\infty} \frac{p(1-p)^{x_1-1}}{1-p(1-p)^{x_1-1}} + \frac{p(1-p)^{x-1}}{1-p(1-p)^{x-1}} \cdot (1-p)^x \\
 &= p^2(1-p)^{x-1} \sum_{x_1=x+1}^{\infty} \frac{(1-p)^{x_1-1}}{1-p(1-p)^{x_1-1}} + \frac{p(1-p)^{2x-1}}{1-p(1-p)^{x-1}}
 \end{aligned}$$

for $x = 1, 2, \dots$, and

$$\begin{aligned}
 f_{X_{(2)}}(x) &= p(1-p)^{x-1} \sum_{x_1=1}^{x-1} \frac{p(1-p)^{x_1-1}}{1-p(1-p)^{x_1-1}} + \frac{p(1-p)^{x-1}}{1-p(1-p)^{x-1}} \cdot [1 - (1-p)^{x-1}] \\
 &= p^2(1-p)^{x-1} \sum_{x_1=1}^{x-1} \frac{(1-p)^{x_1-1}}{1-p(1-p)^{x_1-1}} + \frac{p(1-p)^{x-1} - p(1-p)^{2x-2}}{1-p(1-p)^{x-1}}.
 \end{aligned}$$

As a result, the PDF of the maximum order statistic $X_{(2)}$ for $p = 1/2$ is

$$f_{X_{(2)}}(x) = \left(\frac{1}{2}\right)^{x+1} \sum_{x_1=1}^{x-1} \frac{\left(\frac{1}{2}\right)^{x_1-1}}{1 - \left(\frac{1}{2}\right)^{x_1}} + \frac{\left(\frac{1}{2}\right)^x - \left(\frac{1}{2}\right)^{2x-1}}{1 - \left(\frac{1}{2}\right)^x} \quad x = 1, 2, \dots,$$

and so

$$\begin{aligned} f_{X_{(2)}}(5) &= \left(\frac{1}{2}\right)^6 \sum_{x_1=1}^4 \frac{\left(\frac{1}{2}\right)^{x_1-1}}{1 - \left(\frac{1}{2}\right)^{x_1}} + \frac{\left(\frac{1}{2}\right)^5 - \left(\frac{1}{2}\right)^9}{1 - \left(\frac{1}{2}\right)^5} \\ &= \frac{681}{8680}. \end{aligned}$$

This probability value is computed in APPL with the statements

```
> X := GeometricRV(1 / 2);
> Y := OrderStat(X, 2, 2, "wo");
> PDF(Y, 5);
```

□

3.3 Range Statistics

One natural extension of the `OrderStat` procedure is `RangeStat`, a procedure which determines the PDF of the range of a sample of n items drawn from a discrete population, either with or without replacement. Since `RangeStat` had not been coded in APPL for continuous distributions, the procedure was also extended to cover these distributions for use in Section 3.4, the bootstrap section.

3.3.1 Discrete Distributions

Let X be a discrete random variable with PDF $f(x)$ and support $x_1, x_2, \dots, x_N \in \mathbf{Z}^+$, where $x_1 < x_2 < \dots < x_N$. Also, let $f(x_i) = p_i$ for $i = 1, 2, \dots, N$. Suppose n items are sampled with replacement (which implies that the n draws are independent) from the discrete population. The probability that the maximum value drawn is x_j and

the minimum value drawn is x_i for $i \leq j$, or that the range is $|x_j - x_i|$, for those n draws is computed using the formula (Stockmeyer, 2001):

$$\Pr(X_{(n)} = x_j, X_{(1)} = x_i) = \sum_{k=i}^j (p_k)^n - \sum_{k=i+1}^j (p_k)^n - \sum_{k=i}^{j-1} (p_k)^n + \sum_{k=i+1}^{j-1} (p_k)^n$$

for $i = 1, 2, \dots, N$, $j = i, i + 1, \dots, N$. The term $\sum_{k=i}^j (p_k)^n$ is the probability that all sampled items lie between x_i and x_j inclusive. The term $\sum_{k=i+1}^j (p_k)^n$ removes the probability that the sampled items do not include x_i , since this would result in a range which is less than $|x_j - x_i|$. The term $\sum_{k=i}^{j-1} (p_k)^n$ removes the probability that the sampled items do not include x_j , since this would also result in a range which is also less than $|x_j - x_i|$. The term $\sum_{k=i+1}^{j-1} (p_k)^n$ adds back in the probabilities that were removed twice (by the second and third terms)—specifically, the probabilities of obtaining samples that included neither x_i nor x_j .

To demonstrate how this formula is used in the `RangeStat` procedure, let X be a discrete random variable with PDF

$$f(x) = \begin{cases} 0.5 & x = 1 \\ 0.3 & x = 5 \\ 0.2 & x = 9. \end{cases}$$

If $n = 3$ values are sampled (with replacement) from the population, then the 3^3 possible outcomes can be written as the ten ordered sets $\{1, 1, 1\}$, $\{5, 5, 5\}$, $\{9, 9, 9\}$, $\{1, 1, 5\}$, $\{1, 1, 9\}$, $\{1, 5, 5\}$, $\{1, 9, 9\}$, $\{1, 5, 9\}$, $\{5, 5, 9\}$, and $\{5, 9, 9\}$. The possible range values are 0, 4, and 8. The range statistic algorithm does the following:

1. Two arrays, RS and RP , are built, where RS contains the range support values and RP contains the corresponding range probability mass values. The arrays are initialized to contain all zeros. Given that the support is $x_1 < x_2 < \dots < x_N$,

there are $\sum_{i=1}^{N-1} i = \frac{N(N-1)}{2}$ possible non-zero range support values, some of which may be identical, and one support value of zero. Thus, the arrays are initialized to be of size $\frac{N(N-1)}{2} + 1$. For the example random variable X , the initial RS and RP arrays are

$$RS: \begin{array}{|c|c|c|c|} \hline 0 & 0 & 0 & 0 \\ \hline \end{array}$$

$$RP: \begin{array}{|c|c|c|c|} \hline 0 & 0 & 0 & 0 \\ \hline \end{array}$$

2. Fill the first element of RP with $\sum_{i=1}^N f(x_i)^N$, the probability that the range is zero. For the example random variable X , the arrays are

$$RS: \begin{array}{|c|c|c|c|} \hline 0 & 0 & 0 & 0 \\ \hline \end{array}$$

$$RP: \begin{array}{|c|c|c|c|} \hline 0.16 & 0 & 0 & 0 \\ \hline \end{array}$$

since $(0.5)^3 + (0.3)^3 + (0.2)^3 = 0.16$.

3. For $i = 1, 2, \dots, N$, $j = i+1, i+2, \dots, N$, the difference $x_j - x_i$ is computed and placed in RS array cell k , where $k = 2, 3, \dots, \frac{N(N-1)}{2} + 1$. The corresponding probability, $\Pr(X_{(n)} = j, X_{(1)} = i)$, is computed by the formula discussed on the previous page and placed in the RP array cell k . The updated RS and RP arrays in the example are

$$RS: \begin{array}{|c|c|c|c|} \hline 0 & 4 & 8 & 4 \\ \hline \end{array}$$

$$RP: \begin{array}{|c|c|c|c|} \hline 0.16 & 0.36 & 0.39 & 0.09 \\ \hline \end{array}$$

4. The APPL InsertionSort procedure uses an insertion sort algorithm to sort the array RS and to make the appropriate updates to the RP array. The RS and RP arrays after sorting for the example are

RS :

0	4	4	8
---	---	---	---

RP :

0.16	0.36	0.09	0.39
------	------	------	------

5. The identical values in the range support array are combined and the appropriate updates are made to the probability mass value array. Zeros at the end of the RS array reflect the number of identical range values that are combined. For the example, since two range values of four are combined, one array cell containing zero appears at the end of the RS (and RP) array.

RS :

0	4	8	0
---	---	---	---

RP :

0.16	0.45	0.39	0
------	------	------	---

6. The extraneous zeros are removed from the range support and probability mass value arrays. The zeros (if any) at the end of the RP array are not probability values. The zeros exist because of the redundant values in RS . Use the RP array to determine how many extraneous zeros exist, and reconstruct the RS and RP arrays so the extra zeros are not included as part of the arrays. The final RS and RP arrays for the example are

RS :

0	4	8
---	---	---

RP :

0.16	0.45	0.39
------	------	------

Example 3.9. When giving a multiple choice pre-calculus test, I have noticed that the probability that a student earns a certain score X on the test has the following PDF:

$$f_X(x) = \begin{cases} 0.01 & x = 40 \\ 0.06 & x = 50 \\ 0.12 & x = 60 \\ 0.28 & x = 70 \\ 0.37 & x = 80 \\ 0.14 & x = 90 \\ 0.02 & x = 100. \end{cases}$$

After scoring a test, I give students summary statistics about how they did as a class on the test. One of these statistics includes reporting the range score to them, i.e., the difference between the highest and lowest test scores. At The Ohio State University, I had 200 students taking a particular test, while at Virginia Wesleyan College I had only 20 students who would take that same test. Ohio State students complained more than Virginia Wesleyan students about the test being too hard after looking at the range score. Why is this so?

Solution: APPL can be used to quantify this phenomena. Let X be the discrete random variable representing the typical scores of students on these tests. Assign Y as the range statistic of the scores at Ohio State, and assign Z as the range statistic of the scores at Virginia Wesleyan. The statements

```
> X := [[0.01, 0.06, 0.12, 0.28, 0.37, 0.14, 0.02];
        [40, 50, 60, 70, 80, 90, 100], ["Discrete", "PDF"]];
> Y := RangeStat(X, 200);
> Z := RangeStat(X, 20);
```


produce the rounded PDFs for the ranges as

$$f_Y(y) = \begin{cases} 0.4369 \cdot 10^{-86} & y = 0 \\ 0.3825 \cdot 10^{-37} & y = 10 \\ 0.3373 \cdot 10^{-20} & y = 20 \\ 0.6431 \cdot 10^{-8} & y = 30 \\ 0.2262 \cdot 10^{-2} & y = 40 \\ 0.1470 & y = 50 \\ 0.8507 & y = 60, \end{cases}$$

and

$$f_Z(z) = \begin{cases} 0.2321 \cdot 10^{-8} & z = 0 \\ 0.1827 \cdot 10^{-3} & z = 10 \\ 0.0140 \cdot 10^{-1} & z = 20 \\ 0.1620 & z = 30 \\ 0.4567 & z = 40 \\ 0.3088 & z = 50 \\ 0.0583 & z = 60. \end{cases}$$

As can be seen from the PDF of Y , there is more than a 99% chance at Ohio State that a range of 50 or more is reported on a test. In fact, with a probability of approximately 0.85, the range is 60. The high range score leads students to focus on the fact that very low scores, e.g., 40 and 50, are obtained and the test must be too hard. On the other hand, at Virginia Wesleyan the range is 40 approximately 45% of the time. In students' minds, a range of 40 means there is not a large discrepancy between high and low scores (as in the Ohio State classes). A large range indicates a large variation in grades. Although this can be a good for some students because it

suggests there are those who scored on the high end, most students focus on the fact that a high range indicates that low scores are obtained. \square

If the `RangeStat` procedure is given three arguments, X , n , and "wo", then it returns the PDF of the range of the discrete random variable X when n items are sampled from the random variable's population *without* replacement. In the without replacement case, assume, without loss of generality, that the support of X is $\Omega = \{1, 2, \dots, N\}$, where $N \in \mathbf{Z}^+$.

In the without replacement case, the `RangeStat` algorithm basically follows the same steps as the `OrderStat` algorithm, including the use of the `NextCombination` and `NextPermutation` procedures. If $n = N$, i.e., the entire population is sampled, then the PDF of the range, which is called Y here, is

$$f(y) = \begin{cases} 1 & y = N - 1 \\ 0 & \text{otherwise.} \end{cases}$$

If $n = 2, 3, \dots, N - 1$ in the without replacement case, then a single-dimensional array of length $N - n + 1$ is defined to contain the PDF range values $y = n - 1, n, \dots, N - 1$. As in the `OrderStat` procedure, the first lexicographical combination of n items sampled from the sequence of integers 1 to N is formed. Given a combination, the algorithm generates all corresponding permutations. The probability for each permutation is calculated (as described in the Section 3.1.2), and then the maximum and minimum values of that permutation is determined. The permutation's range, i.e., the difference of its maximum and minimum values, is computed and then the appropriate array cell is incremented by that permutation's probability.

For example, let $f(x) = \binom{4}{x-1} \frac{1}{16}$ for $x = 1, 2, \dots, 5$. Suppose the permutation that the algorithm is currently processing is $[5, 3, 2]$. Then the probability of this

permutation is $\frac{1}{90}$ (as computed in Section 3.1.2). The maximum and minimum values of the permutation are 5 and 2, respectively, which means the range is 3. The PDF range cell $y = 3$ is incremented by $\frac{1}{90}$.

To demonstrate this algorithm, let X be the discrete random variable with PDF

$$f(x) = \begin{cases} \frac{1}{6} & x = 1 \\ \frac{1}{3} & x = 2 \\ \frac{1}{2} & x = 3. \end{cases}$$

If $n = 2$ values are sampled (without replacement) from the population, then the possible outcomes are the ordered sets $\{1, 2\}$, $\{1, 3\}$, $\{2, 1\}$, $\{2, 3\}$, $\{3, 1\}$, and $\{3, 2\}$. The possible range values are one and two. To calculate the PDF of the range, the six permutations and their probabilities are computed and added to the appropriate range array cells. The array cell that represents the probability that the range is one is computed as

$$\begin{aligned} \Pr(Y = 1) &= \Pr(X_1 = 1, X_2 = 2) + \Pr(X_1 = 2, X_2 = 1) + \Pr(X_1 = 2, X_2 = 3) + \Pr(X_1 = 3, X_2 = 2) \\ &= \frac{1}{15} + \frac{1}{12} + \frac{1}{4} + \frac{1}{3} \\ &= \frac{11}{15}. \end{aligned}$$

Thus, array cell one holds the value $\frac{11}{15}$, while array cell two holds the value $\frac{4}{15}$, which is computed in a similar manner. The APPL statements

```
> X := [[1 / 6, 1 / 3, 1 / 2], [1, 2, 3], ["Discrete", "PDF"]];
> RangeStat(X, 2, "wo");
```

produce the range PDF

$$f(y) = \begin{cases} \frac{11}{15} & y = 1 \\ \frac{4}{15} & y = 2. \end{cases}$$

3.3.2 Continuous Distributions

Let X be a continuous random variable with support Ω and PDF $f_X(x)$ and CDF $F_X(x)$. For $n \geq 2$, the joint PDF of $X_{(1)}$ and $X_{(n)}$ is (Hogg & Craig, 1995, pages 199–200)

$$f_{X_{(1)}, X_{(n)}}(x_{(1)}, x_{(n)}) = n \cdot (n - 1) [F_X(x_{(n)}) - F_X(x_{(1)})]^{n-2} \cdot f_X(x_{(1)}) \cdot f_X(x_{(n)})$$

for $\min\{\Omega\} < x_{(1)} < x_{(n)} < \max\{\Omega\}$.

The goal is to determine the PDF of the range $X_{(n)} - X_{(1)}$. Using the transformation technique, let $Y_1 = X_{(n)} - X_{(1)}$ and define the dummy transformation $Y_2 = X_{(n)}$. Consider the one-to-one transformations $y_1 = x_{(n)} - x_{(1)}$ and $y_2 = x_{(n)}$, and their inverses $x_{(1)} = y_2 - y_1$ and $x_{(n)} = y_2$, so that the corresponding Jacobian of the inverse transformation is

$$J = \begin{vmatrix} \frac{\partial x_{(1)}}{\partial y_1} & \frac{\partial x_{(1)}}{\partial y_2} \\ \frac{\partial x_{(n)}}{\partial y_1} & \frac{\partial x_{(n)}}{\partial y_2} \end{vmatrix} = \begin{vmatrix} -1 & 1 \\ 0 & 1 \end{vmatrix} = -1.$$

The joint PDF of Y_1 and Y_2 is

$$f_{Y_1, Y_2}(y_1, y_2) = |-1| \cdot n \cdot (n - 1) [F_X(y_2) - F_X(y_2 - y_1)]^{n-2} \cdot f_X(y_2 - y_1) \cdot f_X(y_2)$$

for $\min\{\Omega\} < y_1 < y_2 < \max\{\Omega\}$. Hence, the PDF of the range $Y_1 = X_{(n)} - X_{(1)}$ is

$$f_{Y_1}(y_1) = \int_{y_1}^{\max\{\Omega\}} n \cdot (n - 1) [F_X(y_2) - F_X(y_2 - y_1)]^{n-2} \cdot f_X(y_2 - y_1) \cdot f_X(y_2) dy_2$$

for $0 < y_1 < \max\{\Omega\}$.

Example 3.10. (Parzen, 1960, page 328) Find the probability that in a random sample of size n of a random variable uniformly distributed on the interval $[0, 1]$ the range will exceed 0.8.

Solution: Assume that $n \geq 2$ since a range is being determined and at least two values are necessary. Thus, the following APPL statements

```
> assume(n >= 2);
> X := UniformRV(0, 1);
> Y := RangeStat(X, n);
> SF(Y, 0.8);
```

determine the probability as $1 - n(0.8)^{n-1} + (n - 1)(0.8)^n$. \square

Example 3.11. (Bain & Engelhart, 1992, pages 219–220) Consider a random sample of size $n = 2$ from a distribution with PDF $f(x) = 2x$ for $0 < x < 1$. Find the PDF of the range.

Solution: Define X to be the random variable of interest in APPL. The statements

```
> X := [[x -> 2 * x], [0, 1], ["Continuous", "PDF"]];
> Y := RangeStat(X, 2);
```

produce the PDF of the range Y , which is

$$f(y) = \frac{8}{3} + \frac{4y^3}{3} - 4y \quad 0 < y < 1. \quad \square$$

3.4 Eliminating Resampling Error in Bootstrapping

Bootstrapping procedures require that B bootstrap samples be generated in order to perform some statistical inference concerning a data set. Although the requirements for the magnitude of B are typically modest, a practitioner would prefer to avoid the resampling error introduced by choosing a finite B , if possible. This section shows how APPL can be used to perform exact bootstrapping analysis in certain applications, eliminating the need for resampling in the analysis of a data set.

3.4.1 Introduction

Using Efron and Tibshirani's (1993) notation, consider the elimination of the generation of B bootstrap samples when performing a bootstrap analysis by calculating the exact distribution of the statistic of interest. There are several reasons for considering this approach:

- A bootstrapping novice can easily confuse the sample size n and number of bootstrap samples B . Eliminating the resampling of the data set B times simplifies the conceptualization of the bootstrapping process.
- In many situations, computer time is saved using the exact approach.
- A practitioner does not need to be concerned about problem-specific requirements for B , e.g., " B in the range of 50 to 200 usually makes $\hat{s}e_{boot}$ a good standard error estimator, even for estimators like the median" (Efron and Tibshirani, 1993, page 14) or " B should be ≥ 500 or 1000 in order to make the variability of acceptably low" for estimating 95th percentiles (Efron and Tibshirani, 1993, page 275).
- Exact values are always preferred to approximations. One should not add resampling error to sampling error unnecessarily.

By way of example, this section shows how APPL can be used to perform exact bootstrap analysis. The use of APPL eliminates the resampling variability that is present in a bootstrap procedure. The application area that is presented here is the estimation of standard errors.

3.4.2 Estimation of Standard Errors

The standard error of the sample mean, s/\sqrt{n} , is useful when comparing means, but standard errors for comparing other quantities (e.g., fractiles) are often intractable. This section considers the estimation of standard errors associated with the rat survival data given in Table 3.2 (Efron and Tibshirani, 1993, page 11). Seven rats are given a treatment and their survival times, given in days, are shown in the first row of the table. Nine other rats constitute a control group, and their survival times are shown in the second row of the table.

Table 3.2: Rat survival data.

Group	Data	n	Median	Mean	Range
Treatment	16, 23, 38, 94, 99, 141, 197	7	94	86.86	181
Control	10, 27, 30, 40, 46, 51, 52, 104, 146	9	46	56.22	136

Example 3.12. (Comparing medians.) Consider first the estimation of the standard error of the difference between the medians of the two samples. The standard bootstrapping approach to estimating the standard error of the median for the treatment group is to generate B bootstrap samples, each of which consists of seven samples drawn with replacement from 16, 23, 38, 94, 99, 141, and 197. The sample standard deviation of the medians of these B bootstrap samples is an estimate of the standard error of the median. Using the Splus commands

```
set.seed(1)
x <- c(16, 23, 38, 94, 99, 141, 197)
medn <- function(x){quantile(x, 0.50)}
bootstrap(x, medn, B = 50)
```

yields an estimated standard error of 41.18 for the treatment data with $B = 50$ bootstrap replicates. With the `set.seed` function used to call a stream number

corresponding to the associated column, Table 3.3 shows the estimated standard errors for several B values, where the $B = +\infty$ column will be calculated subsequently.

Table 3.3: Bootstrap estimates of the standard error of the median.

	$B = 50$	$B = 100$	$B = 250$	$B = 500$	$B = 1000$	$B = +\infty$
Treatment	41.18	37.63	36.88	37.90	38.98	37.83
Control	20.30	12.68	9.538	13.10	13.82	13.08

There is considerable resampling error introduced for smaller values of B . The $B = +\infty$ column of Table 3.3 corresponds to the *ideal bootstrap estimate of the standard error* of $\hat{\theta}$, or $se_{\hat{F}}(\hat{\theta}^*) = \lim_{B \rightarrow +\infty} \hat{se}_B$, to use the terminology and notation in Efron and Tibshirani (1993, page 46).

The APPL statements below eliminate the resampling error (i.e., $B = +\infty$):

```
> treatment := [16, 23, 38, 94, 99, 141, 197];
> X := BootstrapRV(treatment);
> Y := OrderStat(X, 7, 4);
> sqrt(Variance(Y));
```

The `BootstrapRV` procedure creates a discrete random variable X that can assume the values 16, 23, 38, 94, 99, 141, and 197, each with probability $\frac{1}{7}$. The call `OrderStat(X, 7, 4)` determines the distribution of the fourth order statistic in seven draws with replacement from the population associated with X , i.e., the distribution of the median. This call returns the distribution of the random variable Y as

$$f(y) = \begin{cases} \frac{8359}{823543} & y = 16 \\ \frac{80809}{823543} & y = 23 \\ \frac{196519}{823543} & y = 38 \\ \frac{252169}{823543} & y = 94 \\ \frac{196519}{823543} & y = 99 \\ \frac{80809}{823543} & y = 141 \\ \frac{8359}{823543} & y = 197. \end{cases}$$

Finally, `sqrt(Variance(Y))` returns the standard error as $\frac{2}{823543}\sqrt{242712738519382}$ which can be approximated using Maple's `evalf` procedure as 37.83467. In a similar fashion, the *ideal bootstrap estimate of the standard error* of the median can be calculated in the control case with the APPL statements

```
> control := [10, 27, 30, 40, 46, 51, 52, 104, 146]
> X := BootstrapRV(control);
> Y := OrderStat(X, 9, 5);
> sqrt(Variance(Y));
```

which yields $\frac{1}{387420489}\sqrt{25662937134123797402} \cong 13.07587$. Finally, although the difference between the two sample medians ($94 - 46 = 48$) seems large, it is only $48/\sqrt{37.83^2 + 13.08^2} \cong 1.19$ standard deviation units away from zero, indicating that the observed difference in the medians is not statistically significant. Had the standard bootstrapping procedure been applied with $B = 50$ bootstrap replications, Table 3.3 indicates that the number of standard deviation units would have been estimated to be $48/\sqrt{41.18^2 + 20.30^2} \cong 1.05$. Although the conclusion in this case is the same, the difference between using $B = 50$ and $B = +\infty$ could result in different conclusions for the same data set. \square

Example 3.13. (Comparing means.) Although the standard error of the mean can be expressed in closed-form, the previous analysis and attempt to compare the sample means to illustrate how to adapt APPL for comparing means is continued. `Splus` can be used to create bootstrap estimates given in Table 3.4 with the commands

```
set.seed(1)
x <- c(16, 23, 38, 94, 99, 141, 197)
bootstrap(x, mean, B = 50)
```

producing the upper-left-hand entry.

The APPL statements required to produce the $B = +\infty$ column associated with the treatment case of Table 3.4 are

Table 3.4: Bootstrap estimates of the standard error of the mean.

	$B = 50$	$B = 100$	$B = 250$	$B = 500$	$B = 1000$	$B = +\infty$
Treatment	23.89	24.29	23.16	24.36	23.75	23.36
Control	17.07	13.83	13.40	13.13	13.55	13.35

```

> n := 7;
> data := [16, 23, 38, 94, 99, 141, 197];
> X := BootstrapRV(data);
> Y := ConvolutionIID(X, n);
> Y := Transform(Y, [[x -> x / n], [-infinity, infinity]]);
> sqrt(Variance(Y));

```

which yield the PDF of the mean Y as

$$f(y) = \begin{cases} 1/7^7 = 1/823543 & y = 16 \\ \binom{7}{1}/7^7 = 1/117649 & y = 17 \\ \binom{7}{2}/7^7 = 3/117649 & y = 18 \\ \binom{7}{3}/7^7 = 5/117649 & y = 19 \\ \binom{7}{4}/7^7 = 1/117649 & y = 134/7 \\ \binom{7}{4}/7^7 = 5/117649 & y = 20 \\ \vdots & \vdots \\ 1/7^7 = 1/823543 & y = 197 \end{cases}$$

and the standard error as

$$\frac{2}{49}\sqrt{327649},$$

or approximately 23.36352. This is, of course, equal to $\sqrt{\frac{n-1}{n}} \cdot \frac{s}{\sqrt{n}} = \sqrt{\frac{6}{7}} \cdot \frac{s}{\sqrt{7}}$, where s is the standard deviation of the treatment survival times. This fact is the fortunate consequence of the mathematical tractability of the standard error for means. Other, less fortunate, situations can be handled in a similar manner.

Similar APPL statements for the control case yield an estimated standard error in the $B = +\infty$ case of

$$\frac{1}{27}\sqrt{129902},$$

or approximately 13.34886.

To complete the analysis of the difference of the means between the treatment and control group ($86.86 - 56.22 = 30.64$), this difference is only $30.64/\sqrt{23.36^2 + 13.35^2} \cong 1.14$ standard deviation units away from zero, indicating that the observed difference in the medians is also not statistically significant. \square

Example 3.14. (Comparing ranges.) The previous two examples have estimated the standard errors of measures of central tendency (e.g., the median and mean). The estimation of the standard error of a measure of dispersion, the sample range R , will now be considered.

The APPL statements required to produce the standard error of the range R for the treatment case are

```
> n := 7;
> data := [16, 23, 38, 94, 99, 141, 197];
> X := BootstrapRV(data);
> R := RangeStat(X, n);
> sqrt(Variance(R));
```

which yield

$$\frac{2}{16807}\sqrt{88781509983},$$

or approximately 35.45692. Similar APPL statements for the control case yield an estimated standard error for the range as

$$\frac{2}{4782969}\sqrt{5666287777334555},$$

or approximately 31.4762.

To complete the analysis of the difference of the ranges between the treatment and control group ($181 - 136 = 45$), we observe that the difference is $45/\sqrt{35.46^2 + 31.48^2} \cong 0.95$ standard deviations away from zero, indicating that the observed difference in the range is not statistically significant. \square

Example 3.15. (Confidence interval for range.) The previous three examples estimated the standard errors of measures of central tendency and a measure of dispersion. This example constructs a confidence interval for the sample range of the rat treatment group.

Let R be the range of the $n = 7$ observations. The APPL statements

```
> n := 7;
> data := [16, 23, 38, 94, 99, 141, 197];
> X := BootstrapRV(data);
> R := RangeStat(X, n)
> IDF(R, 0.025);
> IDF(R, 0.975);
```

result in a 95% confidence interval of $76 < R < 181$. This confidence interval has the unappealing property that the point estimator, $\hat{R} = 181$, is also the upper limit of the confidence interval.

Trosset (2001) suggested an alternative method for computing a confidence interval for the range R , which involves parametric bootstrapping. First, an exponential distribution with mean $1/\theta$ is fit to the treatment data using the APPL MLE (maximum likelihood estimator) procedure. The procedure identifies the parameter estimate for the distribution as $\hat{\theta} = \frac{7}{608}$. The (continuous) distribution of the sample range of $n = 7$ observations drawn from an exponential population with parameter $\hat{\theta} = \frac{7}{608}$ is then computed. The confidence interval is determined with the CDF procedure.

The following APPL statements yield a 95% confidence interval for the range R for $n = 7$ samples as $68 < R < 475$.

```

> data := [16, 23, 38, 94, 99, 141, 197];
> X := ExponentialRV(theta);
> thetahat := op(MLE(X, treatment, [theta]));
> Y := ExponentialRV(thetahat);
> Z := RangeStat(Y, 7);
> IDF(Z, 0.025);
> IDF(Z, 0.975);

```

The last two statements fail since IDF is currently designed to analytically (instead of numerically) determine quantiles. Thus the following two APPL statements are required to return the endpoints of the 95% confidence interval.

```

> lo := fsolve(CDF(Z, a) = 0.025, a = 0 .. 100);
> hi := fsolve(CDF(Z, a) = 0.975, a = lo .. 500);

```

□

3.4.3 Conclusion

For moderate sample sizes and test statistics having distributions that APPL can determine, the exact approach to bootstrapping can reduce computation time and eliminate resampling error.

Chapter 4

Convolutions and Products

An important operation in probability theory is to calculate the distribution of the convolution of two independent random variables X and Y . Applications of convolutions appear in many areas of mathematics, probability theory, physics, and engineering. Most texts devote the majority of their attention to convolutions of continuous random variables, rather than discrete random variables. The distribution of $Z = X + Y$, where X and Y are continuous and independent, can be obtained as

$$\begin{aligned} F_Z(z) &= F_{X+Y}(z) \\ &= \Pr(X + Y \leq z) \\ &= \iint_{x+y \leq z} f_X(x) f_Y(y) dx dy \\ &= \int_{-\infty}^{\infty} \int_{-\infty}^{z-y} f_X(x) f_Y(y) dx dy \\ &= \int_{-\infty}^{\infty} \int_{-\infty}^{z-y} f_X(x) dx f_Y(y) dy \\ &= \int_{-\infty}^{\infty} F_X(z - y) f_Y(y) dy. \end{aligned}$$

By differentiating $F_Z(z)$, we obtain the PDF $f_Z(z)$ of $Z = X + Y$, which is the

convolution of the PDFs $f_X(x)$ and $f_Y(y)$ of the random variables X and Y . The convolution formula for $f_Z(z)$ is

$$f_Z(z) = f_{X+Y}(z) = \int_{-\infty}^{\infty} f_X(z-y) f_Y(y) dy.$$

The following example illustrates the use of this formula.

Example 4.1. (Hogg & Craig, 1995, page 179) Determine the PDF of $Z = X + Y$, where X and Y are iid random variables with PDFs $f_X(x) = e^{-x}$, $0 < x < \infty$, zero elsewhere and $f_Y(y) = e^{-y}$, $0 < y < \infty$, zero elsewhere.

Solution: The PDF $f_Z(z)$ is

$$\begin{aligned} f_Z(z) &= \int_0^z e^{-z+y} \cdot e^{-y} dy \\ &= \int_0^z e^{-z} dy \\ &= ze^{-z} \quad z > 0. \end{aligned} \quad \square$$

We now turn to the case in which X and Y are discrete random variables. Without loss of generality, assume that the supports of X and Y are integer valued. For computing the PDF of $Z = X + Y$ in the discrete case, there are several approaches. The event $\{X + Y = z\}$, $z \in \mathbf{Z}$, can be written as the union of the disjoint events $\{X = \zeta, Y = z - \zeta\}$, $\{X = \zeta + 1, Y = z - (\zeta + 1)\}$, \dots , $\{X = z - \zeta, Y = \zeta\}$, where ζ is the minimum of the union of the support values of X and Y . The PDF $f_Z(z)$ of the convolution of the PDFs of the independent random variables X and Y (with integer supports) can be computed as

$$\begin{aligned} \Pr(Z = z) &= \Pr(X + Y = z) \\ &= \sum_{k=\zeta}^{z-\zeta} \Pr(X = k, Y = z - k) \\ &= \sum_{k=\zeta}^{z-\zeta} \Pr(X = k) \Pr(Y = z - k). \end{aligned} \quad (4.1)$$

The following example illustrates the use of this discrete convolution formula.

Example 4.2. (Ross, 1998, pages 270–271) If X and Y are independent Poisson random variables with respective parameters λ_1 and λ_2 , compute the PDF of $Z = X + Y$.

Solution: The PDF of a Poisson random variable X with parameter λ is $f_X(x) = \frac{e^{-\lambda} \lambda^x}{x!}$, $x = 0, 1, 2, \dots$. Since $\zeta = \min\{0, 1, 2, \dots\} = 0$, then

$$\begin{aligned}
 \Pr(Z = z) &= \Pr(X + Y = z) \\
 &= \sum_{k=0}^z \Pr(X = k, Y = z - k) \\
 &= \sum_{k=0}^z \Pr(X = k) \cdot \Pr(Y = z - k) \\
 &= \sum_{k=0}^z \frac{e^{-\lambda_1} \lambda_1^k}{k!} \cdot \frac{e^{-\lambda_2} \lambda_2^{z-k}}{(z-k)!} \\
 &= e^{-(\lambda_1 + \lambda_2)} \sum_{k=0}^z \frac{\lambda_1^k \lambda_2^{z-k}}{k! (z-k)!} \\
 &= \frac{e^{-(\lambda_1 + \lambda_2)}}{z!} \sum_{k=0}^z \frac{z!}{k! (z-k)!} \lambda_1^k \lambda_2^{z-k} \quad (\text{binomial series}) \\
 &= \frac{e^{-(\lambda_1 + \lambda_2)}}{z!} \cdot (\lambda_1 + \lambda_2)^z \quad z = 0, 1, 2, \dots
 \end{aligned}$$

Thus, $Z = X + Y$ has a Poisson distribution with parameter $\lambda_1 + \lambda_2$. □

Grinstead and Snell (1997, page 286) state that if one wants to sum more than two iid random variables, then the PDF for Z can be determined by induction. Let $Z_n = X_1 + X_2 + \dots + X_n$ be the sum of n independent random variables with common PDF $f(x)$ defined on the integers. Then the PDF of Z_1 is $f(x)$. We can write

$$Z_i = Z_{i-1} + X_i,$$

for $i = 2, 3, \dots, n$. Thus, since we know the PDF of X_i is $f(x)$, for $i = 1, 2, \dots, n$, we can find the PDF of Z_n by induction. When summing more than two iid random variables, our ConvolutionIID procedure (see Example 4.8 in Section 4.4) uses this inductive process to compute the PDF of the convolution.

For example, let X_1 , X_2 , and X_3 have PDFs $f(x) = 1/6$ for $x = 1, 2, \dots, 6$. If $Z_2 = X_1 + X_2$, then the PDF $f_{Z_2}(z)$ is

$$f_{Z_2}(z) = \begin{cases} \frac{z-1}{36} & z = 2, 3, \dots, 7 \\ \frac{13-z}{36} & z = 8, 9, \dots, 12. \end{cases}$$

The PDF of $Z_3 = X_1 + X_2 + X_3$ is the convolution of the PDFs of Z_2 and X_3 . For example, $\Pr(Z_3 = 4) = \Pr(Z_2 = 3) \cdot \Pr(X_3 = 1) + \Pr(Z_2 = 2) \cdot \Pr(X_3 = 2) = \frac{2}{36} \cdot \frac{1}{6} + \frac{1}{36} \cdot \frac{1}{6} = \frac{1}{72}$.

Due to the mathematical intractability in implementing the discrete convolution formula (Equation 4.1) for certain non-identically distributed random variables (e.g., $X \sim \text{Poisson}(\lambda)$, $Y \sim \text{geometric}(p)$) and the inefficiency in making computations with this formula for random variables with arbitrary supports (e.g., X with support $\{-216, -57, 23, 81\}$ and Y with support $\{-1002, -15, 2, 62, 211\}$), only certain convolutions can or should be computed using this formula. For random variables with arbitrary supports, the discrete convolution formula *can* be used, but it is often inefficient because one or both of the random variables have support values ranging over a large domain of non-adjacent integer values. The following example displays the inefficiency that can be encountered by using the convolution formula in Equation 4.1, even for random variables with only a small number of support values with non-zero probability.

Example 4.3. Suppose X and Y are independent discrete random variables with

PDFs defined as

$$f_X(x) = \begin{cases} 0.15 & x = -3 \\ 0.25 & x = -1 \\ 0.1 & x = 2 \\ 0.3 & x = 6 \\ 0.2 & x = 8, \end{cases} \quad f_Y(y) = \begin{cases} 0.2 & y = -2 \\ 0.1 & y = 1 \\ 0.3 & y = 5 \\ 0.4 & y = 8. \end{cases}$$

Compute the PDF of $Z = X + Y$.

Solution: The support values for Z are $z = \{-5, -3, -2, 0, 2, 3, 4, 5, 6, 7, 9, 10, 11, 13, 14, 16\}$. We'll use the formula $\sum_{k=\zeta}^{z-\zeta} \Pr(X = k, Y = z - k)$, where $\zeta = -3$, to compute only $\Pr(Z = 4)$.

$$\begin{aligned} \Pr(Z = 4) &= \sum_{k=-3}^7 \Pr(X = k, Y = 4 - k) \\ &= \Pr(X = -3) \cdot \Pr(Y = 7) + \Pr(X = -2) \cdot \Pr(Y = 6) + \\ &\quad \Pr(X = -1) \cdot \Pr(Y = 5) + \Pr(X = 0) \cdot \Pr(Y = 4) + \\ &\quad \Pr(X = 1) \cdot \Pr(Y = 3) + \Pr(X = 2) \cdot \Pr(Y = 2) + \\ &\quad \Pr(X = 3) \cdot \Pr(Y = 1) + \Pr(X = 4) \cdot \Pr(Y = 0) + \\ &\quad \Pr(X = 5) \cdot \Pr(Y = -1) + \Pr(X = 6) \cdot \Pr(Y = -2) + \\ &\quad \Pr(X = 7) \cdot \Pr(Y = -3) \\ &= 0.15 \cdot 0 + 0 \cdot 0 + 0.25 \cdot 0.3 + 0 \cdot 0 + 0 \cdot 0 + 0.1 \cdot 0 + \\ &\quad 0 \cdot 0.1 + 0 \cdot 0 + 0 \cdot 0 + 0.3 \cdot 0.2 + 0 \cdot 0 \\ &= 0.135. \end{aligned}$$

The probabilities for the other support values are computed similarly. Because of the tedious calculations needed to compute the PDF of Z by the discrete convolution

formula (Equation 4.1), we'll compute it fully in the next example using moment generating functions (MGFs). \square

Unlike the discrete convolution formula, the algorithm to be presented in this paper avoids all of the zero term computations in the construction of the PDF of Z . Also, another way to compute the PDF of Z , while avoiding the numerous zero terms, is to use the moment generating function technique.

Example 4.4. Suppose X and Y are the discrete random variables defined in Example 4.3 and $Z = X + Y$. Find the PDF of Z using the moment generating function technique.

Solution: Since X and Y are independent, the MGF of Z is

$$\begin{aligned} M_Z(t) &= E(e^{t(X+Y)}) \\ &= E(e^{tX} e^{tY}) \\ &= E(e^{tX}) E(e^{tY}) \\ &= M_X(t) M_Y(t). \end{aligned}$$

The MGFs of X and Y , respectively, are

$$M_X(t) = E(e^{tX}) = 0.15e^{-3t} + 0.25e^{-t} + 0.1e^{2t} + 0.3e^{6t} + 0.2e^{8t},$$

and

$$M_Y(t) = E(e^{tY}) = 0.2e^{-2t} + 0.1e^t + 0.3e^{5t} + 0.4e^{8t}$$

for $-\infty < t < \infty$. The MGF of Z is

$$\begin{aligned} M_Z(t) &= 0.03e^{-5t} + 0.05e^{-3t} + 0.015e^{-2t} + 0.045 + 0.045e^{2t} + 0.01e^{3t} + \\ &0.135e^{4t} + 0.06e^{5t} + 0.04e^{6t} + 0.16e^{7t} + 0.02e^{9t} + 0.04e^{10t} + \\ &0.09e^{11t} + 0.06e^{13t} + 0.12e^{14t} + 0.08e^{16t} \end{aligned}$$

for $-\infty < t < \infty$. Thus, the PDF of Z is

$$f_Z(z) = \begin{cases} 0.03 & z = -5 \\ 0.05 & z = -3 \\ 0.015 & z = -2 \\ 0.045 & z = 0 \\ 0.045 & z = 2 \\ 0.01 & z = 3 \\ 0.135 & z = 4 \\ 0.06 & z = 5 \\ 0.04 & z = 6 \\ 0.16 & z = 7 \\ 0.02 & z = 9 \\ 0.04 & z = 10 \\ 0.09 & z = 11 \\ 0.06 & z = 13 \\ 0.12 & z = 14 \\ 0.08 & z = 16. \end{cases}$$

□

In complicated examples, especially those involving continuous random variables, using the moment generating function technique to obtain convolution functions can be more efficient than direct summation or integration. Along the same lines as the moment generating function technique, the probability generating function technique can be used for determining the PDF of the convolution of discrete random variables with nonnegative integer-valued supports. Unfortunately, the implementation of these techniques in a computer algebra system (Maple) has drawbacks when the supports of the random variables X and/or Y are not integer-valued. These implementation issues are discussed in Section 4.3.

Besides the integration/summation and generating function methods already described, Parzen (1960, page 395) describes a method for using the characteristic functions of independent random variables to compute their convolution. In order to use this method, though, one must know the inversion formula of the PDF of a random variable in terms of its characteristic function. The complexity of the inversion also

makes this method unappealing for complicated or arbitrary distributions.

The purpose of this chapter is to present an algorithm for determining the distributions of convolutions of discrete random variables, especially those with finite arbitrary supports. The algorithm handles well-known distributions, such as the binomial and Poisson, but was written primarily for arbitrary distributions with moderate cardinality of their support. Computer algebra systems make it feasible to determine the distributions of convolutions for these types of distributions.

Section 4.1 describes the algorithm for determining the PDF of the convolution of discrete random variables. The algorithm that was constructed to compute this convolution appears in Section 4.2; implementation issues that arose when the algorithm was coded in a computer algebra system are given in Section 4.3. Section 4.4 provides a collection of examples that can be solved with the convolution algorithm.

4.1 Conceptual Framework

The convolution of two continuous random variables can be computed by the definition of a convolution presented in the introduction, but the definition does not give insight into the difficulty of this computation for certain random variables, such as those that are piecewise defined random variables. Glen et al. (2001) developed an algorithm for computing the distribution of the *product* of two continuous random variables in a computer algebra system. In order to obtain the convolution of continuous random variables X and Y , one can transform X and Y by the function $g_1(w) = \log(w)$, compute their product with the product algorithm, and transform the resulting product by the function $g_2(w) = e^w$ to obtain the convolution of X and Y . Since an algorithm for computing the convolution of two continuous random variables was already in place, the next natural progression was to construct an algorithm for computing the PDF of the convolution of two discrete random variables.

One way to compute the PDF of the convolution of the PDFs of two independent discrete random variables is by what we call the “brute force method.” Let X have support $\Omega_X = \{x_1, x_2, \dots, x_n\}$ and Y have support $\Omega_Y = \{y_1, y_2, \dots, y_m\}$. This method does just what the name implies, it computes all possible sums between Ω_X and Ω_Y by brute force, e.g., $x_1 + y_1, x_1 + y_2, \dots, x_1 + y_m, x_2 + y_1, x_2 + y_2, \dots, x_n + y_{m-1}, x_n + y_m$. The sums are placed in an one-dimensional array, called s , of length $n \cdot m$. The corresponding probabilities for each of these sums, $f_X(x_1) \cdot f_Y(y_1), f_X(x_1) \cdot f_Y(y_2), \dots, f_X(x_n) \cdot f_Y(y_m)$, are stored in an one-dimensional array called $Probs$, also of length $n \cdot m$. The probability in position $Probs_i$ corresponds to the sum in position $s_i, i = 1, 2, \dots, n \cdot m$.

As an example, let X and Y be the random variables introduced in Example 4.3. The arrays s and $Probs$ for the random variables X and Y are

$$s = [-5, -2, 2, 5, -3, 0, 4, 7, 0, 3, 7, 10, 4, 7, 11, 14, 6, 9, 13, 16]$$

and

$$Probs = [0.03, 0.015, 0.045, 0.06, 0.05, 0.025, 0.075, 0.1, 0.02, 0.01, \\ 0.03, 0.04, 0.06, 0.03, 0.09, 0.12, 0.04, 0.02, 0.06, 0.08].$$

We assume that s is unsorted and may contain identical values (such as 0, 4, and 7 in this particular example). The array s is sorted and appropriate updates are made to the corresponding elements in the array $Probs$. After sorting, the arrays s and $Probs$ are

$$s = [-5, -3, -2, 0, 0, 2, 3, 4, 4, 5, 6, 7, 7, 7, 9, 10, 11, 13, 14, 16]$$

and

$$Probs = [0.03, 0.05, 0.015, 0.025, 0.02, 0.045, 0.01, 0.075, 0.06, 0.06, \\ 0.04, 0.1, 0.03, 0.03, 0.02, 0.04, 0.09, 0.06, 0.12, 0.08].$$

Last, the redundancies in s are removed and the appropriate probabilities corresponding to those redundancies are combined in $Probs$. The final arrays are

$$s = [-5, -3, -2, 0, 2, 3, 4, 5, 6, 7, 9, 10, 11, 13, 14, 16]$$

and

$$Probs = [0.03, 0.05, 0.015, 0.045, 0.045, 0.01, 0.135, 0.06, 0.04, 0.16, \\ 0.02, 0.04, 0.09, 0.06, 0.12, 0.08].$$

The algorithm first employed by the Convolution procedure to sort the array s was *insertion sort* (Weiss, 1994, pages 254–255), which is contained in the APPL `InsertionSort` procedure. When $n \cdot m$ is small, the simplicity of insertion sort makes it an appropriate choice. The general strategy of insertion sort is to partition the array s into two regions: sorted and unsorted. Initially, the entire array s is considered unsorted, as already discussed. At each step, insertion sort takes the first value in the unsorted region and places it in its correct position in the sorted region. The entire array s will be sorted after the final element in the $n \cdot m$ array position is inserted.

Unfortunately, for random variables X and Y with large support sizes n and m , such as $n = m = 10$, insertion sort is inefficient. Since insertion sort is an $O(N^2)$ algorithm, where $N = n \cdot m$ in our setting, it is not an appropriate method for sorting lists containing more than a hundred or so elements. For this reason, another sorting algorithm, *heapsort* (Weiss, 1994, pages 260–262), was chosen to sort the array s . Heapsort uses a heap, which is a binary tree with special properties, to sort s . Heapsort is an $O(N \cdot \log(N))$ algorithm (Carrano et al., 1998, page 430).

Heapsort builds the array s as a maximum heap data structure. It then swaps the maximum element (the root) of the heap with the element in the last array position $s_{n \cdot m}$. The heap is rebuilt with the remaining unsorted elements in array positions s_1

through s_{n-m-1} . The maximum element of the new heap is then swapped with the element in the second to last position of the array s , which is position s_{n-m-1} . Now the last two positions in s are sorted in ascending order. The heap structure is again restored with the remaining unsorted elements in array positions s_1 through s_{n-m-2} . This swap and rebuild process repeats itself until all elements are removed from the unsorted region of the heap and placed in ascending order from the front to the back of the heap. Heapsort proved more efficient than insertion sort, especially for large values of N . The respective CPU times for a given example using insertion sort and heapsort are provided in Section 4.3 for comparison.

Shellsort, an improved insertion sort, is the algorithm employed by the mathematical software package Maple to sort polynomials (Maple 6's online help guide, 2000). Since Shellsort's "performance is quite acceptable in practice, even for N [number of elements] in the tens of thousands" (Weiss, 1994, page 260), we take advantage of Maple's sorting algorithm for polynomials (when possible) by using the moment generating function technique to compute the convolution of discrete random variables. The moment generating functions for X and Y , which are $M_X(t)$ and $M_Y(t)$ respectively, are first computed. Next, the product of the moment generating functions, $M_Z(t)$, is computed. We manipulate the terms of the moment generating function with Maple's `expand` procedure so that they are written in a fashion that Maple interprets as polynomials terms. For example, if the moment generating function is $M_Z(t) = \frac{1}{3}e^{(3t)} + \frac{1}{6}e^{(2t)} + \frac{1}{2}e^{(5t)}$, then `expand($M_Z(t)$)` returns $M_Z(t)$ as $\frac{1}{3}(e^t)^3 + \frac{1}{6}(e^t)^2 + \frac{1}{2}(e^t)^5$. The terms of the resulting expanded moment generating function are then sorted in descending order by the constant appearing in the exponent of each e^t term. Sorting the example expression $M_Z(t)$ returns $\frac{1}{2}(e^t)^5 + \frac{1}{3}(e^t)^3 + \frac{1}{6}(e^t)^2$. The probability and support values are extracted from the terms of the expression $M_Z(t)$, and the PDF of the convolution is formed. The PDF for the example expres-

sion $M_Z(t)$ is

$$f_Z(z) = \begin{cases} \frac{1}{6} & z = 2 \\ \frac{1}{3} & z = 3 \\ \frac{1}{2} & z = 5. \end{cases}$$

Although in theory this is an ideal method, Maple recognizes that expressions, such as $\frac{1}{3}e^{(3.1t)} + \frac{1}{6}e^{(2.5t)} + \frac{1}{2}e^{(5.4t)}$, are not truly polynomials and will incorrectly sort expressions with non-integer valued constants in the exponents. Since the moment generating function $M_Z(t)$ may not always have integer constants for exponents, the moment generating function technique for computing convolutions is only reasonable to use for integer supports. Using probability generating functions to compute the PDF of a convolution of random variables results in the same complications. Further implementation issues faced by moment and probability generating functions are discussed in Section 4.3.

As suggested by Nicol (2000), the sum array s can be constructed in such a way that the next largest sum element is placed in s as s is being built. Instead of constructing the array s first and then sorting it, the new algorithm constructs s by sequentially appending the next ordered element. We refer to this method as the “moving heap method,” and it involves building, deleting, and inserting sums into a minimum heap data structure. A minimum heap contains its smallest element in the root (the top node of the heap), rather than its largest as in a maximum heap.

The idea behind this sorting algorithm is the construction of a two-dimensional “conceptual” array A . The array A is not instantiated to save memory, but is helpful in explaining the nature of the algorithm. The array A has $m + 1$ rows and $n + 1$ columns. The array A , illustrated in Example 4.5, is displayed in an unusual manner in order to resemble the axes in the Cartesian coordinate system. Without loss of generality, we assume that the supports of X and Y are arranged in increasing order;

i.e., $x_1 < x_2 < \dots < x_n$ and $y_1 < y_2 < \dots < y_m$. The array cell (i, j) contains the sum $A_{i,j} = y_i + x_j$ for $i = 1, 2, \dots, m$ and $j = 1, 2, \dots, n$. The cells in row $m + 1$ of A hold a 0 or 1 for each column $j = 1, 2, \dots, n$ to indicate whether the cell in column j is “active,” which means its entry is in the minimum heap. Thus $A_{m+1,j} = 0$ or 1 for $j = 1, 2, \dots, n$. Likewise, the cells in column $n + 1$ of A also hold a 0 or 1 for each row $i = 1, 2, \dots, m$ to indicate whether the cell in row i is “active”; i.e., $A_{i,n+1} = 0$ or 1 for $i = 1, 2, \dots, m$. The $(m + 1, n + 1)$ cell is not used in the algorithm. Example 4.5 illustrates what is meant by an “active” cell.

Since $x_j < x_{j+1}$ for $j = 1, 2, \dots, n - 1$ and $y_i < y_{i+1}$ for $i = 1, 2, \dots, m - 1$, the entry in cell (i, j) is always guaranteed to be less than both the entries in cells $(i + 1, j)$ and $(i, j + 1)$; i.e., $A_{i,j} < A_{i+1,j}$ and $A_{i,j} < A_{i,j+1}$. This result, along with other proven properties of the array A in Appendix C, allow the algorithm to move the smallest number of candidate entries for the next largest sum from the array A to the minimum heap. Thus this algorithm moves from the southwest cell to the northeast cell of the array A placing the next largest sum into s after first placing the competing sums into a minimum heap.

Since this process and its intricacies are best explained by an example, we’ll reintroduce X and Y , the random variables from Example 4.3.

Example 4.5. Let X and Y have PDFs:

$$f_X(x) = \begin{cases} 0.15 & x = -3 \\ 0.25 & x = -1 \\ 0.1 & x = 2 \\ 0.3 & x = 6 \\ 0.2 & x = 8, \end{cases} \quad f_Y(y) = \begin{cases} 0.2 & y = -2 \\ 0.1 & y = 1 \\ 0.3 & y = 5 \\ 0.4 & y = 8. \end{cases}$$

Use the “moving heap method” to determine the PDF of $Z = X + Y$.

Solution: Construct the 5×6 array A . Set $A_{i,n+1} = A_{i,6} = 0$ for $i = 1, 2, 3, 4$ and $A_{m+1,j} = A_{5,j} = 0$ for $j = 1, 2, 3, 4, 5$. The smallest value in A is positioned in cell $(1, 1)$ and is $A_{1,1} = y_1 + x_1 = -5$. The algorithm designates the cell $(1, 1)$ as an “active” cell in A by setting $A_{m+1,1} = A_{5,1} = 1$ and $A_{1,n+1} = A_{1,6} = 1$. The zeros in the other cells of row five and column six remain. Figure 4.1 displays this initial array. The entries of A increase in value as one moves up, to the right, or a combination of both (as in the Cartesian coordinate system).

row 5		1	0	0	0	0
row 4	8					0
row 3	5					0
row 2	1					0
row 1	-2	-5				1
		-3	-1	2	6	8
		col 1	col 2	col 3	col 4	col 5

Figure 4.1: Array A with active cell $(1, 1)$, which contains the entry $A_{1,1} = -5$.

As in the brute force method, the one-dimensional array s of length $n \cdot m$ holds the sums of the supports of the random variables X and Y . The corresponding probabilities for each of these sums will again be stored in the one-dimensional array called $Probs$, also of length $n \cdot m$. Clearly, the first (smallest) sum to be placed in the first position, $s_1 = -5$, of the array s is $A_{1,1}$. Accordingly, $f_X(x_1) \cdot f_Y(y_1) = 0.03$ is placed in the first position, $Probs_1$, in the $Probs$ array. After setting $s_1 = A_{1,1} = -5$ and $Probs_1 = \Pr(Z = A_{1,1}) = f_X(x_1) \cdot f_Y(y_1) = 0.03$, the cell $(1, 1)$ becomes inactive. In order to reflect the absence of an element in the first row and first column, reset

$A_{m+1,1} = A_{5,1} = 0$ and $A_{1,n+1} = A_{1,6} = 0$. The next two cells to become “active” (i.e., these cells may contain the next largest sum) in the array A are $A_{1,2} = y_1 + x_2 = -3$ and $A_{2,1} = y_2 + x_1 = -2$. Since cell (1, 2) in A is now active, reset $A_{1,6} = 1$ and set $A_{5,2} = 1$. Similarly, since cell (2, 1) is active, set $A_{2,6} = 1$ and reset $A_{5,1} = 1$. The purpose of these ones and zeros along the boundary of the A array is to assure that there is no more than one active cell in each row and column. Figure 4.2 contains the current view of array A .

row 5	1	1	0	0	0		
row 4	8					0	
row 3	5					0	
row 2	1	-2				1	
row 1	-2	-5	-3			1	
		-3	-1	2	6	8	
		col 1	col 2	col 3	col 4	col 5	col 6

Figure 4.2: Array A after $A_{1,1}$ has been removed and added to the one-dimensional sum array s . The cells (1, 2) and (2, 1) are active, as indicated by the ones in cells (1, 6), (5, 2), (2, 6), and (5, 1).

The values $A_{1,2}$ and $A_{2,1}$ are used to construct a minimum heap H . Informally, a *heap* is a complete binary tree with a special ordering property of its nodes. A complete binary tree is a tree that is completely filled with the possible exception of the bottom level, which is filled from left to right. Figure 4.3 contains illustrations of structures which are and are not complete binary trees. Each node of the tree has one parent, except the *root* of the tree, which has no parent. In a *minimum heap*, the smallest element of the heap is contained in its root. In the upper-right tree in

Figure 4.3, a is the root of the tree. Nodes b and c are a 's children, where b is the left child and c is the right child. (According to the definition of a complete binary tree, when a node above the bottom level of the tree has only one child, it must be a left child.) Node b is the parent to nodes d and e . The height of a tree is the number of nodes from the root to a node at the bottom level of the tree. The heights of the three top trees in Figure 4.3, for example, are three, three, and four, respectively. A complete binary tree of height h has between 2^{h-1} and $2^h - 1$ nodes (Carrano et al., 1998, page 496).

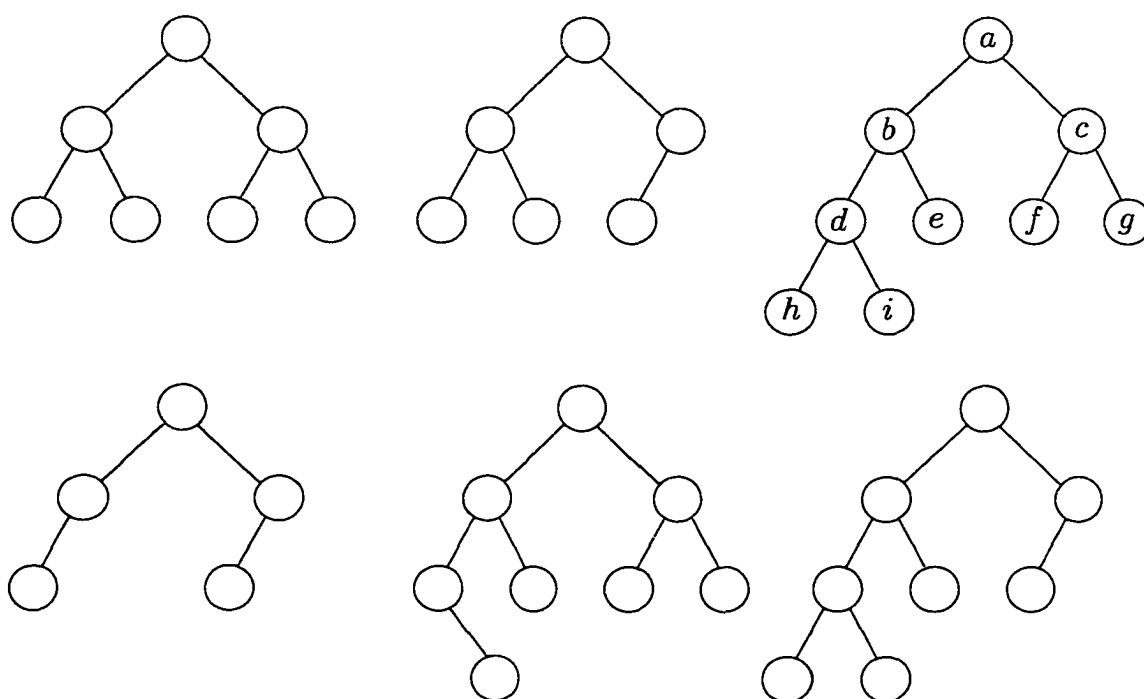


Figure 4.3: Six binary trees. The top three trees are complete binary trees and the bottom three are not.

Thus, a minimum heap is a complete binary tree with the special ordering property that each parent node contains a value less than or equal to the values in its children's nodes. Because of this ordering property, the smallest value in a minimum heap will always be at the root.

The binary heap H formed with the values $A_{1,2}$ and $A_{2,1}$ is in Figure 4.4. The

next sum to be entered into s in position s_2 is the root of the heap. Since $A_{1,2} = -3$ is the root, it is removed from the heap H and placed in s_2 , while its corresponding probability is placed in $Probs_2$. Because the entry $A_{1,2}$ is removed from the array A , reset $A_{1,6} = 0$ and $A_{5,2} = 0$ to indicate that row one and column two no longer contain an active cell. After these changes, array A is displayed in Figure 4.5.

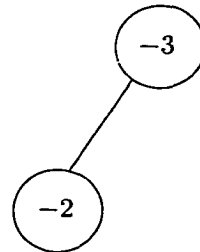


Figure 4.4: Heap H containing entries $A_{1,2} = -3$ and $A_{2,1} = -2$.

row 5		1	0	0	0	0	
row 4	8					0	
row 3	5					0	
row 2	1	-2				1	
row 1	-2	-5	-3			0	
		-3	-1	2	6	8	
		col 1	col 2	col 3	col 4	col 5	col 6

Figure 4.5: Array A after $A_{1,2} = -3$ is removed and appended to s . Cell (2, 1) is the only active cell. Candidates to become active are cells (1, 3) and (2, 2). Cell (2, 2) cannot become active since row two already contains an active cell.

After setting cell (1, 2) to inactive, the two cells that may enter into the array A (if the corresponding row and column do not already contain an active cell) are

cells (3, 1) and (2, 2). Since row two contains an active cell, then entry $A_{2,2}$ is not activated since its sum is greater than $A_{2,1}$. However, cell (1, 3) does become active, and its entry is $A_{1,3} = y_1 + x_3 = 0$. Hence, $A_{1,6} = 1$ and $A_{5,3} = 1$. After these changes, array A is displayed in Figure 4.6.

row 5		1	0	1	0	0	
row 4	8						0
row 3	5						0
row 2	1	-2					1
row 1	-2	-5	-3	0			1
		-3	-1	2	6	8	
		col 1	col 2	col 3	col 4	col 5	col 6

Figure 4.6: Array A with active cells (1, 3) and (2, 1).

The entry $A_{1,3}$ is inserted into the heap H , and the heap is rebuilt to fulfill its ordering property. After the addition of $A_{1,3}$, the heap H is displayed in Figure 4.7. The minimum element, $A_{2,1}$, is removed from the root of the heap and placed in the sum array s in position s_3 . Its corresponding probability is placed in $Probs_3$.

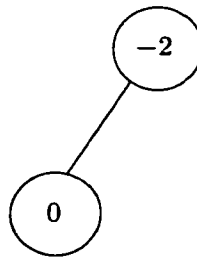


Figure 4.7: Heap H containing entries $A_{2,2} = -2$ and $A_{1,3} = 0$.

The two cells that may enter the array A after the removal of the $A_{2,1}$ entry are in

cells (2, 2) and (3, 1), as indicated by the arrows in Figure 4.8. Both cells (2, 2) and (3, 1) become active, and their values are $A_{2,2} = y_2 + x_2 = 0$ and $A_{3,1} = y_3 + x_1 = 2$. Hence, $A_{2,6} = 1$, $A_{5,2} = 1$, $A_{3,6} = 1$, and $A_{5,1} = 1$, as displayed in Figure 4.9. Entries $A_{2,2}$ and $A_{3,1}$ are inserted into the heap H , and H is again rebuilt. Its structure is displayed in Figure 4.10.

row 5		0	0	1	0	0	
row 4	8					0	
row 3	5					0	
row 2	1	-2				0	
row 1	-2	-5	-3	0		1	
		-3	-1	2	6	8	
		col 1	col 2	col 3	col 4	col 5	col 6

Figure 4.8: Array A after $A_{2,2} = -2$ is removed. Cell (3, 1) is the only active cell. Candidates to become active are cells (2, 2) and (3, 1).

Moving ahead to the seventeenth pass through the construction of the array A , its appearance is displayed in Figure 4.11. $A_{3,4} = 11$ is placed in s_{17} , and values $A_{3,5} = 13$ and $A_{4,4} = 14$ are activated in the array A and inserted into the heap H . Since $A_{3,5}$ is the root of the heap, it is deleted and placed in s_{18} . No new element is allowed to enter the heap, so the root element of the heap is now $A_{4,4} = 14$, and it is removed and placed in s_{19} . The last entry to be activated is $A_{4,5} = 16$, and it is placed in position s_{20} of the sum array s .

Thus, after twenty iterations of this process, s and $Probs$ arrays are

$$s = [-5, -3, -2, 0, 0, 2, 3, 4, 4, 5, 6, 7, 7, 7, 9, 10, 11, 13, 14, 16]$$

row 5		1	1	1	0	0	
row 4	8					0	
row 3	5	2				1	
row 2	1	-2	0			1	
row 1	-2	-5	-3	0		1	
		-3	-1	2	6	8	
		col 1	col 2	col 3	col 4	col 5	col 6

Figure 4.9: Array A with active cells $(1, 3)$, $(2, 2)$, and $(3, 1)$.

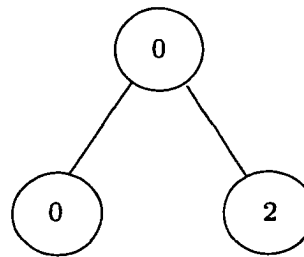


Figure 4.10: Heap H with entries $A_{1,3} = 0$, $A_{2,2} = 0$ and $A_{1,3} = 2$.

and

$$\begin{aligned}
 Probs = & [0.03, 0.05, 0.015, 0.025, 0.02, 0.045, 0.01, 0.075, 0.06, 0.06, \\
 & 0.04, 0.1, 0.03, 0.03, 0.02, 0.04, 0.09, 0.06, 0.12, 0.08].
 \end{aligned}$$

which are the same arrays encountered by using the moment generating function method. The redundancies are removed from s and the appropriate probabilities are combined in $Probs$ to complete the algorithm. This could have been embedded into the iterative steps of the algorithm to save memory. Thus, the PDF of $Z = X + Y$ is the same as determined in Example 4.4. \square

row 5		0	0	0	1	0	
row 4	8	5	7	10		0	
row 3	5	2	4	7	11	1	
row 2	1	-2	0	3	7	9	
row 1	-2	-5	-3	0	4	6	
		-3	-1	2	6	8	
		col 1	col 2	col 3	col 4	col 5	col 6

Figure 4.11: Array A with its seventeenth active cell (3, 4).

4.2 Algorithm

The algorithm for the `Convolution(X, Y)` procedure returns the PDF of the convolution of the PDFs of the independent random variables X and Y . A brief description of the algorithm follows.

If X and Y are continuous, the PDF of the convolution $Z = X + Y$ is computed with the continuous convolution formula. If X and Y are discrete, their supports, finite or infinite, dictate which of the methods described in Section 4.1 is used to compute the convolution.

The convolution of the PDFs of X and Y with finite support is computed either using the `BruteForceMethod` or `MovingHeapMethod` procedures, whose algorithms appear in Appendices D and E, respectively. The PDF of the convolution of Z is stored in a list-of-sublists format. The list of elements $f(z_1), f(z_2), \dots, f(z_{n-m})$ are the probability values of Z , while z_1, z_2, \dots, z_{n-m} are its support values. The one-dimensional array s is created to contain the sums extracted from the array A . Similarly, the one-dimensional array $Probs$ is created to hold the probability values

corresponding to the sums in s .

The zeros (if any) at the end of the *Probs* array do not represent probability values; they correspond to the zeros in the *Probs* array that are not support values. These extra zeros indicate that there are identical values in the support of Z . The non-zero probability values are removed from *Probs* and placed in the array *FinalProbs*. The support values that correspond to the removed probability values are removed and placed in the array *FinalSupport*.

If the supports of the random variables X and Y are infinite, either the discrete convolution formula is used to compute the convolution or the APPL MGF procedure is used to determine the MGF of the product of X and Y .

If either X or Y has infinite support, while the other has finite support, the product of their MGFs is printed. If both X and Y have infinite support and the discrete convolution formula formed with their PDFs results in an intractable sum, then the product of their MGFs is printed. Otherwise, the discrete convolution formula is used to determine the convolution of their PDFs.

Unless the MGF for Z or the PDF of the convolution for X and Y (with $n = 1$ and $m = 1$) has already been printed or returned, the PDF $f_Z(z)$ is returned.

Procedure Convolution: Computes the PDF of the convolution of the PDFs of two independent random variables

Input: The random variables X and Y . Convert X and Y to their PDF representations, if necessary. The support of X is Ω_X and the support of Y is Ω_Y .

Output: The PDF of $Z = X + Y$.

If X and Y are continuous

$$f_Z(z) \leftarrow \int_{-\infty}^{\infty} f_X(z-y) f_Y(y) dy$$

Else

[If X and Y are discrete]

If X and Y have finite support

$$n \leftarrow |\Omega_X|$$

$$m \leftarrow |\Omega_Y|$$

If $n \cdot m \leq 100$

$$f_Z(z) \leftarrow \text{BruteForceMethod}(X, Y)$$

```

Else
   $f_Z(z) \leftarrow \text{MovingHeapMethod}(X, Y)$ 
  If ( $n = 1$  and  $m = 1$ ) [i.e.  $f_Z(z) = 1$  for  $z = c \in \mathbb{R}$ ]
    return( $f_Z(z)$ )
  Dimension  $s[n \cdot m]$  [Create the sums array  $s$ ]
  Dimension  $Probs[n \cdot m]$  [Create the probability array  $Probs$ ]
  For  $i \leftarrow 1$  to  $n \cdot m$ 
     $s_i \leftarrow 0$ 
     $Probs_i \leftarrow 0$ 
   $s_1 \leftarrow z_1$ 
   $s_2 \leftarrow z_2$ 
   $Probs_1 \leftarrow f(z_1)$ 
   $k \leftarrow 2$ 
   $j \leftarrow 2$ 
  While ( $k < n \cdot m$ ) do
     $Probs_j \leftarrow Probs_j + f(z_k)$ 
    If  $z_k \neq z_{k+1}$  then [Eliminate redundant support values]
       $j \leftarrow j + 1$ 
       $s_j \leftarrow z_{k+1}$ 
       $k \leftarrow k + 1$ 
     $Probs_j \leftarrow Probs_j + f(z_k)$ 
   $NumZeros \leftarrow 0$ 
  For  $i \leftarrow n \cdot m$  to 1 by  $-1$  while  $Probs_i = 0$ 
     $NumZeros \leftarrow NumZeros + 1$ 
  Dimension  $FinalProbs[1, n \cdot m - NumZeros]$ 
  Dimension  $FinalSupport[1, n \cdot m - NumZeros]$ 
  For  $i \leftarrow 1$  to  $(n \cdot m - NumZeros)$ 
     $FinalProbs_i \leftarrow Probs_i$ 
     $FinalSupport_i \leftarrow s_i$ 
   $f_Z(z) \leftarrow [FinalSupport, FinalProbs]$ 
Else if ( $X$  or  $Y$  has infinite support or
 $X$  and  $Y$  have infinite support with intractable discrete convolution sum)
   $mgfx \leftarrow \text{MGF}(X)$ 
   $mgfy \leftarrow \text{MGF}(Y)$ 
   $mgfprod \leftarrow mgfx \cdot mgfy$ 
  print( $mgfprod$ )
  return
Else [Discrete convolution formula]
   $f_Z(z) \leftarrow \sum_{k=0}^z (f_X(z)) \cdot (f_Y(z - k))$ 
Else
  print("ERROR:  $X$  and  $Y$  must both be continuous or discrete")
  return
return( $f_Z(z)$ )

```

4.3 Implementation

When the supports Ω_X and Ω_Y of X and Y , respectively, are finite and $|\Omega_X| \cdot |\Omega_Y| \leq 10000$, the `Convolution` procedure uses the “brute force method” to determine every possible sum (and its probability) between the supports of the random variables X and Y . The list of sums are ordered and the identical ones are combined. The corresponding probability values are repositioned to match their corresponding sums. One important reason for sorting the sums is that all other APPL procedures assume that discrete distributions, written in their list-of-sublists *NoDot* form, have supports listed in increasing order without repeated values. To be consistent with the APPL language (and textbooks), the sums are sorted. Also, placing the values of the support into a list in sorted order means that tied $y_i + x_j$ values *can* be combined dynamically as the algorithm proceeds.

As mentioned in Section 4.1, the first sorting method chosen to sort the list of sums created by the brute force method was insertion sort. It was a viable candidate because of its straightforward implementation and efficiency in computing convolutions of random variables with small supports. Unfortunately, as the supports of random variables grew larger (e.g., random variables with 50 random numbers chosen on the interval $(-1, 1)$ as a support), the time used to compute the PDF of their convolutions became unreasonably large. A faster sorting method for random variables with large supports was required. Heapsort was employed and is now implemented in the `Convolution` procedure for sorting the list of sums created by the brute force method.

Maple uses Shellsort to sort polynomials. In order to use the Shellsort procedure in Maple, the MGFs of X and Y need to be computed. The product of the MGFs of X and Y is an expression composed of exponential terms e^{kt} , where $k \in \mathbb{R}$, $t > 0$. Letting $u = e^t$, the MGF of the product can be rewritten as a polynomial-type expression. For example, if $u = e^t$, then $M_Z(t) = \frac{1}{3}e^{3t} + \frac{1}{6}e^{2t} + \frac{1}{2}e^{5t}$ can be rewritten

as $M_Z(u) = \frac{1}{3}u^3 + \frac{1}{6}u^2 + \frac{1}{2}u^5$. The Shellsort procedure sorts the polynomial expression, and the PDF of the convolution of X and Y is retrieved from this expression. Instead of moment generating functions, probability generating functions can be used in the process. MGFs were chosen over PGFs since PGFs can only be formed when the discrete distribution has nonnegative integer support.

The method of computing convolutions via their MGFs for random variables with finite supports was discarded after realizing that Maple can only sort “true” polynomial expressions. Maple is unable to sort an expression with non-integer values as exponents of variables, such as $\frac{1}{3}(e^t)^{3/2} + \frac{1}{6}(e^t)^{1/2} + \frac{1}{2}(e^t)^{5.5}$. Since the `Convolution` procedure was intended to be used on all types of discrete distributions, including those with negative, non-integer supports, the MGF method was abandoned as a method for determining the PDF of the sum of random variables with finite supports in the `Convolution` procedure. The extra time involved in checking for appropriate exponent values also had an effect on the MGF method’s efficiency.

For random variables with large finite supports (e.g., $|\Omega_X| \cdot |\Omega_Y| > 10000$), heapsort also proved inefficient. As an alternative approach to the brute force method with heapsort, Nicol (2000) suggested constructing a heap dynamically and sorting the list of sums sequentially, instead of first building and *then* sorting the sum list. The algorithm for this “moving heap” was written into the APPL `MovingHeapMethod` procedure and implemented in `Convolution` for X and Y with finite supports, where $|\Omega_X| \cdot |\Omega_Y| > 10000$. Not only was the moving heap method efficient, but it saved memory space since the heap always contained $\min\{|\Omega_X|, |\Omega_Y|\}$ or fewer entries.

The `Convolution` procedure was tested on random variables with large supports by using the `BruteForceMethod` with insertion sort, the `BruteForceMethod` with heapsort, and the `MovingHeapMethod`. A brief comparison analysis (by hand) of the three methods suggested that `MovingHeapMethod` would yield the best (fastest) times

for computing a convolution of random variables with large supports (i.e., $|\Omega_X| \cdot |\Omega_Y| > 10000$). Test cases of random variables with increasing support sizes were performed to confirm this assumption.

The test for comparing the methods involved generating $|\Omega_X|$ and $|\Omega_Y|$ random numbers between -1 and 1 and making them the support values for the random variables X and Y , respectively. If $|\Omega_X| = 3$, for example, then three random numbers x_1, x_2, x_3 would be generated as Ω_X . To conform to the APPL list-of-sublists *NoDot* data structure, the values would be sorted and placed into the second sublists of the respective random variables. For our example, we would list x_1, x_2, x_3 in increasing order and rename them as $x_{(1)}, x_{(2)}$, and $x_{(3)}$. The probabilities, which had no effect on the efficiency of the different algorithms, were assigned to be equally-likely for all support values; i.e., $f(x_i) = 1/|\Omega_X|$ for $i = 1, 2, \dots, |\Omega_X|$ and $f(y_j) = 1/|\Omega_Y|$ for $j = 1, 2, \dots, |\Omega_Y|$. For our example, the list-of-sublists form of X is

$$\left[\left[\frac{1}{|\Omega_X|}, \frac{1}{|\Omega_X|}, \frac{1}{|\Omega_X|} \right], [x_{(1)}, x_{(2)}, x_{(3)}], ["Discrete", "PDF"] \right];$$

The CPU times on a 266 mZ machine for determining the PDF of the convolution of random variables of increasing support sizes $|\Omega_X|$ and $|\Omega_Y|$ appear in Table 4.1.

When either one or both random variables' supports are infinite, either the convolution of their PDFs is computed via the discrete convolution formula or the MGF of their product is determined. If one of the random variables has infinite support, while the other has finite support, the MGF of their product is returned. Currently, APPL does not contain a procedure to convert the MGF of a random variable to its PDF form. In future work, this recognition process may become an APPL procedure.

Two random variables with infinite support do not guarantee that the PDF of their convolution can be determined by the discrete convolution formula. Only tractable summations, such as the convolution formula for two Poisson random variables as in

Table 4.1: CPU times (in seconds) for the convolution of random variables X and Y by the BruteForceMethod with insertion sort, the BruteForceMethod with heapsort, and the MovingHeapMethod for arbitrary distributions with arbitrary support values ranging in increasing value from -1 to 1 .

Support sizes, $ \Omega_X = \Omega_Y $	BruteForceMethod with insertion sort	BruteForceMethod with heapsort	MovingHeapMethod
50	70.5	10.6	15.3
60	143.1	18.1	24.0
70	313.3	29.1	34.6
80	518.0	45.5	50.0
90	824.0	69.9	69.3
95	1050.5	85.3	80.6
100	1263.5	101.3	93.5
110	2037.6	153.2	123.3
120	2897.4	201.7	163.0
125	3283.5	257.5	173.9
130	–	284.8	201.6
140	–	394.8	236.4
150	–	541.1	320.1
160	–	728.8	377.3
170	–	969.0	454.6
175	–	1127.9	506.5
180	–	1319.1	578.5
190	–	1723.2	671.8
200	–	2210.3	829.0

Example 4.2, can be computed. This means that instead of determining the PDF for the convolution of the PDFs of some random variables, such as a Poisson with parameter $\lambda = 5$ and a geometric with parameter $p = 0.3$, the Convolution procedure only computes the product of their MGFs.

4.4 Examples

The following examples use the algorithm described in Section 4.2 to determine the PDF of the convolution of independent random variables. Examples for a variety of

random variables are provided to illustrate the utility of the algorithm. Returning first to Examples 4.1 through 4.4 from the introduction of this chapter, we can use the Convolution procedure to determine their solutions.

Example 4.1 Revisited. Compute the PDF of $Z = X + Y$, where X and Y are iid random variables with PDFs $f(x) = e^{-x}$, $0 < x < \infty$, zero elsewhere and $f(y) = e^{-y}$, $0 < y < \infty$, zero elsewhere.

Solution: In APPL, we first define X and Y to be exponential(1) random variables, which are predefined in APPL. The Convolution procedure then finds the PDF of $Z = X + Y$. The statements

```
> X := ExponentialRV(1);
> Y := ExponentialRV(1);
> Z := Convolution(X, Y);
```

return the PDF in its list-of-sublists APPL format as

$$\left[\left[z \rightarrow ze^{-z} \right], [0, \infty], ["Continuous", "PDF"] \right]. \quad \square$$

Example 4.2 Revisited. If X and Y are independent Poisson random variables with respective parameters λ_1 and λ_2 , compute the PDF of $Z = X + Y$.

Solution: In APPL, define X as a Poisson random variable with parameter `lambda1` and Y as a Poisson random variable with parameter `lambda2`. The Poisson random variable is also predefined in APPL. The PDF $Z = X + Y$ is found with the statements

```
> X := PoissonRV(lambda1);
> Y := PoissonRV(lambda2);
> Z := Convolution(X, Y);
```

which returns the PDF of Z as

$$\left[\left[z \rightarrow \frac{(z+1)e^{-\lambda_1-\lambda_2}(\lambda_2+\lambda_1)^z}{(z+1)!} \right], [0.. \infty], ["Discrete", "PDF"] \right].$$

Using the Maple simplify procedure, the resulting PDF after simplification is

$$f(z) = \frac{e^{-\lambda_1 - \lambda_2} (\lambda_2 + \lambda_1)^z}{\Gamma(z + 1)} \quad z = 0, 1, \dots,$$

which is easy to recognize in its standard form as

$$f(z) = \frac{e^{-\lambda_1 - \lambda_2} (\lambda_1 + \lambda_2)^z}{z!} \quad z = 0, 1, \dots \quad \square$$

We are fortunate in Example 4.2 that APPL can compute the PDF by the discrete convolution formula by simplifying the Maple sum $\sum_{k=0}^x \frac{\lambda_1^k e^{-\lambda_1} \lambda_2^{x-k} e^{-\lambda_2}}{k!(x-k)!}$. Unfortunately, Maple can only simplify certain expressions, so in some instances we cannot simplify the PDF by the discrete convolution formula. In Example 4.14, it is shown that Maple can also simplify the discrete convolution formula for a pair of iid geometric random variables.

Examples 4.3, 4.4, & 4.5 Revisited. X and Y are independent discrete random variables with PDFs defined as

$$f_X(x) = \begin{cases} 0.15 & x = -3 \\ 0.25 & x = -1 \\ 0.1 & x = 2 \\ 0.3 & x = 6 \\ 0.2 & x = 8, \end{cases} \quad f_Y(y) = \begin{cases} 0.2 & y = -2 \\ 0.1 & y = 1 \\ 0.3 & y = 5 \\ 0.4 & y = 8. \end{cases}$$

Find the PDF of Z .

Solution: Define the random variables X and Y in APPL's list-of-sublists format.

Compute the PDF of $Z = X + Y$ with the following statements

```
> X := [[0.15, 0.25, 0.1, 0.3, 0.2], [-3, -1, 2, 6, 8],
        ["Discrete", "PDF"]];
```

```
> Y := [[0.2, 0.1, 0.3, 0.4], [-2, 1, 5, 8], ["Discrete", "PDF"]];
> Z := Convolution(X, Y);
```

which return the PDF of Z as

```
[[0.03, 0.05, 0.015, 0.045, 0.045, 0.01, 0.135, 0.06, 0.04, 0.16, 0.02, 0.04,
0.09, 0.06, 0.12, 0.08], [-5, -3, -2, 0, 2, 3, 4, 5, 6, 7, 9, 10, 11, 13, 14, 16],
["Discrete", "PDF"]].
```

□

Example 4.6. (Sveshnikov, 1968, page 136) Let X and Y be independent random variables; X assumes three possible values 0, 1, 3 with probabilities $\frac{1}{2}$, $\frac{3}{8}$, and $\frac{1}{8}$, and Y assumes two possible values 0 and 1 with probabilities $\frac{1}{3}$, $\frac{2}{3}$. Find the PDF of the random variable $Z = X + Y$.

Solution: By hand, we can compute the PDF of Z with probability generating functions (PGF). The PGFs G of X and Y , respectively, are

$$G_X(t) = E[t^X] = \frac{1}{8}t^3 + \frac{3}{8}t + \frac{1}{2} \quad \text{and} \quad G_Y(t) = E[t^Y] = \frac{2}{3}t + \frac{1}{3}$$

for $-\infty < t < \infty$. Thus, the PGF of $Z = X + Y$ is

$$G_Z(t) = \frac{1}{12}t^4 + \frac{1}{24}t^3 + \frac{1}{4}t^2 + \frac{11}{24}t + \frac{1}{6} \quad -\infty < t < \infty.$$

Hence, the PDF of Z is

$$f_Z(z) = \begin{cases} \frac{1}{6} & z = 0 \\ \frac{11}{24} & z = 1 \\ \frac{1}{4} & z = 2 \\ \frac{1}{24} & z = 3 \\ \frac{1}{12} & z = 4. \end{cases}$$

In APPL, define X and Y as list-of-sublists and then apply the Convolution procedure to achieve the same result.

```

> X := [[1 / 2, 3 / 8, 1 / 8], [0, 1, 3], ["Discrete", "PDF"]];
> Y := [[1 / 3, 2 / 3], [0, 1], ["Discrete", "PDF"]];
> Z := Convolution(X, Y);

```

□

Other measures, such as the mean and variance, of a distribution can be found with the use of additional APPL procedures, as seen in the next example.

Example 4.7. (Hogg & Tanis, 1993, page 297) Let X_1 and X_2 be observations of a random sample of size $n = 2$ from a distribution with PDF $f(x) = \frac{x}{6}$, $x = 1, 2, 3$. Find the PDF of $Y = X_1 + X_2$, and determine the mean and variance of the sum.

Solution: The PGFs of X_1 and X_2 are

$$G_{X_1}(t) = G_{X_2}(t) = \frac{1}{6}t + \frac{1}{3}t^2 + \frac{1}{2}t^3 \quad -\infty < t < \infty.$$

Thus, $G_Y(t)$ is

$$G_Y(t) = \frac{1}{4}t^6 + \frac{1}{3}t^5 + \frac{5}{18}t^4 + \frac{1}{9}t^3 + \frac{1}{36}t^2 \quad -\infty < t < \infty$$

and $f_Y(y)$ is

$$f_Y(y) = \begin{cases} \frac{1}{36} & y = 2 \\ \frac{1}{9} & y = 3 \\ \frac{5}{18} & y = 4 \\ \frac{1}{3} & y = 5 \\ \frac{1}{4} & y = 6. \end{cases}$$

The mean and the variance of Y , respectively, are

$$E[Y] = G'_Y(t) \Big|_{t=1} = \left[\frac{3}{2}t^5 + \frac{5}{3}t^4 + \frac{10}{9}t^3 + \frac{1}{3}t^2 + \frac{1}{18}t \right] \Big|_{t=1} = \frac{14}{3}$$

and

$$G''_Y(1) + G'_Y(1) - [G'_Y(1)]^2 = \frac{10}{9}.$$

In APPL, the mean and variance of Y are computed with the statements

```

> X := [[x -> x / 6], [1 .. 3], ["Discrete", "PDF"]];
> Y := Convolution(X, X);
> Mean(Y);
> Variance(Y);

```

□

Example 4.8. (Hogg & Craig, 1995, page 230) Find the probability of obtaining a total of 14 in a single toss of four dice.

Solution: Let X be the PDF $f_X(x) = \frac{1}{6}$, $x = 1, 2, \dots, 6$. The PGF G of X is

$$G_X(t) = \frac{1}{6}t^6 + \frac{1}{6}t^5 + \frac{1}{6}t^4 + \frac{1}{6}t^3 + \frac{1}{6}t^2 + \frac{1}{6}t \quad -\infty < t < \infty.$$

The PDF of $Z = X_1 + X_2 + X_3 + X_4$ can be found by computing $[G_X(t)]^4$, which is

$$\begin{aligned}
[G_X(t)]^4 = & \frac{1}{1296}t^{24} + \frac{1}{324}t^{23} + \frac{5}{648}t^{22} + \frac{5}{324}t^{21} + \frac{35}{1296}t^{20} + \\
& \frac{7}{162}t^{19} + \frac{5}{81}t^{18} + \frac{13}{162}t^{17} + \frac{125}{1296}t^{16} + \frac{35}{324}t^{15} + \frac{73}{648}t^{14} + \\
& \frac{35}{324}t^{13} + \frac{125}{1296}t^{12} + \frac{13}{162}t^{11} + \frac{5}{81}t^{10} + \frac{7}{162}t^9 + \frac{35}{1296}t^8 \\
& \frac{5}{324}t^7 + \frac{5}{648}t^6 + \frac{1}{324}t^5 + \frac{1}{1296}t^4 \quad -\infty < t < \infty.
\end{aligned}$$

Thus, $\Pr(Z = 14) = \frac{73}{648}$.

In APPL, define X as a uniform discrete random variable (predefined in APPL) with parameter $1/6$. The `ConvolutionIID(X, n)` procedure computes the PDF of the convolution of n iid random variables X . This procedure contains a Maple “for loop” which calls `Convolution` n times.

```

> X := UniformDiscreteRV(1, 6);
> Z := ConvolutionIID(X, 4);
> PDF(Z, 14);

```

The APPL PDF procedure computes the probability that Z is 14, which is $\frac{73}{648}$. □

Examples 4.9 and 4.10 are from the article “Getting Normal Probability Approximations without Using Normal Tables” by Thompson (2000). His paper discusses an

alternate approach to approximating probabilities involving sums of discrete random variables using the PDF for the normal distribution. He lets S denote the sum of n independent discrete random variables, and assumes that S takes on consecutive integer values. Letting $\mu = E(S)$ and $\sigma^2 = \text{Var}(S)$, he argues that for sufficiently large values of n , S is approximately normally distributed. Using the standard continuity correction, he gets

$$\Pr(S = s) = \Pr(s - 0.5 < N(\mu, \sigma^2) < s + 0.5).$$

Calculating a midpoint approximation using a single subinterval, the *normal PDF approximation* is obtained, which is

$$\Pr(S = s) = \frac{1}{\sqrt{2\pi}\sigma} e^{-(s-\mu)^2/2\sigma^2}.$$

Instead of settling for approximations of the probabilities, we will show how APPL procedures, including **Convolution**, can retrieve exact solutions, while still giving the same insight into the problem.

Example 4.9. (Thompson, 2000, page 53) Suppose that X_1, X_2, \dots, X_{20} are independent, $X_i \sim \text{Bernoulli}(p_i = \frac{29+2i}{100})$, $i = 1, 2, \dots, 20$. Let $S = \sum_{i=1}^{20} X_i$. (Here S denotes the total number of successes obtained in a series of independent trials where the probability of success varies from trial to trial.) Give an exact probability table for S for $s = 2, 3, \dots, 10$.

Solution: Using Thompson's notation, $\mu = 10$ and $\sigma^2 = 2367/500$, and so $\Pr(S = s) \cong \frac{1}{\sqrt{2\pi}\sqrt{2367/500}} e^{-(s-10)^2/(2367/250)}$, $s = 0, 1, \dots, 20$. Using APPL, we obtain the exact distribution of S with the statements

```
> p := (29 + 2 * 1) / 100;
> S := BernoulliRV(p);
> for i from 2 to 20 do
>   p := (29 + 2 * i) / 100:
>   X := BernoulliRV(p):
>   S := Convolution(S, X):
> od:
```

> S;

Table 4.2 contains the exact probabilities found with APPL and the normal PDF approximations for $s = 2, 3, \dots, 10$. □

Table 4.2: Exact probabilities and normal PDF approximations of $\Pr(S = s)$ for $s = 2, 3, \dots, 10$.

s	True $\Pr(S = s)$ (APPL)	Approximation of true $\Pr(S = s)$	Normal PDF Approximation
2	$\frac{204658765144989225788930713729011}{1600000000000000000000000000000000}$	0.0001	0.0002
3	$\frac{670581044381861117271962962043967}{8000000000000000000000000000000000}$	0.0008	0.0010
4	$\frac{12306309890051216090420607156481161}{3200000000000000000000000000000000}$	0.0038	0.0041
5	$\frac{13130118961411820609429234497062639}{10000000000000000000000000000000000}$	0.0131	0.0131
6	$\frac{13845545992556016094922419904605161}{4000000000000000000000000000000000}$	0.0346	0.0338
7	$\frac{14429186684261724023997491367619439}{2000000000000000000000000000000000}$	0.0721	0.0709
8	$\frac{193196528593089153025093245904930293}{1600000000000000000000000000000000}$	0.1207	0.1202
9	$\frac{65549414450257125600014354447607969}{4000000000000000000000000000000000}$	0.1639	0.1650
10	$\frac{725313008476889512417635294011302541}{4000000000000000000000000000000000}$	0.1813	0.1834

Example 4.10. (Thompson, 2000, pages 53–54) There are 20 girls and 30 boys in Group 1, 25 girls and 25 boys in Group 2, and 10 girls and 10 boys in Group 3. If 10 children are randomly chosen from each group and S denotes the total number of girls chosen, give an exact probability table for S for $s = 7, 8, \dots, 21$.

Solution: Let $X_1, X_2,$ and X_3 be the three independent hypergeometric random variables, and let $S = X_1 + X_2 + X_3$. The mean and variance of S are $\mu = E[S] = E[X_1] + E[X_2] + E[X_3] = 14$ and $\sigma^2 = \text{Var}(S) = \text{Var}(X_1) + \text{Var}(X_2) + \text{Var}(X_3) = 101/19$ (since $X_1, X_2,$ and X_3 are independent). Table 4.3 shows the normal PDF approximation values $\Pr(S = s) \cong \frac{1}{\sqrt{2\pi}\sqrt{101/19}}e^{-(s-14)^2/(202/19)}$ for $s = 7, 8, \dots, 21$.

Using the APPL Convolution procedure, we can calculate the PDF of S with the statements

```
> X1 := HypergeometricRV(50, 20, 10);
> X2 := HypergeometricRV(50, 25, 10);
> X3 := HypergeometricRV(20, 10, 10);
> Y := Convolution(X3, Convolution(X1, X2));
```

The exact values for $s = 7, 8, \dots, 21$ are shown in Table 4.3. \square

Table 4.3: The exact probabilities and normal PDF approximations for $\Pr(S = s)$ for $s = 7, 8, \dots, 21$.

s	True $\Pr(S = s)$ (APPL)	Approximation of true $\Pr(S = s)$	Normal PDF Approximation
7	$\frac{4641594894759547665}{3082276280132202064912}$	0.0015	0.0017
8	$\frac{97479371530863990}{17512933409842057187}$	0.0056	0.0059
9	$\frac{12613791756912076515}{770569070033050516228}$	0.0164	0.0165
10	$\frac{74849525260411094591}{1926422675082626290570}$	0.0389	0.0384
11	$\frac{57967137494173367365}{770569070033050516228}$	0.0752	0.0742
12	$\frac{2096975232909133615}{17512933409842057187}$	0.1197	0.1188
13	$\frac{22076335771392253895}{140103467278736457496}$	0.1576	0.1575
14	$\frac{317244095646532855}{1843466674720216546}$	0.1721	0.1730
15	$\frac{9955623438355053449}{63683394217607480680}$	0.1563	0.1575
16	$\frac{217921905682010165}{1843466674720216546}$	0.1182	0.1188
17	$\frac{1894259194489549345}{25473357687042992272}$	0.0744	0.0742
18	$\frac{71588441634588035}{1843466674720216546}$	0.0388	0.0384
19	$\frac{10756216836381565}{641205799902684016}$	0.0168	0.0165
20	$\frac{1208983087163529637}{202781334219223820060}$	0.0060	0.0059
21	$\frac{280730797358534065}{162225067375379056048}$	0.0017	0.0017

Example 4.11. (Grinstead & Snell, 1997, pages 290–291) Assume that $r > 2$ is a non-prime integer. Show that there are non-trivial distributions for X and Y on

the nonnegative integers such that the convolution of X and Y is the equiprobable distribution on the set $0, 1, 2, \dots, r - 1$.

Solution: Let n be the smallest prime factor of a given non-prime integer $r > 2$.

The random variables X and Y can have PDFs

$$f_X(x) = \begin{cases} \frac{1}{n} & x = 0 \\ \frac{1}{n} & x = 1 \\ \frac{1}{n} & x = 2 \\ \vdots & \vdots \\ \frac{1}{n} & x = n - 1, \end{cases} \quad f_Y(y) = \begin{cases} \frac{n}{r} & y = 0 \\ \frac{n}{r} & y = n \\ \frac{n}{r} & y = 2 \cdot n \\ \vdots & \vdots \\ \frac{n}{r} & y = \left(\frac{r}{n} - 1\right) \cdot n. \end{cases}$$

Loading the `factorset` procedure from Maple's number theory package, we test our conjecture for $r = 12$, for instance, with the statements

```
> with(numtheory, factorset):
> r := 12;
> n := min(op(factorset(r)));
> X := [[seq(1 / n, i = 0 .. n - 1)], [seq(j, j = 0 .. n - 1)],
        ["Discrete", "PDF"]];
> Y := [[seq(n / r, i = 0 .. n / r - 1)],
        [seq(j * n, j = 0 .. r / n - 1)], ["Discrete", "PDF"]];
> Z := Convolution(X, Y);
```

The PDF of Z is returned in its list-of-sublists APPL format as the discrete equiprobable distribution on the set $0, 1, 2, \dots, 11$. \square

APPL can also handle some convolutions of discrete random variables with infinite support. The PDF of the convolution of the random variables needs to be tractable so Maple can simplify the resulting sum.

Example 4.12. (Larsen & Marx, 1986, page 220) A *wet spell of x days* is defined to be a “run” of x days on each of which measurable precipitation occurs. Under the assumption that the weather tomorrow depends only on the weather today, there is a

probability p_0 that a wet day will be followed by a dry day. From this, the probability of an x -day-long wet spell is $p_0(1 - p_0)^{x-1} = p_0q_0^{x-1}$, $x = 1, 2, \dots$, which is a geometric distribution. Let X denote the random variable representing the lengths of the wet spells.

Similarly, the random variable Y is the length of *dry* spells; furthermore, $f_Y(y) = p_1q_1^{y-1}$, $y = 1, 2, \dots$, where p_1 is the probability of a dry day followed by a wet one. Since a given day's weather is affected only by the previous day's, it follows that X and Y are independent.

Now, a *weather cycle* will be defined as, say, a wet spell followed by a dry spell. If Z denotes the length of such a cycle, then $Z = X + Y$. Find the PDF of Z .

Solution: The APPL statements

```
> X := [[x -> p0 * q0 ^ (x - 1)], [1 .. infinity],
        ["Discrete", "PDF"]];
> Y := [[x -> p1 * q1 ^ (x - 1)], [1 .. infinity],
        ["Discrete", "PDF"]];
> Z := Convolution(X, Y);
```

yield the PDF in its APPL list-of-sublists form as

$$\left[\left[z \rightarrow \frac{p_1 p_0 \left(q_1^{z-1} \left(\frac{q_0}{q_1} \right)^z q_0 - q_1^z \right)}{q_0 (q_0 - q_1)} \right], [2 .. \infty], ["Discrete", "PDF"] \right]. \quad \square$$

4.5 Products of Random Variables with Finite Supports

The algorithm that was constructed (with heaps) to compute the PDF of the convolution was extended to determine the PDF of the product. Because of possible negative, zero, and positive support values for the random variables X and Y , the product algorithm “splits” the conceptual array A into four quadrants (as in the

Cartesian coordinate system) before heaping. The quadrants are conceptually split based on X 's and Y 's negative and nonnegative support values. Figure 4.12 illustrates the evolution of the algorithm in the most general case when both X and Y have negative and nonnegative support values. The product algorithm conceptually starts at the northwest and southeast corners of quadrants two and four and moves toward the center where the quadrants meet, considers the case where one or both of the supports include zero, and then works outward from the center through quadrants one and three.

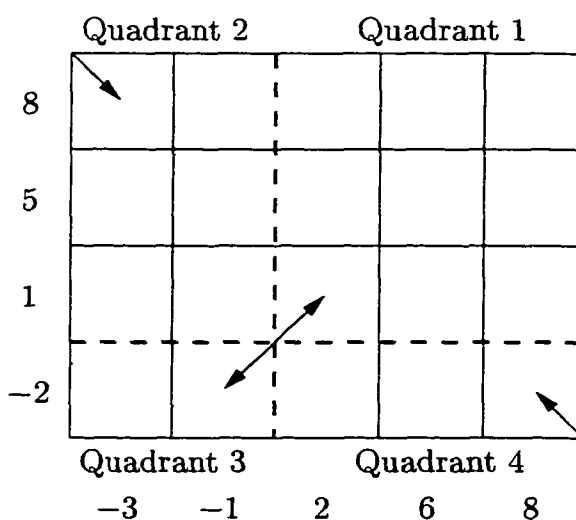


Figure 4.12: Array A split into four quadrants for the product algorithm.

To compute the product for two random variables X and Y with finite supports $\Omega_X = \{x_1, x_2, \dots, x_n\}$, where $x_1 < x_2 < \dots < x_n$, and $\Omega_Y = \{y_1, y_2, \dots, y_m\}$, where $y_1 < y_2 < \dots < y_m$, the `MovingHeapProductMethod` was developed. It basically mimics the logic of the algorithm in the `MovingHeapMethod` (for convolutions), except it determines increasing product values and their probabilities. In order to take advantage of the moving heap method (as constructed for convolutions), the support sublists of the random variables X and Y are manipulated (and sometimes split)

to produce increasing product values as the algorithm moves northeasterly from the southwest corner of A to its northeast corner. When thinking in terms of the Cartesian coordinate system, there are nine subcases that need to be considered when constructing the conceptual array A . The subcases are based on the signs of x_1 , x_n , y_1 , and y_m . Details of the nine subcases follow and examples with APPL solutions illustrate some of these subcases.

1. If $x_1 \geq 0$ and $y_1 \geq 0$, then the support of the product of X and Y contains zero and/or positive values. For $i = 1, 2, \dots, n$, each x_i support value is listed in increasing order (i.e., x_1, x_2, \dots, x_n) from left to right along the bottom of the array A . Similarly for $j = 1, 2, \dots, m$, each y_j support value is listed in increasing order (i.e., y_1, y_2, \dots, y_m) from bottom to top along the left side of the array A . The `MovingHeapProductMethod` procedure uses the same method as `MovingHeapMethod` for convolutions—it starts computing products in the lower-left corner of A and moves northeasterly computing increasing product values until it reaches the upper-right corner of A . Since $0 \leq x_1 < x_2 < \dots < x_n$ and $0 \leq y_1 < y_2 < \dots < y_m$, then $x_{i+1} \cdot y_{j+1} \geq x_i \cdot y_j$, $x_{i+1} \cdot y_j \geq x_i \cdot y_j$, and $x_i \cdot y_{j+1} \geq x_i \cdot y_j$ for $i = 1, 2, \dots, n - 1$ and $j = 1, 2, \dots, m - 1$. Thus, we are guaranteed that moving northeasterly within the array A produces increasing product values.

Example 4.13. A spinner yields three equally likely outcomes: 1, 2, 3. If the random variable Z denotes the product of the outcomes of the two spins, compute $\Pr(Z \leq 6)$.

Solution: The APPL statements

```
> X := UniformDiscreteRV(1, 3);
> Y := UniformDiscreteRV(1, 3);
> Z := Product(X, Y);
> prob := CDF(Z, 6);
```

return the probability of 8/9. □

2. If $x_n < 0$ and $y_m < 0$, then the support of the product of X and Y contains positive values. For $i = 1, 2, \dots, n$, each x_i support value is listed in decreasing order (i.e., x_n, x_{n-1}, \dots, x_1) from left to right along the bottom of the array A . Similarly for $j = 1, 2, \dots, m$, each y_j support value is listed in decreasing order (i.e., y_m, y_{m-1}, \dots, y_1) from bottom to top along the left side of the array A . The `MovingHeapProductMethod` procedure again computes products starting in the lower-left corner of A and moves northeasterly computing increasing product values until it reaches the upper-right corner of A . Since $x_1 < x_2 < \dots < x_n < 0$ and $y_1 < y_2 < \dots < y_m < 0$, then $x_{i-1} \cdot y_{j-1} > x_i \cdot y_j$, $x_{i-1} \cdot y_j > x_i \cdot y_j$, and $x_i \cdot y_{j-1} > x_i \cdot y_j$ for $i = 2, \dots, n$ and $j = 2, \dots, m$. Moving northeasterly within the array A again produces increasing product values. The following APPL implementation example and accompanying figure illustrates this subcase.

To determine the PDF of the product in APPL, the order of the elements in the first and second sublists of the random variables X and Y is reversed. Then X and Y have their product determined by the `MovingHeapProductMethod` procedure with their first two sublists in this new order.

For example, if X and Y are represented in APPL as

$$X = [[0.2, 0.1, 0.3, 0.4], [-6, -3, -2, -1], ["Discrete", "PDF"]]$$

and

$$Y = [[0.3, 0.3, 0.4], [-5, -3, -2], ["Discrete", "PDF"]],$$

then they are rewritten as

$$X = [[0.4, 0.3, 0.1, 0.2], [-1, -2, -3, -6], ["Discrete", "PDF"]]$$

and

$$Y = [[0.4, 0.3, 0.3], [-2, -3, -5], ["Discrete", "PDF"]]$$

before entering the `MovingHeapProductMethod` procedure. Using the *newly ordered sublists*, `MovingHeapProductMethod` determines the correct product $Z = X \cdot Y$, namely

$$Z = [[0.16, 0.12, 0.12, 0.12, 0.13, 0.03, 0.09, 0.08, 0.03, 0.06, 0.06], \\ [2, 3, 4, 5, 6, 9, 10, 12, 15, 18, 30], ["Discrete", "PDF"]].$$

Figure 4.13 illustrates the increasing product values for $i = 3$ and $j = 2$ as one moves northeasterly in the array A .

$y_1 - 5$		10	15	
$y_2 - 3$		6	9	
$y_3 - 2$				
	-1	-2	-3	-6
	x_4	x_3	x_2	x_1

Figure 4.13: Product array A for subcase two. For $i = 2, j = 3$, $x_2 \cdot y_1 > x_3 \cdot y_2$, $x_2 \cdot y_2 > x_3 \cdot y_2$, and $x_3 \cdot y_1 > x_3 \cdot y_2$, i.e., cells to the northeast of the cell with product $x_3 \cdot y_2 = 6$ have larger product values.

3. If $x_1 \geq 0$ and $y_m < 0$, then the support of the product of X and Y contains zero and/or negative values. For $i = 1, 2, \dots, n$, each x_i support value is listed in decreasing order (i.e., x_n, x_{n-1}, \dots, x_1) from left to right along the bottom of the array A . For $j = 1, 2, \dots, m$, each y_j support value is listed in increasing order (i.e., y_1, y_2, \dots, y_m) from bottom to top along the left side of the array A . The `MovingHeapProductMethod` procedure computes products starting in the lower-left corner of A and moves northeasterly computing increasing product values until it reaches the upper-right corner of A . Since $0 \leq x_1 < x_2 < \dots < x_n$ and $y_1 < y_2 < \dots < y_m < 0$, then $x_{i-1} \cdot y_{j+1} \geq x_i \cdot y_j$, $x_{i-1} \cdot y_j \geq x_i \cdot y_j$, and

$x_i \cdot y_{j+1} \geq x_i \cdot y_j$ for $i = 2, \dots, n$ and $j = 1, 2, \dots, m - 1$. Thus, moving northeasterly within the array A produces increasing product values.

To determine the PDF of the product in APPL, the order of the elements in the first and second sublists of only the random variable X is reversed. Then X and Y have their product determined by `MovingHeapProductMethod` with X 's first two sublists in this new order.

For example, if X and Y are represented in APPL as

$$X = [[0.2, 0.1, 0.3, 0.4], [1, 2, 4, 6], ["Discrete", "PDF"]],$$

and

$$Y = [[0.3, 0.3, 0.4], [-5, -3, -2], ["Discrete", "PDF"]],$$

then just X is rewritten as

$$X = [[0.4, 0.3, 0.1, 0.2], [6, 4, 2, 1], ["Discrete", "PDF"]]$$

before entering the `MovingHeapProductMethod` procedure. Using the newly ordered sublists of X and the original sublists of Y , `MovingHeapProductMethod` determines the product correctly. The PDF of $Z = X \cdot Y$ is

$$\begin{aligned} Z = & [[0.12, 0.09, 0.12, 0.25, 0.03, 0.12, 0.03, 0.06, 0.04, 0.06, 0.08], \\ & [-30, -20, -18, -12, -10, -8, -6, -5, -4, -3, -2], \\ & ["Discrete", "PDF"]]. \end{aligned}$$

Figure 4.14 illustrates the increasing product values for $i = 2$ and $j = 2$ as one moves northeasterly in the array A .

4. If $x_n < 0$ and $y_1 \geq 0$, then the support of the product of X and Y contains zero and/or negative values. Proceed as in subcase three, except reverse only the sublists for the random variable Y in APPL.
5. If $x_1 < 0$, $x_n \geq 0$, and $y_1 \geq 0$, then the support of the product of X and Y contains both negative and nonnegative values. For example, let X and Y be

$y_3 -2$			-4	-2
$y_2 -3$			-6	-3
$y_1 -5$				
	6	4	2	1
	x_4	x_3	x_2	x_1

Figure 4.14: Product array A for subcase three. For $i = 2, j = 2$, $x_2 \cdot y_3 > x_2 \cdot y_2$, $x_1 \cdot y_2 > x_2 \cdot y_2$, and $x_1 \cdot y_3 > x_2 \cdot y_2$, i.e., cells to the northeast of the cell with product $x_2 \cdot y_2 = -6$ have larger product values.

random variables with PDFs

$$f_X(x) = \begin{cases} 0.2 & x = -3 \\ 0.1 & x = -2 \\ 0.3 & x = 1 \\ 0.4 & x = 2, \end{cases} \quad f_Y(y) = \begin{cases} 0.3 & y = 2 \\ 0.3 & y = 3 \\ 0.4 & y = 5. \end{cases}$$

If $Z = X \cdot Y$, then the PDF of Z is

$$f_Z(z) = \begin{cases} 0.08 & z = -15 \\ 0.04 & z = -10 \\ 0.06 & z = -9 \\ 0.09 & z = -6 \\ 0.03 & z = -4 \\ 0.09 & z = 2 \\ 0.09 & z = 3 \\ 0.12 & z = 4 \\ 0.12 & z = 5 \\ 0.12 & z = 6 \\ 0.16 & z = 10. \end{cases}$$

The random variable X is split into two separate lists. The negative support values and their corresponding probabilities are placed in the variable X_- , while

the nonnegative support values and their corresponding probabilities are placed in the variable X_+ . Since $y_j \geq 0$ for $j = 1, 2, \dots, m$ and $x_i < 0$ for each $x_i \in \Omega_{X_-}$, then we can use the `MovingHeapProductMethod` procedure on X_- and Y first, proceeding as in subcase four. Since $y_j \geq 0$ for $j = 1, 2, \dots, m$ and $x_i \geq 0$ for each $x_i \in \Omega_{X_+}$, then we can use the `MovingHeapProductMethod` procedure on X_+ and Y next, proceeding as in subcase one.

For the given example, the two sublists of X_- and X_+ are

$$X_- = [[0.2, 0.1], [-3, -2]] \quad \text{and} \quad X_+ = [[0.4, 0.3], [1, 2]].$$

The two conceptual arrays, A_- and A_+ , for computing the increasing products from the lower-left to the upper-right corners of the arrays are displayed in Figure 4.15.

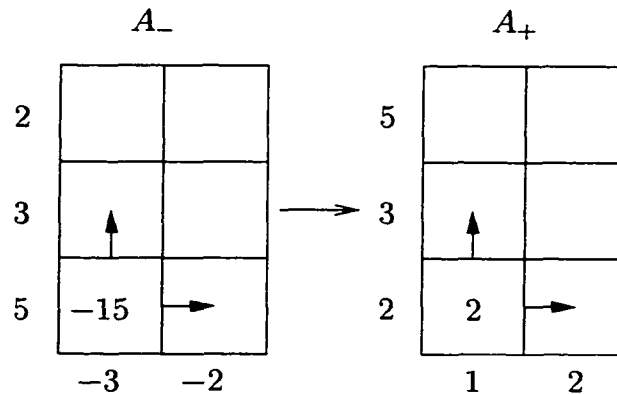


Figure 4.15: Product arrays A_- and A_+ for subcase five.

6. If $x_1 < 0$, $x_n \geq 0$, and $y_m < 0$, then the support of the product of X and Y contains both negative and nonnegative values. Again, the random variable X is split into two separate lists. The negative support values and their corresponding probabilities are placed in the variable X_- , while the nonnegative support values and their corresponding probabilities are placed in the variable X_+ . Since $y_j < 0$ for $j = 1, 2, \dots, m$ and $x_i \geq 0$ for each $x_i \in \Omega_{X_+}$ (which results in negative and/or zero products), then we can use the `MovingHeapProductMethod`

procedure on X_+ and Y first, proceeding as in subcase three. Since $y_j < 0$ for $j = 1, 2, \dots, m$ and $x_i < 0$ for each $x_i \in \Omega_{X_-}$ (which will result in positive products), then we can use the `MovingHeapProductMethod` procedure on X_- and Y next, proceeding as in subcase two.

Example 4.14 Let X have a uniform discrete distribution between -4 and 1 , and let Y have a uniform discrete distribution between 1 and 6 . Use APPL to compute the PDF of their product.

Solution: In APPL, the statements

```
> X := UniformDiscreteRV(-4, 1);
> Y := UniformDiscreteRV(1, 6);
> Z := Product(X, Y);
```

return the product Z as

$$Z = \left[\left[\begin{array}{cccccccccccccccccccc} \frac{1}{36}, \frac{1}{36}, \frac{1}{36}, \frac{1}{36}, \frac{1}{36}, \frac{1}{12}, \frac{1}{36}, \frac{1}{36}, \frac{1}{18}, \frac{1}{12}, \frac{1}{36}, \frac{1}{12}, \frac{1}{18}, \frac{1}{18}, \frac{1}{36}, \frac{1}{6}, \frac{1}{36}, \frac{1}{36}, \frac{1}{36}, \frac{1}{36}, \frac{1}{36}, \frac{1}{36} \end{array} \right], \right. \\ \left. \begin{array}{cccccccccccccccc} -24, -20, -18, -16, -15, -12, -10, -9, -8, -6, -5, -4, -3, -2, -1, 0, 1, 2, 3, 4, 5, 6 \end{array}, \right. \\ \left. \text{["Discrete", "PDF"]} \right]. \quad \square$$

7. If $x_1 \geq 0$, $y_1 < 0$, and $y_m \geq 0$, then the support of the product of X and Y contains both negative and nonnegative values. In this case, the random variable Y is split into two separate lists. The negative support values and their corresponding probabilities are placed in the variable Y_- , while the nonnegative support values and their corresponding probabilities are placed in the variable Y_+ . Since $x_i \geq 0$ for $i = 1, 2, \dots, n$ and $y_j < 0$ for each $y_j \in \Omega_{Y_-}$ (which results in negative and/or zero products), then we can use the `MovingHeapProductMethod` procedure on X and Y_- first, proceeding as in subcase three. Since $x_i \geq 0$ for $i = 1, 2, \dots, n$ and $y_j \geq 0$ for each $y_j \in \Omega_{Y_+}$ (which results in positive and/or zero products), then we can use the `MovingHeapProductMethod` procedure on X and Y_+ next, proceeding as in subcase one.

8. If $x_n < 0$, $y_1 < 0$, and $y_m \geq 0$, then the support of the product of X and Y contains both negative and nonnegative values. The random variable Y is again split into two separate lists. The negative support values and their corresponding probabilities are placed in the variable Y_- , while the nonnegative support values and their corresponding probabilities are placed in the variable Y_+ . Since $x_i < 0$ for $i = 1, 2, \dots, n$ and $y_j \geq 0$ for each $y_j \in \Omega_{Y_+}$, then we can use the `MovingHeapProductMethod` procedure on X and Y_+ first, proceeding as in subcase four. Since $x_i < 0$ for $i = 1, 2, \dots, n$ and $y_j < 0$ for each $y_j \in \Omega_{Y_-}$, then we can use the `MovingHeapProductMethod` procedure on X and Y_- next, proceeding as in subcase two.
9. If $x_1 < 0$, $x_n \geq 0$, $y_1 < 0$, and $y_m \geq 0$, then the support of the product of X and Y contains both negative and nonnegative values. This is the most difficult case, displayed pictorially in Figure 4.12. The random variables X and Y are both split into two separate lists, X_- , X_+ , and Y_- , Y_+ , respectively. In subcase nine, there are “dueling heaps,” and we use the `APPL MovingHeapDuelMethod` procedure.

The products of the support values of the variables X_+ and Y_- are negative, as well as the products of the support values for X_- and Y_+ . All four variables are sent to the `MovingHeapDuelMethod` so their support products can be computed in increasing order. The array A_{1-} is used for X_+ and Y_- , and the products of their support values are computed according to the method in subcase three. The array A_{2-} is used for X_- and Y_+ , and the products of their support values are computed according to the method in subcase four. As the next largest products arise in A_{1-} and A_{2-} , they “duel” each other as the larger product. If the product in A_{1-} is larger, for example, it is selected as the next largest product and the `MovingHeapDuelMethod` continues to move northeast-

erly through A_{1-} . The next largest value in A_{1-} then challenges this same A_{2-} value as the next largest product. This process continues until the northeast corners of both arrays are reached and their values are recorded in increasing order.

The products of the support values of the variables X_- and Y_- are positive, and the products of the support values for X_+ and Y_+ are nonnegative. All four variables are sent to the `MovingHeapDuelMethod` so their support products can be computed in increasing order. The array A_{1+} is used for X_- and Y_- , and the products of their support values are computed according to the method in subcase two. The array A_{2+} is used for X_+ and Y_+ , and the products of their support values are computed according to the method in subcase one. As the next largest products arise in A_{1+} and A_{2+} , they “duel” each other as the larger product. If the product in A_{2+} is larger, for example, it is selected as the next largest product and the `MovingHeapDuelMethod` continues to move northeasterly through A_{2+} . The next largest value in A_{2+} then challenges this same A_{1+} value as the next largest product. This process continues until the northeast corners of both arrays are reached and their values are recorded in increasing order.

In order to visualize how subcase nine works, let X and Y be the random variables indicated in Figure 4.12. Their first sublists are $[-3, -1, 2, 6, 8]$ and $[-2, 1, 5, 8]$. Figure 4.16 shows the conceptual arrays A_{1-} , A_{2-} , A_{1+} , and A_{2+} for X and Y . The arrows in the figure indicate the direction the products will be computed (in order to obtain larger product values). Figure 4.17 displays the first three products of X and Y as the `MovingHeapDuelMethod` progresses simultaneously through A_{1-} and A_{2-} .

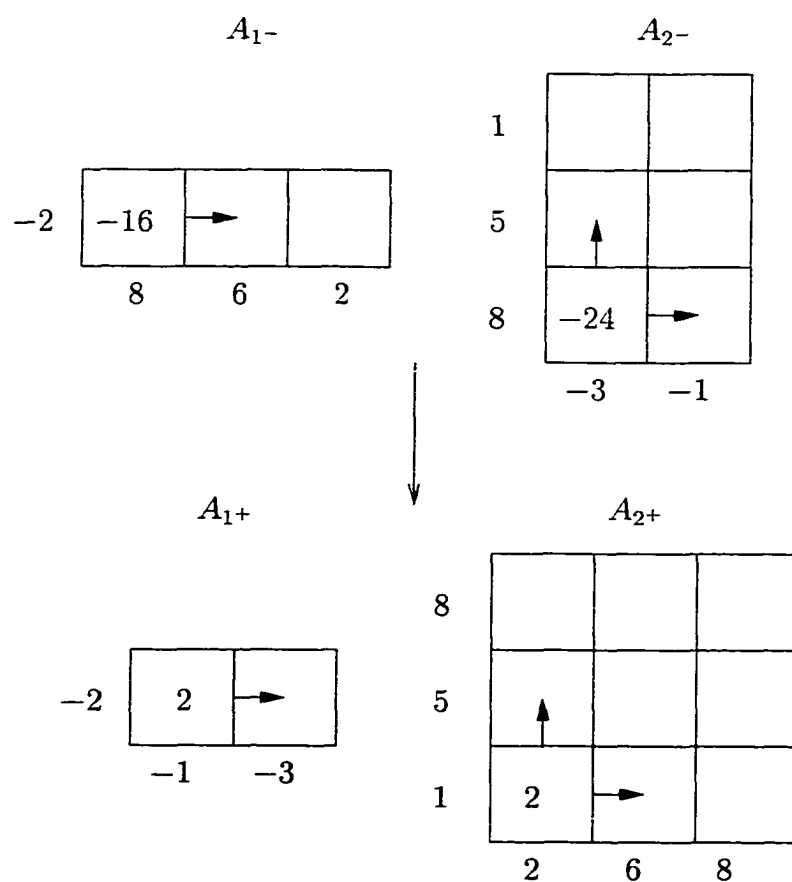


Figure 4.16: Dueling product arrays A_{1-} and A_{2-} , and A_{1+} and A_{2+} for subcase nine.

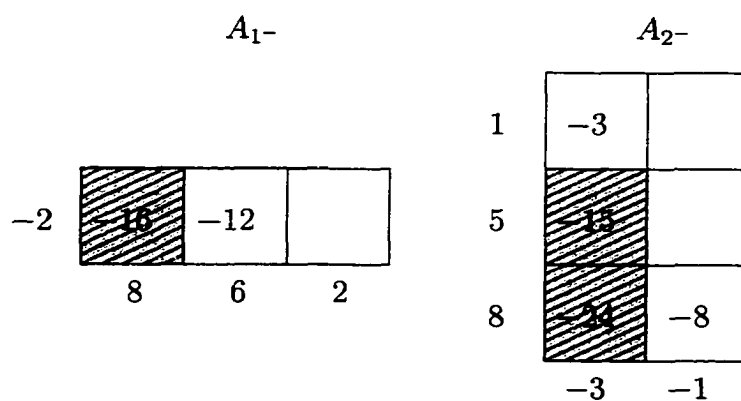


Figure 4.17: The **MovingHeapDuelMethod** as it progresses simultaneously through arrays A_{1-} and A_{2-} for subcase nine. The first three elements of the product support list are -24 , -16 , and -15 .

Chapter 5

Transformations

A method of finding the distribution of a function of one or more random variables is called the *change-of-variable* technique. The change-of-variable technique can be used to determine the distribution of a discrete random variable $Y = g(X)$ given the distribution of the discrete random variable X and a one-to-one transformation g from the support of X , Ω_X , to the support of Y , Ω_Y . Further, the transformation may be “piecewise many-to-one,” as presented in Bain and Engelhardt (1992, page 203). A “piecewise many-to-one” transformation denotes a transformation that is either one-to-one, two-to-one, three-to-one, etc. (i.e., many-to-one) on disjoint subsets (“pieces”) of Ω_X . For example, if X is a uniform discrete random variable for $x = -2, -1, \dots, 3$, then $Y = |X|$ is a two-to-one transformation on the subset $\{-2, -1, 1, 2\}$ and a one-to-one transformation on the subsets $\{0\}$ and $\{3\}$. Glen et al. (1997) presents a generalized version of the univariate change-of-variable technique for transforming *continuous* random variables. The purpose of this chapter is to extend their technique to *discrete* random variables for both one-to-one and piecewise many-to-one transformations.

5.1 Theory

5.1.1 One-to-One Transformations

Theorem 5.1. (Bain & Engelhardt, 1992, page 197) Suppose that X is a discrete random variable with PDF $f_X(x)$ and $Y = g(X)$ defines a one-to-one transformation from Ω_X to Ω_Y , i.e., $y = g(x)$ can be solved uniquely, say $x = g^{-1}(y)$. Then the PDF of Y is

$$f_Y(y) = f_X(g^{-1}(y)) \quad y \in \Omega_Y$$

where $\Omega_Y = \{y \mid f_Y(y) > 0\}$.

Proof: By substitution,

$$f_Y(y) = \Pr(Y = y) = \Pr(g(X) = y) = \Pr(X = g^{-1}(y)) = f_X(g^{-1}(y)). \quad \square$$

The following two examples show how Theorem 5.1 is applied to discrete transformation problems.

Example 5.1. (Miller & Miller, 1999, page 242) If X is the number of heads obtained in four tosses of a fair coin, find the PDF of $Y = \frac{1}{X+1}$.

Solution: The random variable X is binomial with parameters $n = 4$ and $p = 1/2$, and support $\Omega_X = \{0, 1, 2, 3, 4\}$. Let $Y = g(X) = \frac{1}{X+1}$, which defines a one-to-one transformation from Ω_X to $\Omega_Y = \{\frac{1}{5}, \frac{1}{4}, \frac{1}{3}, \frac{1}{2}, 1\}$. By Theorem 5.1, the PDF of Y is

$$f_Y(y) = \Pr\left(\frac{1}{1+X} = y\right) = \Pr\left(X = \frac{1-y}{y}\right) = f_X\left(\frac{1-y}{y}\right) = \binom{4}{\frac{1-y}{y}} \left(\frac{1}{2}\right)^4,$$

for $y = \frac{1}{5}, \frac{1}{4}, \frac{1}{3}, \frac{1}{2}, 1$. □

Example 5.2. (Hogg & Craig, 1995, page 51) Let $f(x) = x/6$, $x = 1, 2, 3$, zero elsewhere, be the PDF of X . Find the CDF of $Y = X^2$.

Solution: The function $Y = g(X) = X^2$ defines a one-to-one transformation in this example since Ω_X contains strictly positive values. By Theorem 5.1, the PDF of Y is

$$f_Y(y) = \Pr(Y = y) = \Pr(X^2 = y) = \Pr(X = \sqrt{y}) = \frac{\sqrt{y}}{6},$$

for $y = 1, 4, 9$. Thus, $F_Y(y) = \sum_{i=1}^y \frac{\sqrt{i}}{6}$ for $y = 1, 4, 9$. □

5.1.2 “Piecewise Many-to-One” Transformations

If the function $Y = g(X)$ is piecewise many-to-one on Ω_X , then there is no unique solution to the equation $Y = g(X)$ on Ω_X . Bain and Engelhardt (1992, page 202) suggest partitioning Ω_X into disjoint subsets $\Omega_{X_1}, \Omega_{X_2}, \dots$ such that $Y = g(X)$ is one-to-one over each Ω_{X_i} . Then for each $y \in \{g(x) \mid x \in \Omega_X\}$, the equation $y = g(x)$ has a unique solution $x_i = g_i^{-1}(y)$ on the subset Ω_{X_i} . Hence, Theorem 5.1 can be extended to functions that are piecewise many-to-one by replacing $f_Y(y) = f_X(g^{-1}(y))$ with

$$f_Y(y) = \sum_i f_X(g_i^{-1}(y)).$$

The following example shows how Theorem 5.1 is extended to cover problems where $Y = g(X)$ is a piecewise many-to-one transformation.

Example 5.3. (Miller & Miller, 1999, page 243) If X is again the number of heads obtained in four tosses of a fair coin, find the PDF of the random variable $Y = (X - 2)^2$.

Solution: The transformation $Y = g(X) = (X - 2)^2$ is a two-to-one transformation for $X = 0, 1, 3, 4$ and a one-to-one transformation for $X = 2$, as can be seen in Figure 5.1. Partition Ω_X such that $\Omega_{X_1} = \{0, 1, 2\}$, and $\Omega_{X_2} = \{3, 4\}$. Then the transformation $g(X) = (X - 2)^2$ is a one-to-one mapping of Ω_{X_1} and Ω_{X_2} into $\Omega_Y =$

$\{0, 1, 4\}$. Since $g_1^{-1}(y) = -\sqrt{y} + 2$ and $g_2^{-1}(y) = \sqrt{y} + 2$, then

$$f_Y(0) = f_X(-\sqrt{0} + 2) = f_X(2) = \binom{4}{2} \left(\frac{1}{2}\right)^4 = \frac{6}{16},$$

$$f_Y(1) = f_X(-\sqrt{1} + 2) + f_X(\sqrt{1} + 2) = f_X(1) + f_X(3) = 2 \cdot \binom{4}{1} \left(\frac{1}{2}\right)^4 = \frac{8}{16}, \quad \text{and}$$

$$f_Y(4) = f_X(-\sqrt{4} + 2) + f_X(\sqrt{4} + 2) = f_X(0) + f_X(4) = 2 \cdot \binom{4}{0} \left(\frac{1}{2}\right)^4 = \frac{2}{16}.$$

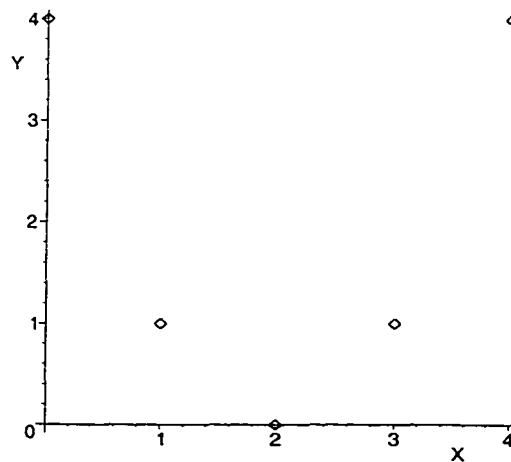


Figure 5.1: The transformation $Y = g(X) = (X - 2)^2$ for $X = 0, 1, 2, 3, 4$. The transformation is two-to-one for $X = 0, 1, 3, 4$ and a one-to-one for $X = 2$.

Another way to write the PDF of Y is

$$f_Y(y) = \frac{3}{8} \quad y = 0, \quad \text{and}$$

$$f_Y(y) = 2 \cdot \binom{4}{\sqrt{y} + 2} \left(\frac{1}{2}\right)^4 \quad y = 1, 4.$$

5.2 Implementation

Glen et al. (1997) provide a theorem and resulting computer algebra system implementation for “determining the PDF of $Y = g(X)$ for any univariate random variable X of the continuous type with few restrictions on the transformation $g(X)$.” The algorithm discussed in this section for determining the PDF of $Y = g(X)$ for any univariate random variable X of the *discrete* type is modeled after their theorem and implementation. Naturally, there are differences in the implementation of their algorithm and the algorithm for discrete random variables, especially since Theorem 5.1 does not hold as stated for continuous random variables. But many issues, such as Maple producing more than one inverse function $g_i^{-1}(y)$ (e.g., if $g_i(x) = x^2$) or partitioning Ω_X into disjoint subsets so that $Y = g(X)$ is one-to-one on each of them, is a problem for both continuous random variables and discrete random variables.

As with all procedures that operate on discrete random variables, the **Transform** procedure treats *NoDot* and *Dot* formatted random variables separately. It is not difficult to transform a random variable X with a *NoDot* data structure format into a new random variable $Y = g(X)$ since the PDF and support of X are lists. The transformation $g(x)$, whether one-to-one, piecewise many-to-one, or even discontinuous, is applied by brute force to each element in X 's support list. Although it is necessary to determine which “piece” of $g(x)$ corresponds to the various support values in Ω_X , we do not need to partition Ω_X since identical support values in $Y = g(X)$ can be combined before the transformed random variable is returned. Thus, the difficult task of partitioning Ω_X into subsets Ω_{X_i} and determining the unique solution $x = g_i^{-1}(y)$ for each transformation $y = g_i(x)$ on Ω_{X_i} is not required in the *NoDot* case.

The implementation for random variables with a *Dot* data structure format posed several difficult issues, one in particular with the APPL discrete list-of-sublists data structure. The implementation (and implementation issues) for both data structure

formats, *NoDot* and *Dot*, are discussed in more detail in the subsections entitled “*NoDot* Format” and “*Dot* Format.”

The APPL `Transform(X, g)` procedure requires two arguments: a random variable X (either continuous or discrete) and a transformation function g . The transformation function $g(x)$ is provided as a list of two sublists, where the first sublist contains the transformation function (or functions) and the second sublist specifies the domain of $g(x)$ by either

- listing its piecewise components and the endpoints of the corresponding domains for random variables with *NoDot* formats; or
- listing its monotone piecewise components and the endpoints of the corresponding domains for random variables with *Dot* (or *NoDot*) formats.

For example, if $Y = g(X) = |X - 2|$ and X is a discrete random variable with a *NoDot* format, then $g(x)$ is entered in APPL as

$$[[x \rightarrow |x - 2|], [-\infty, \infty]].$$

If X has a *Dot* format (or *NoDot* format), the example $g(x)$ is entered as

$$[[x \rightarrow 2 - x, x \rightarrow x - 2], [-\infty, 2, \infty]].$$

In APPL, discrete random variables, unlike continuous ones, need direction as to how to interpret the inclusion of endpoints in the second sublist of $g(x)$. Let $g(x)$ have the form

$$[[x \rightarrow g_1(x), x \rightarrow g_2(x), \dots, x \rightarrow g_m(x)], [a_1, a_2, \dots, a_{m+1}]],$$

where a_1 and a_{m+1} may be negative or positive infinity, respectively. Then the procedure Transform assumes that $g(x)$ is defined as

$$g(x) = \begin{cases} g_1(x) & a_1 < x \leq a_2 \\ g_2(x) & a_2 < x \leq a_3 \\ \vdots & \vdots \\ g_m(x) & a_m < x \leq a_{m+1}. \end{cases}$$

Let X , for example, be a uniform discrete random variable for $x = -1, 0, \dots, 8$ in its *NoDot* format. A discontinuous transformation function, such as

$$g(X) = \begin{cases} X^2 & x < 4 \\ 8 - X & x \geq 4, \end{cases}$$

must be entered as

$$[[x \rightarrow x^2, x \rightarrow 8 - x], [-\infty, 3, \infty]]$$

in order for APPL to interpret the transformation correctly. Although this may seem like an awkward way of writing this transformation, a structure for working with discontinuous transformation functions had to be put in place to handle situations like this one. If X has a *Dot* format, $g(x)$ is entered as

$$[[x \rightarrow x^2, x \rightarrow x^2, x \rightarrow 8 - x], [-\infty, 0, 3, \infty]]$$

5.2.1 *NoDot* Format

In the APPL *NoDot* data structure format, the random variable X has each support value listed in sublist two of the list-of-sublists. Let X , for example, have PDF

$f(x) = x/15$ for $x = 1, 2, \dots, 5$. The standard *NoDot* format of X is

$$\left[\left[\frac{1}{15}, \frac{2}{15}, \frac{1}{5}, \frac{4}{15}, \frac{1}{3} \right], [1, 2, 3, 4, 5], ["Discrete" "PDF"] \right].$$

Since the support values of X are contained in sublist two, any transformation that is applied to a random variable X affects this second sublist. The support values change based on the specific transformation $g(x)$. The probability values for the random variable $Y = g(X)$ [before they are combined if $g(x)$ is not one-to-one] are the same as those for X and contained in the first sublist.

For ease and consistency in the implementation of the *NoDot* case, we assume that $g(x)$ is a piecewise many-to-one transformation function. The process for working with a piecewise many-to-one function (which includes one-to-one functions) for discrete *NoDot* random variables follows.

- The random variable X is converted to its PDF representation, if necessary.
- Since X has a *NoDot* format, when $g(x)$ is one-to-one, sublist one will not change (since the probability values for X and Y are the same). Sublist three remains or becomes ["Discrete", "PDF"].
- In the *NoDot* case, each piecewise segment of $g(x)$ is indicated in the first sublist in its list of two sublists format. The algorithm identifies which segment of the transformation function is applied to which support value or values of X . (If every monotone segment is listed in the first sublist, such as $g := [[\mathbf{x} \rightarrow -\mathbf{x}, \mathbf{x} \rightarrow \mathbf{x}], [-\text{infinity}, 0, \text{infinity}]]$ for $Y = g(X) = |X|$, the algorithm still applies the appropriate segment of $g(x)$ to the support values of X .)
- The appropriate transformation function $g_i(x)$ is applied to the partitioned support values of Ω_X that are in sublist two. (If $Y = g(X)$ is a one-to-one transfor-

mation of $\Omega_X = \{x_1, x_2, \dots, x_n\}$ into Ω_Y , for example, then each support value x_i is transformed by $g(x)$, i.e., the second sublist becomes $[g(x_1), g(x_2), \dots, g(x_n)]$.

- Y 's support values (and corresponding probability values) are sorted in increasing value to conform to the APPL discrete random variable format.
- If identical support values exist in sublist two, they are removed and their probabilities are combined. (Identical support values will occur when the transformation is not one-to-one.) If X is a uniform discrete random variable for $x = -1, 0, 1$ and $Y = X^2$, for example, then $x = -1, 1$ produce the identical support value $y = 1$.
- The transformed random variable is returned to the user in a *NoDot* standard format.

Three examples are presented to illustrate the algorithm for the *NoDot* case. In the first example, $Y = g(X)$ is a one-to-one transformation; in the second example, $Y = g(X)$ is a piecewise many-to-one transformation; and in the last example, $Y = g(X)$ is not only piecewise many-to-one, but also discontinuous.

Example 5.4. Let X have PDF $f(x) = x/15$ for $x = 1, 2, \dots, 5$ as introduced in the beginning of this subsection. Find the PDF of $Y = 2X + 1$.

Solution: The standard *NoDot* format of X is

$$\left[\left[\frac{1}{15}, \frac{2}{15}, \frac{1}{5}, \frac{4}{15}, \frac{1}{3} \right], [1, 2, 3, 4, 5], ["Discrete" "PDF"] \right].$$

Since X is in its *NoDot* format and the transformation $g(X) = 2X + 1$ is a one-to-one transformation for $\Omega_X = \{1, 2, 3, 4, 5\}$, then Ω_X is not partitioned. The only step required in determining the PDF of Y in this case is transforming the support

values of X by the one and only transformation segment $g(X) = 2X + 1$. Applying the transformation $g(X)$ to Ω_X yields the support values $\Omega_Y = \{3, 5, 7, 9, 11\}$ in increasing order. The statements

```
> X := [[x -> x / 15], [1, 2, 3, 4, 5], ["Discrete", "PDF"]];
> g := [[x -> 2 * x + 1], [-infinity, infinity]];
> Y := Transform(X, g);
```

return the PDF of Y as

$$\left[\left[\frac{1}{15}, \frac{2}{15}, \frac{1}{5}, \frac{4}{15}, \frac{1}{3} \right], [3, 5, 7, 9, 11], ["Discrete", "PDF"] \right]. \quad \square$$

In the following example where $Y = g(X)$ is a piecewise many-to-one transformation, the only additional step required of the algorithm (as compared to the one-to-one transformation in Example 5.4) is the clean up process of Y , i.e., the removal of identical support values from the PDF of Y .

Example 5.5. (Bain & Engelhardt, 1992, page 203) Let $f(x) = \frac{4}{31}(\frac{1}{2})^x$ for $x = -2, -1, 0, 1, 2$. Determine the PDF of $Y = |X|$.

Solution: The transformation $g(X) = |X|$ is a piecewise many-to-one transformation for $\Omega_X = \{-2, -1, 0, 1, 2\}$ to $\Omega_Y = \{0, 1, 4\}$. Since X will be entered in APPL in its *NoDot* format in this example, the function $g(x)$ may be entered in APPL as one segment, i.e., `[[x -> abs(x)], [-infinity, infinity]]`. The function $g(X) = |X|$ is applied to Ω_X to yield the support values in Ω_Y . The support values obtained by these transformations (in the second APPL sublist of Y) are

$$[g(-2), g(-1), g(0), g(1), g(2)] = [2, 1, 0, 1, 2].$$

Before sorting and combining identical support values, Y has the form

$$\left[\left[\frac{16}{31}, \frac{8}{31}, \frac{4}{31}, \frac{2}{31}, \frac{1}{31} \right], [2, 1, 0, 1, 2], ["Discrete", "PDF"] \right].$$

After sorting the support values (and corresponding probability values) in increasing order, Y has the form

$$\left[\left[\frac{4}{31}, \frac{8}{31}, \frac{2}{31}, \frac{16}{31}, \frac{1}{31} \right], [0, 1, 1, 2, 2], ["Discrete", "PDF"] \right].$$

The identical support values are removed, and the final form of Y is

$$\left[\left[\frac{4}{31}, \frac{10}{31}, \frac{17}{31} \right], [0, 1, 2], ["Discrete", "PDF"] \right].$$

The APPL statements needed to return this PDF are

```
> X := [[x -> (4 / 31) * (1 / 2) ^ x], [-2, -1, 0, 1, 2],
        ["Discrete", "PDF"]];
> g := [[x -> abs(x)], [-infinity, infinity]];
> Y := Transform(X, g);
```

□

Example 5.6. Let X be a uniform discrete random variable for $x = -1, 0, \dots, 8$ and

$$Y = g(X) = \begin{cases} X^2 & x < 4 \\ 8 - X & x \geq 4. \end{cases}$$

Determine the PDF of Y .

Solution: The support Ω_X is partitioned into two subsets: $\Omega_{X_1} = \{-1, 0, 1, 2, 3\}$ and $\Omega_{X_2} = \{4, 5, 6, 7, 8\}$. The transformation $Y_1 = g_1(X) = X^2$ is applied to Ω_{X_1} and the transformation $Y_2 = g_2(X) = 8 - X$ is applied to Ω_{X_2} . The PDF of Y ,

$$f_Y(y) = \begin{cases} 1/5 & y = 0 \\ 3/10 & y = 1 \\ 1/10 & y = 2 \\ 1/10 & y = 3 \\ 1/5 & y = 4 \\ 1/10 & y = 9, \end{cases}$$

is returned by the APPL statements

```
> X := UniformDiscreteRV(-1, 8);
> g := [[x -> x ^ 2, x -> 8 - x], [-infinity, 3, infinity]];
> Y := Transform(X, g);
```

□

5.2.2 *Dot* Format

In the APPL *Dot* data structure format, the random variable X has a formulaic PDF in the first sublist and pattern describing X 's support in the second sublist. Let X , for example, have PDF $f(x) = x/15$ for $x = 1, 2, \dots, 5$. The standard *Dot* format of X is

$$\left[\left[x \rightarrow \frac{x}{15} \right], [1 .. 5, 1, x \rightarrow x], ["Discrete", "PDF"] \right].$$

Since the second sublist indicates that the support starts at the value one, each additional value is incremented by one and transformed by $x \rightarrow x$, and the last support value is five, then the support is clearly $\{1, 2, 3, 4, 5\}$. Since the PDF of X is formulaic, then any transformation $g(x)$ that is applied to the random variable X will affect both its first and second sublists. The formulaic probability function is determined by finding the appropriate inverse function $g_i^{-1}(x)$ for each monotone segment Ω_{X_i} , substituting the inverse $g_i^{-1}(y)$ for x into the formulaic PDF (for the appropriate Ω_{X_i} partition), and making the appropriate adjustments to the support of X . There are four separate subcases in APPL for determining the transformation of a discrete random variable X with a *Dot* format that are outlined in Table 5.1 and described separately in the following four subsections.

Table 5.1: Categories for computing the PDF of the random variable $Y = g(X)$ when X is a discrete random variable with support Ω_X in a *Dot* support format.

Ω_X	$g(x)$ Continuous, $g(x)$ in One Piece, $g(x)$ One-to-One on Ω_X	Action	Resulting Y
Finite	Yes	Substitute $g(X)$ into formulaic PDF; reverse support list, if necessary	Random variable in standard <i>Dot</i> format
	No	Convert X to standard <i>NoDot</i> format	Random variable in standard <i>NoDot</i> format
Infinite	Yes	Substitute $g(X)$ into formulaic PDF; reverse support list, if necessary	Random variable in standard <i>Dot</i> format
	No	—	Random variable with an “alien” APPL format

Finite support, $Y = g(X)$ a Continuous One-to-One Function Given in One Piece

The process for determining the PDF of $Y = g(X)$ when X is a discrete random variable with finite support with an APPL *Dot* format and $g(x)$ is a continuous one-to-one function that is given in one piece is:

- The random variable X is converted to its PDF representation, if necessary.
- Each monotone segment of $g(x)$ is indicated in the first sublist in its list of two sublists format. If $g(x)$ is continuous and one-to-one on Ω_X , then only one transformation will be in the first sublist. The correct inverse must be determined for $x \in \Omega_X$, which is sometimes difficult since Maple sometimes produces several candidates for $g^{-1}(y)$. The correct inverse is selected by requiring that

$g^{-1}(g(c)) = c$, where $c = (\min\{\Omega_X\} + \max\{\Omega_X\})/2$, which is valid because $\min\{\Omega_X\} \neq -\infty$, $\max\{\Omega_X\} \neq \infty$.

- The correct inverse $g^{-1}(y)$ is substituted into the formulaic PDF for x .
- The support Ω_Y is returned as a *Dot* support range with either the format:

(a) $[\min\{\Omega_X\} .. \max\{\Omega_X\}, 1, x \rightarrow g(x)]$, or

(b) $[\max\{\Omega_X\} .. \min\{\Omega_X\}, -1, x \rightarrow g(x)]$.

If $g(x)$ is an increasing function on Ω_X [i.e., $g(\min\{\Omega_X\}) < g(\max\{\Omega_X\})$], then the support of Ω_Y has format type (a). If $g(x)$ is a decreasing function [i.e., $g(\min\{\Omega_X\}) > g(\max\{\Omega_X\})$], then the support of Ω_Y has format type (b).

- The PDF of Y is returned in its standard *Dot* format.

Example 5.7. Let X again have PDF $f(x) = x/15$ for $x = 1, 2, \dots, 5$. Find the PDF of $Y = 2X + 1$ in its APPL *Dot* format.

Solution: The random variable X and transformation $g(X)$ are entered into APPL as

```
> X := [[x -> x / 15], [1 .. 5], ["Discrete", "PDF"]];
> g := [[x -> 2 * x + 1], [-infinity, infinity]];
```

The transformation $g(X) = 2X + 1$ is a one-to-one transformation for $\Omega_X = \{1, 2, 3, 4, 5\}$. The procedure **Transform** recognizes that $g(x)$ is one-to-one since there is only one function in its first sublist. The unique inverse $g^{-1}(y) = (y - 1)/2$ is substituted for x in X 's formulaic PDF. The PDF of Y is

$$f_Y(y) = \frac{y-1}{30}$$

for $y \in \{3, 5, 7, 9, 11\}$. The support Ω_Y is returned in its *Dot* format as $[1..5, x \rightarrow 2x + 1]$ (since $g(x) = 2x + 1$ is an increasing function for $x \in \Omega_X$). The final APPL statement

```
> Y := Transform(X, gX);
```

returns the PDF of Y as

$$\left[\left[x \rightarrow \frac{x-1}{30} \right], [1..5, x \rightarrow 2x + 1], ["Discrete", "PDF"] \right]. \quad \square$$

Finite support, $Y = g(X)$ a Discontinuous or Piecewise Many-to-One Function

The process for determining the PDF of $Y = g(X)$ when X is a discrete random variable with finite support with an APPL *Dot* format and $g(x)$ is a discontinuous or piecewise many-to-one function is:

- The random variable X is converted to its PDF representation, if necessary.
- Each monotone segment of $g(x)$ is indicated in the first sublist in its list of two sublists format. If $g(x)$ is discontinuous or piecewise many-to-one on Ω_X , then more than one transformation will be in the first sublist. To avoid returning an “alien” APPL random variable format (as discussed in the upcoming “Infinite support, $Y = g(X)$ a Discontinuous or Piecewise Many-to-One Function” subsection) for the PDF of Y , X is converted to its *NoDot* format with the `ConvertToNoDot` procedure.
- The PDF of Y is determined as discussed in the *NoDot* format section. Although this approach poses no difficulty within the `Transform` procedure itself,

the procedure does return a random variable in a different format than the one in which it was entered. We prefer this consequence (*NoDot* format) to the currently inevitable alternative—an unfriendly APPL random variable that is not acceptable as an argument to other APPL procedures.

Infinite support, $Y = g(X)$ a Continuous One-to-One Function Given in One Piece

The process for determining the PDF of $Y = g(X)$ when X is a discrete random variable with infinite support with an APPL *Dot* format and $g(x)$ is a continuous one-to-one function given in one piece is:

- The random variable X is converted to its PDF representation, if necessary.
- If $g(x)$ is continuous and one-to-one on Ω_X , then only one transformation will be in the first sublist. The correct inverse is determined (for $x \in \Omega_X$) by using Glen et al.'s approach (1997, page 289). The correct inverse is selected by requiring that $g^{-1}(g(c)) = c$, where c is a point in the support range of Ω_X . The portion of the algorithm for determining c is
 1. If $\min\{\Omega_X\} = -\infty$ and $\max\{\Omega_X\} = \infty$, then $c = 0$.
 2. If $\min\{\Omega_X\} = -\infty$ and $\max\{\Omega_X\} \neq \infty$, then $c = \max\{\Omega_X\} - 1$.
 3. If $\min\{\Omega_X\} \neq -\infty$ and $\max\{\Omega_X\} = \infty$, then $c = \min\{\Omega_X\} + 1$.
 4. For all other cases, $c = (\min\{\Omega_X\} + \max\{\Omega_X\})/2$.
- The correct inverse $g^{-1}(y)$ is substituted into the formulaic PDF for x .
- The support Ω_Y is returned as a *Dot* support range with either the format:
 - (a) $[\min\{\Omega_X\} .. \max\{\Omega_X\}, 1, x \rightarrow g(x)]$, or

(b) $[\max\{\Omega_X\} .. \min\{\Omega_X\}, -1, x \rightarrow g(x)]$.

If $g(x)$ is an increasing function on Ω_X , then the support of Ω_Y has format type

(a). If $g(x)$ is a decreasing function, then the support of Ω_Y has format type (b).

In some cases, the support sublist may look awkward, such as $[\infty .. 1, -1, x \rightarrow \frac{1}{x}]$ for $Y = 1/X$ where X is a geometric random variable, but it is in a standard *Dot* format.

- The PDF of Y is returned in its standard *Dot* format.

Example 5.8. (Hogg & Craig, 1995, pages 163–164) Let X have the Poisson PDF

$$f(x) = \frac{\mu^x e^{-\mu}}{x!} \quad x = 0, 1, 2, \dots$$

Find the PDF of $Y = 4X$.

Solution: The PDF of $Y = 4X$ is determined with the APPL statements

```
> X := PoissonRV(mu);
> Y := Transform(X, [[x -> 4 * x], [-infinity, infinity]]);
```

The resulting PDF for Y is

$$f(y) = \frac{\mu^{y/4} e^{-\mu}}{(y/4)!} \quad y = 0, 4, 8, \dots$$

APPL returns the lists-of-sublists for Y as

$$\left[\left[x \rightarrow \frac{\mu^{x/4} e^{-\mu}}{(x/4)!} \right], [0 .. \infty, x \rightarrow 4x], ["Discrete", "PDF"] \right].$$

□

Infinite support, $Y = g(X)$ a Discontinuous or Piecewise Many-to-One Function

The process for determining the PDF of $Y = g(X)$ when X is a discrete random variable with infinite support with an APPL *Dot* format and $g(x)$ is a discontinuous or piecewise many-to-one function is:

- The random variable X is converted to its PDF representation, if necessary.
- Each monotone segment of $g(x)$ is indicated in the first sublist in its list of two sublists format. If $g(x)$ is discontinuous or many-to-one on Ω_X , then more than one transformation will be in the first sublist. In this situation, it is currently impossible to avoid returning an “alien” APPL random variable format for the PDF of Y . Since X has infinite support, it cannot be converted to a *NoDot* format. Thus, at least two formulaic (piecewise) PDFs are returned in the first sublist of Y , and sublist two must be adjusted to reflect the piecewise PDFs. The algorithm must identify which segment of the transformation function is applied to which support value or values of X . This is more difficult than in the *NoDot* case since each support value is not listed, but rather alluded to by the second sublist data structure. Since it is impossible to mechanically calculate an infinite number of support values, the algorithm simply returns the transformed support as a range of support values followed by the transformation that applies to it. For example, if $X \sim \text{Poisson}(2)$, then its support in sublist two is $[0 .. \infty]$. If the transformation is $Y = |X - 2|$, then the transformed random variable Y is

$$\left[\left[x \rightarrow \frac{2^{(2-x)}e^{-2}}{(2-x)!}, x \rightarrow \frac{2^{(2+x)}e^{-2}}{(2+x)!} \right], \right. \\ \left. [2 .. 0, -1, x \rightarrow 2 - x, 3 .. \infty, x \rightarrow x - 2], ["Discrete", "PDF"] \right].$$

- The algorithm determines the appropriate inverse for the i th monotone segment with corresponding transformation function $g_i(x)$, for each i . The correct inverse is selected by requiring that $g_i^{-1}(g_i(c_i)) = c_i$ when c_i is a point in the support range of Ω_{X_i} . The algorithm for determining c_i (where the i th subinterval has endpoints x_i and x_{i+1}) is
 1. If $x_1 = -\infty$ and $x_2 \neq \infty$, then $c_1 = x_2 - 1$.
 2. If $x_n \neq -\infty$ and $x_{n+1} = \infty$, then $c_n = x_n + 1$.
 3. For all other cases, $c_i = (x_i + x_{i+1})/2$.
- The correct inverses $g_i^{-1}(y)$ are substituted into the formulaic PDF for x .
- If $g_i(x)$ is an increasing function on Ω_{X_i} , then the support of the transformed random variable, Ω_Y , has format type:

$$[x_i .. x_{i+1}, 1, x \rightarrow g_i(x)].$$

If $g_i(x)$ is a decreasing function on Ω_{X_i} , then the support of the transformed random variable, Ω_Y , has format type:

$$[x_{i+1} .. x_i, -1, x \rightarrow g_i(x)].$$

- The transformed random variable Y is printed (not in its standard *Dot* format) and a warning message is returned. This is the first time APPL has output a random variable with more than one formulaic PDF in its first sublist. Other APPL procedures are unable to work with a random variable in this format. Further work on the discrete data structure will handle this situation. Unfortunately, the transformed PDF Y is being output in a form that is foreign (or alien) to other APPL procedures.

5.3 Applications

When a sample X_1, X_2, \dots, X_n is drawn, some summary statistics are computed, such as the mean, variance, and median. It is also possible to compute the distributions of these summary statistics provided that the X_i are iid random samples from a given distribution. Examples using the Transform procedure to compute distributions for the sample mean, sample geometric mean, sample harmonic mean, and sample quadratic mean are displayed on the next several pages.

Example 5.9. (Sample mean) Let X_1, X_2, \dots, X_{10} be iid Bernoulli random variables with general parameter $0 < p < 1$. Find the PDF of the sample mean \bar{X} .

Solution: The APPL statements below define X as a Bernoulli random variable with parameter p , find the convolution of the ten Bernoulli random variables, and transform the resulting convolution by $1/10$.

```
> n := 10;
> X := BernoulliRV(p);
> Y := ConvolutionIID(X, n);
> Z := Transform(Y, [[x -> x / n], [-infinity, infinity]]);
```

The resulting PDF for the sample mean $Z = \bar{X}$ is

$$f_Z(z) = \begin{cases} (1-p)^{10} & z = 0 \\ 10p(1-p)^9 & z = 1/10 \\ 45p^2(1-p)^8 & z = 1/5 \\ 120p^3(1-p)^7 & z = 2/5 \\ 210p^4(1-p)^6 & z = 3/10 \\ 252p^5(1-p)^5 & z = 1/2 \\ 210p^6(1-p)^4 & z = 3/5 \\ 120p^7(1-p)^3 & z = 7/10 \\ 45p^8(1-p)^2 & z = 4/5 \\ 10p^9(1-p) & z = 9/10 \\ p^{10} & z = 1, \end{cases}$$

which is the distribution of the ratio of a binomial(10, p) random variable and 10. \square

Example 5.10. (Sample geometric mean) Suppose that an urn contains 2 red balls and 8 white balls. If 5 balls are drawn at random without replacement, then the total number of red balls selected X has a hypergeometric distribution with PDF

$$f_X(x) = \frac{320}{(2-x)!x!(3+x)!(5-x)!} \quad x = 0, 1, 2.$$

Let X_1, X_2, \dots, X_{12} be iid hypergeometric random variables with the above PDF. Find the distribution of the sample geometric mean G .

Solution: The sample geometric mean G is defined by

$$G = \left(\prod_{i=1}^n X_i \right)^{1/n}.$$

The APPL statements

```
> n := 12;
> X := HypergeometricRV(10, 2, 5);
> Y := ProductIID(X, n);
> G := Transform(Y, [[x -> x ^ (1 / n)], [-infinity, infinity]]);
```

return the PDF of G in its list-of-sublists as

$$G := \left[\left[\frac{268588249280}{282429536481}, \frac{244140625}{282429536481}, \frac{390625000}{94143178827}, \frac{859375000}{94143178827}, \right. \right. \\ \left. \frac{3437500000}{282429536481}, \frac{343750000}{31381059609}, \frac{220000000}{31381059609}, \frac{308000000}{94143178827}, \right. \\ \left. \frac{35200000}{35200000}, \frac{8800000}{8800000}, \frac{14080000}{14080000}, \frac{563200}{563200}, \right. \\ \left. \frac{31381059609}{40960}, \frac{31381059609}{4096}, \frac{282429536481}{282429536481}, \frac{94143178827}{94143178827}, \right. \\ \left. \frac{40960}{94143178827}, \frac{4096}{282429536481} \right], [0, 1, 2^{1/12}, 2^{1/6}, 2^{1/4}, 2^{1/3}, 2^{5/12}, \\ \sqrt{2}, 2^{7/12}, 2^{2/3}, 2^{3/4}, 2^{5/6}, 2^{11/12}, 2], ["Discrete", "PDF"]]. \quad \square$$

Example 5.11. (Sample harmonic mean) Let X_1, X_2, X_3, X_4 be iid uniform discrete

random variables with PDF $f_X(x) = 1/6$ for $x = 1, 2, \dots, 6$. Find the distribution of the sample harmonic mean H .

Solution: The sample harmonic mean H is defined by

$$H = \frac{1}{\frac{1}{n} \sum_{i=1}^n \frac{1}{X_i}}$$

The APPL statements

```
> n := 4;
> X := UniformDiscreteRV(1, 6);
> Temp1 := Transform(X, [[x -> 1 / x], [0, infinity]]);
> Temp2 := ConvolutionIID(Temp1, n);
> H := Transform(Temp2, [[x -> n / x], [0, infinity]]);
```

return the PDF of H as

$$f_H(h) = \left\{ \begin{array}{ll} 1/1296 & h = 1, 5, 6 \\ 1/324 & h = 8/7, 6/5, 16/13, 5/4, 24/19, 5/2, 40/11, 30/7, 80/17, \\ & 120/23, 16/3, 40/7 \\ 1/216 & h = 4/3, 5/3, 20/7, 60/11 \\ 1/108 & h = 24/17, 16/11, 40/27, 48/31, 30/19, 80/49, 48/29, 120/71, \\ & 16/9, 20/11, 40/19, 30/13, 80/33, 120/47, 80/29, 120/37, \\ & 80/23, 240/59, 240/49, 240/47 \\ 1/72 & h = 3/2, 12/7, 15/4, 40/9 \\ 11/648 & h = 8/5 \\ 1/54 & h = 24/13, 48/25, 120/61, 80/39, 240/107, 240/97, 48/19, 60/23, \\ & 48/17, 120/41, 240/77, 60/13 \\ 41/1296 & h = 2 \\ 1/36 & h = 48/23, 15/7, 240/67, 120/31 \\ 5/162 & h = 24/11, 40/17, 10/3 \\ 2/81 & h = 16/7, 16/5, \\ 17/648 & h = 12/5, 24/7 \\ 13/648 & h = 8/3 \\ 31/1296 & h = 3 \\ 7/324 & h = 48/13, 80/19 \\ 23/1296 & h = 4 \\ 1/81 & h = 48/11 \\ 5/648 & h = 24/5. \end{array} \right.$$

□

Example 5.12. (Sample quadratic mean) Let X_1, X_2, \dots, X_{10} be iid Bernoulli random variables with parameter p , where $0 < p < 1$. Find the distribution of the sample quadratic mean Q .

Solution: The sample quadratic mean Q is defined by

$$Q = \sqrt{\frac{1}{n} \sum_{i=1}^n X_i^2}.$$

The APPL statements

```
> n := 10;
> X := BernoulliRV(p);
> Y := Transform(X, [[x -> x ^ 2], [-infinity, infinity]]);
> Z := ConvolutionIID(Y, n);
> T := Transform(Z, [[x -> x / n], [-infinity, infinity]]);
> Q := Transform(T, [[x -> sqrt(x)], [0, infinity]]);
```

return the PDF of Q as

$$f_Q(q) = \begin{cases} (1-p)^{10} & q = 0 \\ 10p(1-p)^9 & q = \sqrt{10}/10 \\ 45p^2(1-p)^8 & q = \sqrt{5}/5 \\ 120p^3(1-p)^7 & q = \sqrt{30}/10 \\ 210p^4(1-p)^6 & q = \sqrt{10}/5 \\ 252p^5(1-p)^5 & q = \sqrt{2}/2 \\ 210p^6(1-p)^4 & q = \sqrt{15}/5 \\ 120p^7(1-p)^3 & q = \sqrt{70}/10 \\ 45p^8(1-p)^2 & q = 2\sqrt{5}/5 \\ 10p^9(1-p) & q = 3\sqrt{10}/10 \\ p^{10} & q = 1. \end{cases}$$

□

Example 5.13. (Reliability) Three different components, numbered one, two, and three, are tested. They are to be arranged in a series system. The number of components tested and successes for each type of component are listed in Table 5.1. The

point estimate for the system reliability is $\frac{21}{23} \cdot \frac{27}{28} \cdot \frac{82}{84} = \frac{1107}{1288} \cong 0.8595$. Find the lower 95% bootstrap confidence interval bound.

Table 5.2: Life tests on a three-component system.

Component	Number on Test	Number of Passes
1	23	21
2	28	27
3	84	82

Solution: The APPL statements below, which utilize the `Product` and `Transform` procedures, are used to determine the lower 95% bootstrap confidence interval bound for the system reliability.

```
> n1 := 23; s1 := 21;
> X1 := BinomialRV(n1, s1 / n1);
> X1 := Transform(X1, [[x -> x / n1], [-infinity, infinity]]);
> n2 := 28; s2 := 27;
> X2 := BinomialRV(n2, s2 / n2);
> X2 := Transform(X2, [[x -> x / n2], [-infinity, infinity]]);
> n3 := 84; s3 := 82;
> X3 := BinomialRV(n3, s3 / n3);
> X3 := Transform(X3, [[x -> x / n3], [-infinity, infinity]]);
> Temp := Product(X1, X2);
> T := Product(Temp, X3);
```

Out of the possible $24 \cdot 29 \cdot 85 = 59,610$ potential mass values for T determined by the `Product` procedure, only 6,419 remain since the procedure combines redundant values. The lower 95% bootstrap confidence interval bound is $120/161 \cong 0.7453$. This lower bound was verified by the following `Splus` function, which samples 10,000 systems to determine the lower bound:

```
seriessystemboot <- function(n, y, alpha) {
```

```

nn <- length(n)
nrep <- 10000
yy <- rep(1, nrep)
point <- prod(y) / prod(n)
for (j in 1:nrep) {
  for (i in 1:nn) {
    yy[j] <- yy[j] * rbinom(1, n[i], y[i] / n[i]) / n[i]
  }
}
yy <- sort(yy)
interval <- yy[floor(alpha * nrep)]
c(point, interval)
}

```

The function `seriessystemboot` was called five times with the command:

```
seriessystemboot(c(23, 28, 84), c(21, 27, 82), 0.05)
```

yielding 0.7457, 0.7487, 0.7457, 0.7402, and 0.7457.

□

Chapter 6

Minimums and Maximums

Let X and Y be two independent discrete random variables with supports Ω_X and Ω_Y , respectively. This chapter outlines a procedure for determining the PDF of the minimum and maximum of X and Y . At first glance, it appears that computing the PDF of $\min\{X, Y\}$ is no more difficult than determining the PDF of the smallest order statistic for *some* discrete random variable. This is true when X and Y are identically distributed; we can just use the `OrderStat` procedure (introduced in Chapter 3) with the following three parameters:

- $\mathbf{x} = X$ (or Y),
- $\mathbf{n} = 2$ (the sample size drawn from the population), and
- $\mathbf{r} = 1$, i.e., minimum.

When X and Y are not identically distributed, their supports (finite or infinite) and specific support value relationships (e.g., $\max\{\Omega_X\} < \max\{\Omega_Y\}$) determine how the PDFs of their minimum and maximum are calculated. Table 6.1 illustrates the various categories considered when the `Minimum` procedure determines the PDF of the minimum of independent discrete random variables X and Y . Each category is discussed in its own subsection. Since computing the PDF of the maximum is more

than just a reversal of the algorithm for computing the PDF of the minimum, it is discussed separately with examples and figures at the end of this chapter.

Table 6.1: Categories for computing the PDF of the minimum of two independent, non-identically distributed random variables X and Y . The notation $U(a, b)$ represents a uniform discrete random variable on the interval $[a, b]$, $\text{Geo}(p)$ represents a geometric random variable with parameter p , and $\text{NegBinom}(r, p)$ represents a negative binomial random variable with parameters r and p .

Supports Ω_X & Ω_Y	Support Value Relationships	Examples
Ω_X finite, Ω_Y finite	$\max\{\Omega_X\} = \max\{\Omega_Y\}$	$X \sim U(1, 6), Y \sim U(3, 6)$
	$\max\{\Omega_X\} < \max\{\Omega_Y\}$ (or vice versa)	$X \sim U(1, 4), Y \sim U(3, 5)$
Ω_X infinite, Ω_Y infinite	$\min\{\Omega_X\} = \min\{\Omega_Y\}$	$X \sim \text{Geo}(1/2), Y \sim \text{Geo}(1/4)$
	$\min\{\Omega_X\} < \min\{\Omega_Y\}$ (or vice versa)	$X \sim \text{Geo}(1/2),$ $Y \sim \text{NegBinom}(2, 1/2)$
Ω_X infinite, Ω_Y finite (or vice versa)	$\min\{\Omega_X\} = \max\{\Omega_Y\}$	$X \sim \text{Geo}(1/2), Y \sim U(1, 4)$
	$\min\{\Omega_X\} < \max\{\Omega_Y\}$	$X \sim \text{Geo}(1/2), Y \sim U(3, 5)$
	$\min\{\Omega_X\} > \max\{\Omega_Y\}$	$X \sim \text{NegBinom}(2, 1/2),$ $Y \sim U(1, 4)$

6.1 PDF of the Minimum

Let X and Y be two independent discrete random variables with supports Ω_X and Ω_Y , respectively. Let $Z = \min\{X, Y\}$. Then the CDF of Z is computed as

$$\begin{aligned}
 F_Z(z) &= \Pr(Z \leq z) \\
 &= 1 - \Pr(Z > z) \\
 &= 1 - \Pr(\min\{X, Y\} > z) \\
 &= 1 - \Pr(X > z) \cdot \Pr(Y > z) \quad (X \text{ and } Y \text{ are independent}) \\
 &= 1 - (1 - \Pr(X \leq z)) \cdot (1 - \Pr(Y \leq z)) \\
 &= 1 - (1 - F_X(z)) \cdot (1 - F_Y(z)) \tag{6.1}
 \end{aligned}$$

for $z \in \Omega_Z \subset \Omega_X \cup \Omega_Y$, where Ω_Z is discussed further in the various subsections. The CDF of Z can be converted to its PDF representation if desired. When X and Y are identically distributed (as discussed in the following subsection), the CDF of Z when n samples are drawn from X 's population is

$$F_Z(z) = 1 - (1 - F_X(z))^n \quad z \in \Omega_X. \tag{6.2}$$

6.1.1 Identically Distributed Random Variables

For discrete iid random variables, the `MinimumIID(X, n)` procedure was written to determine the PDF of the minimum of n iid random variables X . The `MinimumIID(X, n)` procedure determines the PDF of the minimum by making the procedure call: `OrderStat(X, n, 1)`. From Chapter 3, `OrderStat(X, n, r)` determines the PDF of the r th order statistic when n random samples are drawn (with replacement) from the parent population corresponding to the random variable X . Thus, the statement

$\text{OrderStat}(X, n, 1)$ determines the PDF of the first order statistic, the minimum, when n random samples are drawn from the parent population. Unlike the Minimum procedure, MinimumIID can determine the PDF of the minimum for more than just two random variables. Also, as shown in Example 6.2, MinimumIID can be used for random variables with infinite supports.

Example 6.1. (Adapted from Bain & Engelhardt, 1992, page 54) A fair four-sided die is rolled twice. Determine the PDF of the minimum.

Solution: Let X_i be the outcome of the die on its i th roll, $i = 1, 2$. Then $f_{X_i}(x) = 1/4$ for $x = 1, 2, 3, 4$; $i = 1, 2$. Let $Z = \min\{X_1, X_2\}$. The diagram in Figure 6.1 shows that the PDF of Z is

$$f_Z(z) = \begin{cases} \frac{7}{16} & z = 1 \\ \frac{5}{16} & z = 2 \\ \frac{3}{16} & z = 3 \\ \frac{1}{16} & z = 4. \end{cases}$$

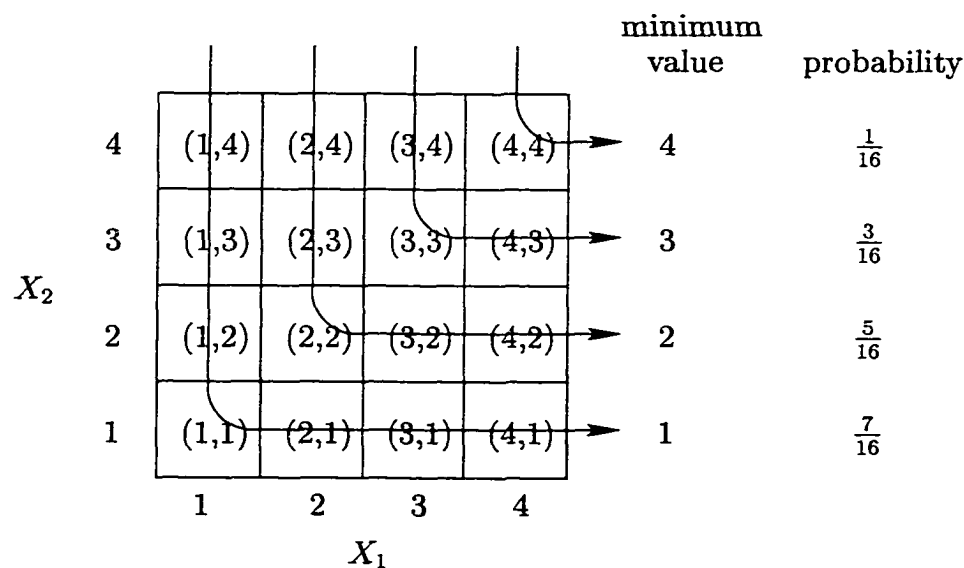


Figure 6.1: The PDF of the minimum when a four-sided die is rolled twice.

The APPL statements required to determine this PDF are

```
> X := UniformDiscreteRV(1, 4);
> Z := MinimumIID(X, 2);
```

□

Example 6.2. (Adapted from Bain & Engelhardt, 1992, page 229) Consider a random sample of size $n = 4$ from a geometric distribution with PDF $f_X(x) = p \cdot (1-p)^{x-1}$ for $x = 1, 2, \dots$; $0 < p < 1$. Determine the PDF of the minimum.

Solution: Let Z be the minimum of the four geometric random variables. The CDF of X is $F_X(x) = 1 - (1-p)^x$, and thus the CDF of Z (by equation 6.2) is

$$\begin{aligned} F_Z(z) &= 1 - (1 - (1 - (1 - p)^z))^4 \\ &= 1 - (1 - p)^{4z} \quad z = 1, 2, \dots \end{aligned}$$

The simplified PDF representation of Z , found by differencing, is

$$f(z) = \frac{-p(p^4 - 4p^3 + 6p^2 - 4p + 1)^z (p^3 - 4p^2 + 6p - 4)}{(1-p)^4} \quad z = 1, 2, \dots,$$

which is obtained with the APPL statements

```
> X := GeometricRV(p);
> Z := MinimumIID(X, 4);
```

□

6.1.2 Non-identically Distributed Random Variables

One of the largest obstacles in determining the PDF of Z when X and Y do not have the same distribution is working with supports Ω_X and Ω_Y that are often not identical [e.g., $X \sim \text{geometric}(1/2)$, $Y \sim \text{negative binomial}(2, 1/2)$]. There are three categories to consider: (1) Ω_X and Ω_Y both finite, (2) Ω_X and Ω_Y infinite, and (3) Ω_X infinite and Ω_Y finite (or vice versa).

Ω_X Finite, Ω_Y Finite

In APPL, random variables with finite supports come in one of two formats: *Dot* or *NoDot*. Instead of working with random variables in their *Dot* formats and producing alien APPL PDFs (e.g., the transformed random variable displayed in its list-of-sublists at the bottom of page 162), random variables with *Dot* formats are converted to their *NoDot* formats with the `ConvertToNoDot` procedure at the start of the `Minimum` procedure. A brute force method is used to determine the PDF of the minimum Z .

For explanation purposes, let X and Y be the random variables

$$f_X(x) = \begin{cases} 0.2 & x = 1 \\ 0.3 & x = 5 \\ 0.4 & x = 7 \\ 0.1 & x = 9, \end{cases} \quad f_Y(y) = \begin{cases} 0.6 & y = 4 \\ 0.1 & y = 5 \\ 0.3 & y = 6. \end{cases}$$

The support of $Z = \min\{X, Y\}$ contains only the support values of Ω_X and Ω_Y that result in nonzero probabilities (for minima), i.e., $\Omega_Z = \{1, 4, 5, 6\}$. Thus, if

- $\max\{\Omega_X\} = \max\{\Omega_Y\}$, then $\Omega_Z = \Omega_X \cup \Omega_Y$;
- $\max\{\Omega_X\} > \max\{\Omega_Y\}$, then the $\Omega_Z = \{x \in \Omega_X \mid x \leq \max\{\Omega_Y\}\} \cup \Omega_Y$. The example introduced in this subsection falls into this category;
- $\max\{\Omega_X\} < \max\{\Omega_Y\}$, then the $\Omega_Z = \Omega_X \cup \{y \in \Omega_Y \mid y \leq \max\{\Omega_X\}\}$.

For each support value $z \in \Omega_Z$, the probability value $F_Z(z)$ is computed using equation 6.1. In APPL, the value $F_Z(z)$ is computed using the `CDF` procedure. That is,

$$F_Z(z) = 1 - (1 - \text{CDF}(X, z)) \cdot (1 - \text{CDF}(Y, z))$$

for each $z \in \Omega_Z$. After the CDF of Z is determined in the Minimum procedure, it is converted to its PDF representation with the PDF procedure.

Example 6.3. Determine the PDF of $Z = \min\{X, Y\}$ for the random variables X and Y introduced in this subsection.

Solution: Figure 6.2 illustrates how the five minimum support values in Ω_Z are obtained.

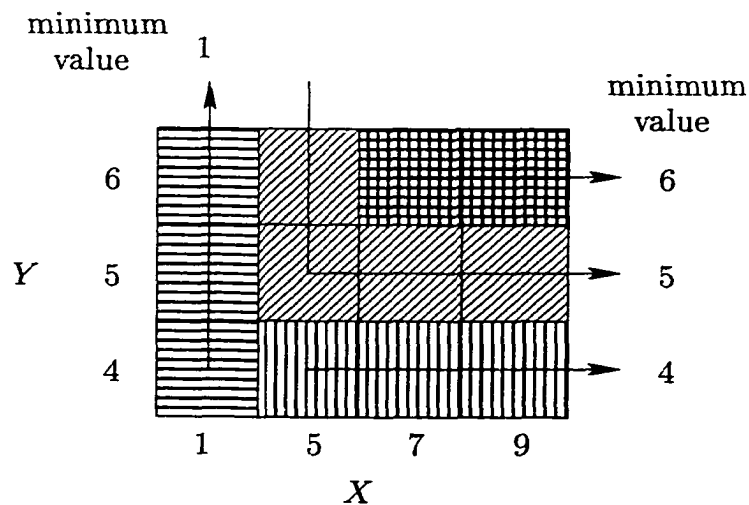


Figure 6.2: The minimum values $\Omega_Z = \{1, 4, 5, 6\}$ for X and Y in Example 6.3.

Using equation 6.1, the CDF of Z and its corresponding PDF are

$$F_Z(z) = \begin{cases} 0.2 & z = 1 \\ 0.68 & z = 4 \\ 0.85 & z = 5 \\ 1 & z = 6, \end{cases} \quad f_Z(z) = \begin{cases} 0.2 & z = 1 \\ 0.48 & z = 4 \\ 0.17 & z = 5 \\ 0.15 & z = 6. \end{cases}$$

The PDF of Z is determined with the APPL statements

```
> X := [[0.2, 0.3, 0.4, 0.1], [1, 5, 7, 9], ["Discrete", "PDF"]];
> Y := [[0.6, 0.1, 0.3], [4, 5, 6], ["Discrete", "PDF"]];
> Z := Minimum(X, Y);
```

□

Example 6.4. Fair six-sided and twelve-sided dice are rolled. Determine the proba-

bility that the minimum face showing is a three or less.

Solution: The probability that the minimum face Z is less than or equal to three is $5/8$, as determined by the APPL statements

```
> X := UniformDiscreteRV(1, 6);
> Y := UniformDiscreteRV(1, 12);
> Z := Minimum(X, Y);
> CDF(Z, 3);
```

□

Ω_X Infinite, Ω_Y Infinite

When Ω_X and Ω_Y have infinite supports, the random variables X and Y have *Dot* formats in APPL. Thus, a formulaic PDF for $Z = \min\{X, Y\}$ will be returned in APPL. Assume that the supports Ω_X and Ω_Y are subsets of adjacent integer values. (This is often the case since many random variables, such as the geometric, Poisson, and negative binomial random variables, have this type of support.) In this particular category, it is important to determine whether or not $\min\{\Omega_X\} = \min\{\Omega_Y\}$.

In the case where $\min\{\Omega_X\} = \min\{\Omega_Y\}$, the CDF of Z is computed using equation 6.1 and the CDFs of X and Y . In APPL, the CDF procedure is used to determine the formulaic CDFs for X and Y . Once the CDF of Z is determined, its PDF is calculated—in APPL, this is done with the PDF procedure.

Example 6.5. Determine the probability that the minimum of a Poisson random variable X with a mean of one and a Poisson random variable Y with a mean of two is greater than or equal to three.

Solution: The CDF of $Z = \min\{X, Y\}$ can be determined with equation 6.1, and then converted to its PDF representation. In APPL, the desired probability, $\Pr(Z \geq 3)$, is computed with the SF (survivor function) procedure. The statements

```
> X := PoissonRV(1);
> Y := PoissonRV(2);
```

```
> Z := Minimum(X, Y);
> evalf(SF(Z, 3));
```

return the probability as the floating point approximation 0.0260. \square

In the case where $\min\{\Omega_X\} \neq \min\{\Omega_Y\}$, the PDF of $Z = \min\{X, Y\}$ is a piecewise defined function. Assume without loss of generality that $\min\{\Omega_X\} < \min\{\Omega_Y\}$. The support of the minimum Z is identical to the support of X , i.e., $\Omega_Z = \Omega_X$. For $z \in \{\min\{\Omega_X\}, \min\{\Omega_X\} + 1, \dots, \min\{\Omega_Y\} - 1\} \subset \Omega_X$, $f_Z(z) = f_X(z)$. For $z \in \{\min\{\Omega_Y\}, \min\{\Omega_Y\} + 1, \dots\} = \Omega_X - \{\min\{\Omega_X\}, \min\{\Omega_X\} + 1, \dots, \min\{\Omega_Y\} - 1\}$, the formulaic piece of the CDF $F_Z(z)$ is computed using equation 6.1. The corresponding formulaic piece of the PDF $f_Z(z)$ for $z \in \{\min\{\Omega_Y\}, \min\{\Omega_Y\} + 1, \dots\}$ is computed using both the formulaic CDF segment of $F_Z(z)$ for $z \in \{\min\{\Omega_Y\}, \min\{\Omega_Y\} + 1, \dots\}$ and the PDF values $f_Z(z)$ for $z \in \{\min\{\Omega_X\}, \min\{\Omega_X\} + 1, \dots, \min\{\Omega_Y\} - 1\}$.

For example, let X be a geometric random variable with parameter $p = 1/2$, and let Y be a negative binomial random variable with parameters $r = 2$ and $p = 1/2$. The PDF of X is $f_X(x) = (1/2)^x$, $x = 1, 2, \dots$, and the PDF of Y is $f_Y(y) = \frac{(y-1)!(1/2)^{y-2}}{4(y-2)!}$, $y = 2, 3, \dots$. For $z \in \{\min\{\Omega_X\}, \min\{\Omega_X\} + 1, \dots, \min\{\Omega_Y\} - 1\} = \{1\}$, the first segment of the PDF of Z is $f_Z(z) = (1/2)^z$. As indicated by the first column in Figure 6.3 with $x = 1$, the minimum value $z = 1$ occurs with probability $1/2$, since $\sum_{i=2}^{\infty} \frac{(y-1)!(1/2)^{y-2}}{4(y-2)!} = 1$. For $z \in \{\min\{\Omega_Y\}, \min\{\Omega_Y\} + 1, \dots\} = \{2, 3, \dots\}$, the formulaic piece of the CDF of Z is computed using equation 6.1 with the CDFs $F_X(x) = 1 - (1/2)^x$, $x = 1, 2, \dots$, and $F_Y(y) = 1 - (y+1)/2^y$, $y = 2, 3, \dots$

$$F_X(x) = 1 - (1/2)^x, \quad x = 1, 2, \dots, \quad \text{and} \quad F_Y(y) = 1 - (y+1)/2^y, \quad y = 2, 3, \dots$$

The resulting PDF for the minimum Z for the example random variables X and

$$f_Z(z) = \begin{cases} 1/2 & z = 1 \\ \frac{3z-1}{4^z} & z = 2, 3, \dots \end{cases}$$

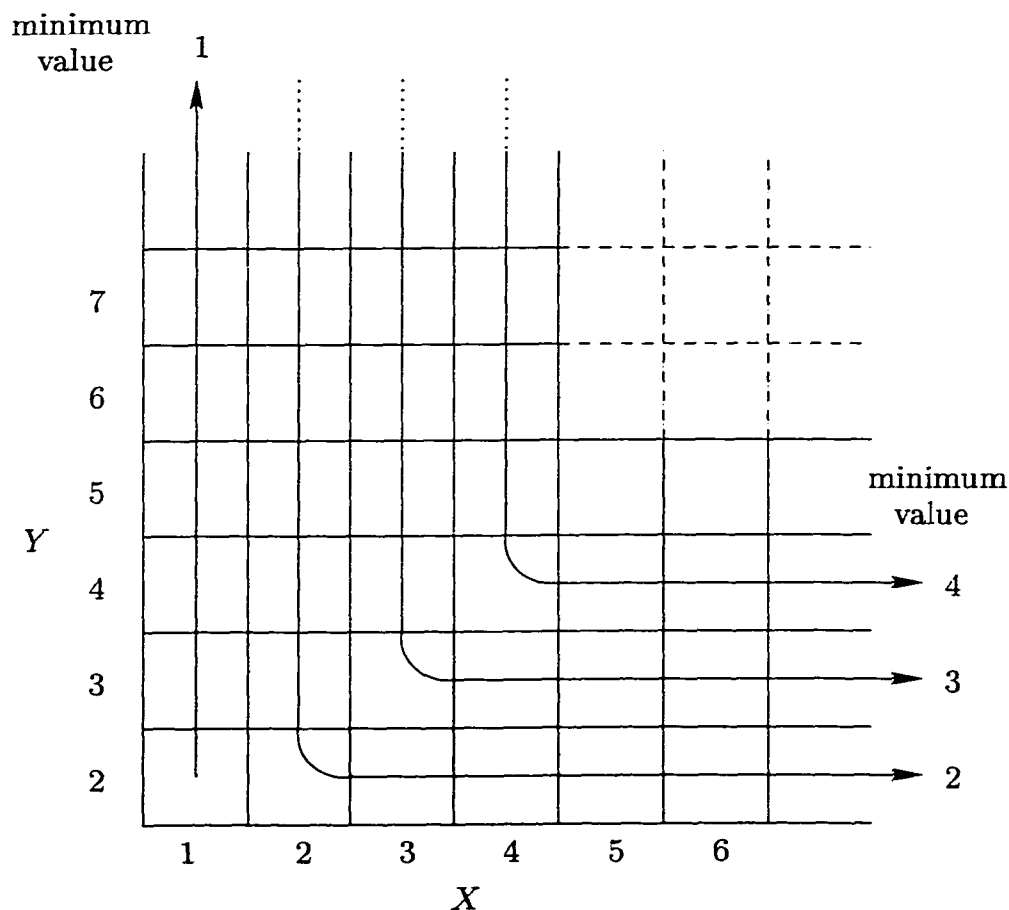


Figure 6.3: The support values for a geometric random variable with support $\Omega_X = \{1, 2, \dots\}$ and a negative binomial random variable with support $\Omega_Y = \{2, 3, \dots\}$.

As discussed in Chapter 5 on transformations, APPL is forced to return an error message and print the PDF of Z in a form that is alien to other APPL procedures. APPL prints the PDF for Z as

$$\left[\left[x \rightarrow \left(\frac{1}{2} \right)^x, x \rightarrow \frac{3x-1}{4^x} \right], [1, 2 .. \infty], ["Discrete", "PDF"] \right].$$

Ω_X Infinite, Ω_Y Finite

For discussion purposes, X has an infinite support and Y has a finite support in this subsection, though they could be swapped without consequence. Assume that

the support Ω_X is a subset of adjacent integer values. There are three cases to consider in this category: (1) $\min\{\Omega_X\} = \min\{\Omega_Y\}$, (2) $\min\{\Omega_X\} > \min\{\Omega_Y\}$, and (3) $\min\{\Omega_X\} < \min\{\Omega_Y\}$.

- $\min\{\Omega_X\} = \min\{\Omega_Y\}$

Example 6.6. Let $X \sim \text{geometric}(1/2)$ with PDF $f_X(x) = (1/2)^x$ for $x = 1, 2, \dots$, and Y have PDF

$$f_Y(y) = \begin{cases} 1/4 & y = 1 \\ 1/8 & y = 3 \\ 1/2 & y = 4 \\ 1/8 & y = 6. \end{cases}$$

Determine the PDF of the minimum Z .

Solution: In the case where $\min\{\Omega_X\} = \min\{\Omega_Y\}$, $\Omega_Z = \{\min\{\Omega_X\}, \min\{\Omega_X\} + 1, \dots, \max\{\Omega_Y\}\}$. Since $|\Omega_Z|$ is finite, the probability value $F_Z(z)$ is computed (by brute force) for each $z \in \Omega_Z$ using equation 6.1. In APPL, Z will have a *NoDot* format, where its second sublist will range over all integers between and including $\min\{\Omega_X\}$ to $\max\{\Omega_Y\}$. Figure 6.4 illustrates how the six minimum values are determined in this example.

The CDF of X is $F_X(x) = 1 - (1/2)^x$ for $x = 1, 2, \dots$, and the CDF of Y is

$$F_Y(y) = \begin{cases} 1/4 & y = 1 \\ 3/8 & y = 3 \\ 7/8 & y = 4 \\ 1 & y = 6. \end{cases}$$

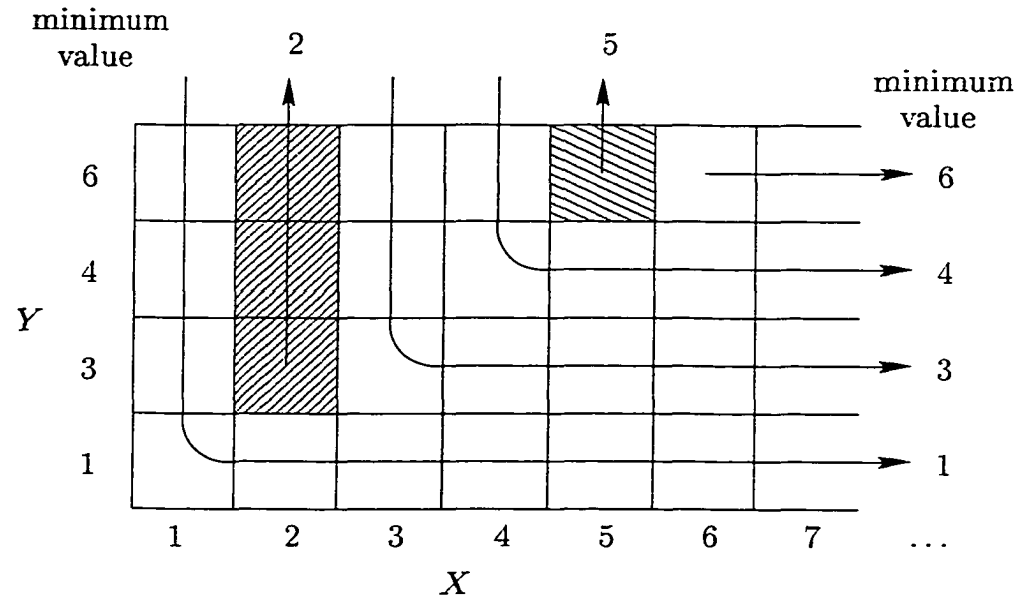


Figure 6.4: The support values for the random variable X with infinite support and the random variable Y with finite support, where $\min\{\Omega_X\} = \min\{\Omega_Y\}$.

For $z = 5$, for example,

$$\begin{aligned}
 F_Z(5) &= \Pr(Z \leq 5) \\
 &= 1 - \Pr(\min\{X, Y\} > 5) \\
 &= 1 - (1 - \Pr(X \leq 5)) \cdot (1 - \Pr(Y \leq 5)) \\
 &= 1 - \left(\frac{1}{2}\right)^5 \cdot \left(\frac{1}{8}\right) \\
 &= \frac{255}{256}.
 \end{aligned}$$

The other CDF values are obtained similarly to yield the CDF of Z as

$$F_Z(z) = \begin{cases} 5/8 & z = 1 \\ 13/16 & z = 2 \\ 59/64 & z = 3 \\ 127/128 & z = 4 \\ 255/256 & z = 5 \\ 1 & z = 6. \end{cases}$$

The resulting PDF, computed by the APPL statements

```

> X := GeometricRV(1 / 2);
> Y := [[1 / 4, 1 / 8, 1 / 2, 1 / 8], [1, 3, 4, 6],
        ["Discrete", "PDF"]];
> Z := Minimum(X, Y);

```

is

$$f_Z(z) = \begin{cases} 5/8 & z = 1 \\ 3/16 & z = 2 \\ 7/64 & z = 3 \\ 9/128 & z = 4 \\ 1/256 & z = 5 \\ 1/256 & z = 6. \end{cases} \quad \square$$

- $\min\{\Omega_X\} > \min\{\Omega_Y\}$

Example 6.7. Let X be a negative binomial(4, 1/2) random variable with PDF $f_X(x) = \frac{(x-1)!(1/2)^{x-4}}{96(x-4)!}$, $x = 4, 5, \dots$ and Y be a random variable with PDF

$$f_Y(y) = \begin{cases} 1/4 & y = 1 \\ 1/4 & y = 3 \\ 1/4 & y = 5 \\ 1/4 & y = 7. \end{cases}$$

Find the PDF of $Z = \min\{X, Y\}$.

Solution: Figure 6.5 illustrates the support values of X and Y . When $\min\{\Omega_X\} > \min\{\Omega_Y\}$, $\Omega_Z = \{y \in \Omega_Y \mid y < \min\{\Omega_X\}\} \cup \{\min\{\Omega_X\}, \min\{\Omega_X\} + 1, \dots, \max\{\Omega_Y\}\}$. For $z \in \{y \in \Omega_Y \mid y < \min\{\Omega_X\}\}$, $F_Z(z) = F_Y(z)$. For $z \in \{\min\{\Omega_X\}, \min\{\Omega_X\} + 1, \dots, \max\{\Omega_Y\}\}$, the value $F_Z(z)$ is computed using equation 6.1 and the CDFs of X and Y . Finally, the CDF of Z is converted to its PDF representation.

In APPL, the assignment $Y := \text{UniformDiscreteRV}(a, b, k)$; defines Y as a uniform discrete random variable (provided that k divides $b - a$) with PDF

$$f_Y(y) = \frac{1}{n+1} \quad y = a, a+k, a+2k, \dots, a+nk,$$

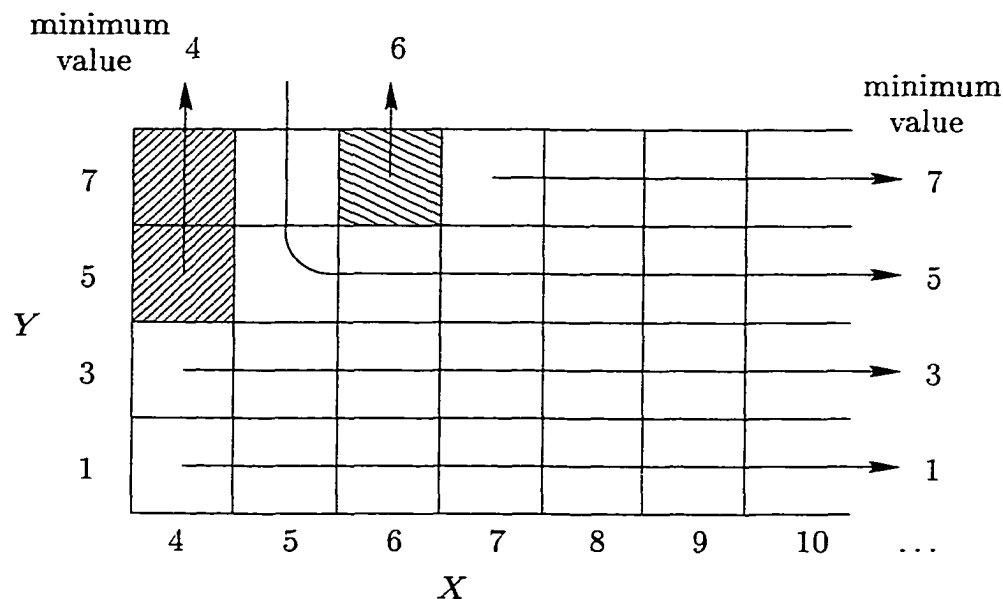


Figure 6.5: The support values for the random variable X with infinite support and the random variable Y with finite support, where $\min\{\Omega_X\} > \min\{\Omega_Y\}$.

where $n = (b - a)/k$. Thus, the statements

```
> X := NegativeBinomialRV(4, 1 / 2);
> Y := UniformDiscreteRV(1, 7, 2);
> Z := Minimum(X, Y);
```

return the PDF of Z as

$$f_Z(z) = \begin{cases} 1/4 & z = 1 \\ 1/4 & z = 3 \\ 1/32 & z = 4 \\ 17/64 & z = 5 \\ 5/128 & z = 6 \\ 21/128 & z = 7. \end{cases}$$

□

- $\min\{\Omega_X\} < \min\{\Omega_Y\}$

Example 6.8. Let X be a geometric($1/4$) random variable with PDF $f_X(x) =$

$\frac{1}{4}\left(\frac{3}{4}\right)^{x-1}$, $x = 1, 2, \dots$, and Y be a random variable with PDF

$$f_Y(y) = \begin{cases} 1/4 & y = 4 \\ 1/4 & y = 6 \\ 1/4 & y = 7 \\ 1/4 & y = 9. \end{cases}$$

Find the PDF of $Z = \min\{X, Y\}$.

Solution: Figure 6.6 illustrates the support values of X and Y . When $\min\{\Omega_X\} < \min\{\Omega_Y\}$, $\Omega_Z = \{x \in \Omega_X \mid x < \min\{\Omega_Y\}\} \cup \{\min\{\Omega_Y\}, \min\{\Omega_Y\} + 1, \dots, \max\{\Omega_Y\}\}$. For $z \in \{x \in \Omega_X \mid x < \min\{\Omega_Y\}\}$, $F_Z(z) = F_X(z)$. For $z \in \{\min\{\Omega_Y\}, \min\{\Omega_Y\} + 1, \dots, \max\{\Omega_Y\}\}$, $F_Z(z)$ is computed using equation 6.1 and the CDFs of X and Y . Finally, the CDF of Z is converted to its PDF representation.

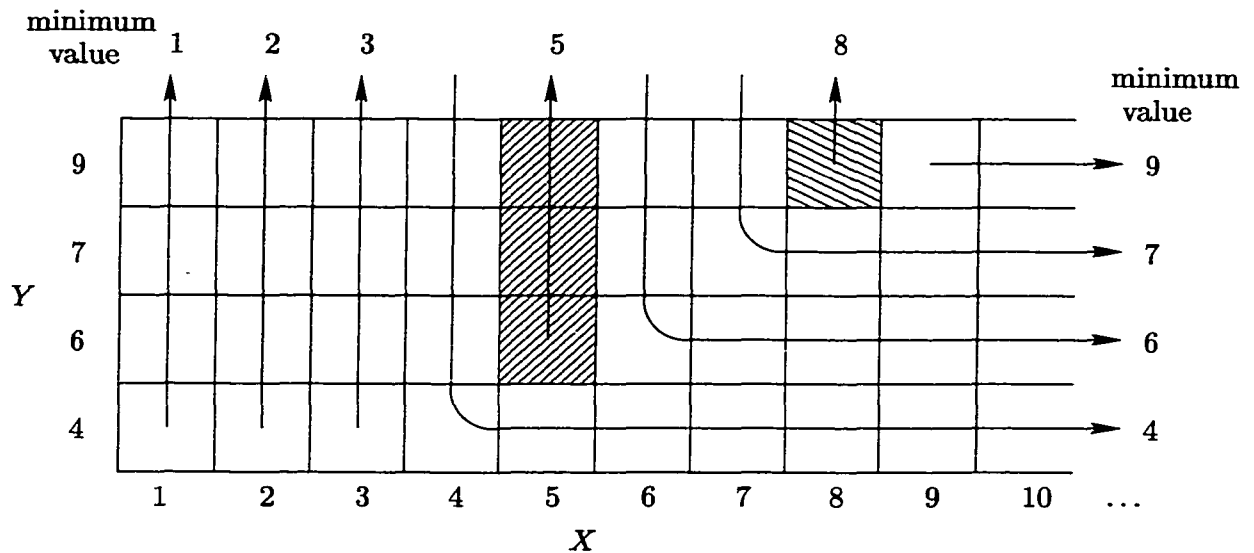


Figure 6.6: The support values for the random variable X with infinite support and the random variable Y with finite support, where $\min\{\Omega_X\} < \min\{\Omega_Y\}$.

In APPL, the statements

```
> X := GeometricRV(1 / 4);
```

```

> Y := [[1 / 4, 1 / 4, 1 / 4, 1 / 4], [4, 6, 7, 9],
        ["Discrete", "PDF"]];
> Z := Minimum(X, Y);

```

return the PDF of Z (in its *NoDot* format) as

$$f_Z(z) = \begin{cases} 1/4 & z = 1 \\ 3/16 & z = 2 \\ 9/64 & z = 3 \\ 189/1024 & z = 4 \\ 243/4096 & z = 5 \\ 729/8192 & z = 6 \\ 3645/65536 & z = 7 \\ 2187/262144 & z = 8 \\ 6561/262144 & z = 9. \end{cases}$$

□

6.2 PDF of the Maximum

Let X and Y be two independent discrete random variables with supports Ω_X and Ω_Y , respectively. Let $M = \max\{X, Y\}$. Then the CDF of M is computed as

$$\begin{aligned}
F_M(m) &= \Pr(M \leq m) \\
&= \Pr(\max\{X, Y\} \leq m) \\
&= \Pr(X \leq m) \cdot \Pr(Y \leq m) && (X \text{ and } Y \text{ are independent}) \\
&= \Pr(X \leq m) \cdot \Pr(Y \leq m) \\
&= F_X(m) \cdot F_Y(m) && (6.3)
\end{aligned}$$

for $m \in \Omega_M \subset \Omega_X \cup \Omega_Y$, where Ω_M is discussed further in the various subsections. The CDF of M can be converted to its PDF representation if desired. When X and Y are identically distributed, the CDF of M when n samples are drawn from X 's

population is

$$F_M(m) = F_X(m)^n \quad m \in \Omega_X. \quad (6.4)$$

For discrete iid random variables or random variables with finite support, the `MaximumIID` and `Maximum` procedures are closely related to their counterpart minimum procedures in APPL. The `MaximumIID(X, n)` procedure determines the PDF of the maximum of n iid random variables X . The `MaximumIID` procedure determines the PDF of the maximum by calling the `OrderStat` procedure with the random variable $X = X$, the number of samples drawn $n = n$, and the order statistic $r = n$. The statement `OrderStat(X, n, n)` determines the PDF of the largest order statistic. Unlike the `Maximum` procedure, `MaximumIID` can determine the PDF of the maximum for more than just two random variables.

Example 6.9. A fair twelve-sided die is rolled five times. Determine the PDF of the maximum.

Solution: Let X_i be the outcome of the die on its i th roll, $i = 1, 2, 3, 4, 5$. Then $f_{X_i}(x) = 1/12$ for $x = 1, 2, \dots, 12$; $i = 1, 2, \dots, 5$. Let $M = \max\{X_1, X_2, X_3, X_4, X_5\}$. The CDF of M is computed using equation 6.4. Its corresponding PDF is

$$f_M(m) = \begin{cases} 1/248832 & m = 1 \\ 31/248832 & m = 2 \\ 211/248832 & m = 3 \\ 781/248832 & m = 4 \\ 2101/248832 & m = 5 \\ 4651/248832 & m = 6 \\ 9031/248832 & m = 7 \\ 15961/248832 & m = 8 \\ 26281/248832 & m = 9 \\ 40951/248832 & m = 10 \\ 61051/248832 & m = 11 \\ 87781/248832 & m = 12. \end{cases}$$

The APPL statements required to determine this PDF are

```
> X := UniformDiscreteRV(1, 12);
> M := MaximumIID(X, 5);
```

□

As in the `Minimum` procedure, random variables with *Dot* formats are converted to their *NoDot* formats with the `ConvertToNoDot` procedure. A brute force method is used to determine the PDF of the maximum M .

For explanation purposes, again let X and Y be the random variables

$$f_X(x) = \begin{cases} 0.2 & x = 1 \\ 0.3 & x = 5 \\ 0.4 & x = 7 \\ 0.1 & x = 9, \end{cases} \quad f_Y(y) = \begin{cases} 0.6 & y = 4 \\ 0.1 & y = 5 \\ 0.3 & y = 6. \end{cases}$$

The support of M contains only the support values of Ω_X and Ω_Y that result in nonzero probabilities (as maxima), i.e., $\Omega_M = \{4, 5, 6, 7, 9\}$. Thus, if

- $\min\{\Omega_X\} = \min\{\Omega_Y\}$, then $\Omega_M = \Omega_X \cup \Omega_Y$;
- $\min\{\Omega_X\} > \min\{\Omega_Y\}$, then the $\Omega_M = \Omega_X \cup \{y \in \Omega_Y \mid y \leq \min\{\Omega_X\}\}$;
- $\min\{\Omega_X\} < \min\{\Omega_Y\}$, then the $\Omega_M = \{x \in \Omega_X \mid x \leq \min\{\Omega_Y\}\} \cup \Omega_Y$. The example reintroduced in this subsection falls into this category.

For each support value $m \in \Omega_M$, the probability value $F_M(m)$ is computed using equation 6.3. In APPL, the value $F_M(m)$ is computed using the CDF procedure. That is,

$$F_M(m) = \text{CDF}(X, m) \cdot \text{CDF}(Y, m)$$

for each $m \in \Omega_M$. After the CDF of M is determined in the `Maximum` procedure, it is converted to its PDF representation with the `PDF` procedure.

Example 6.10. Determine the PDF of $M = \max\{X, Y\}$ for the random variables X and Y reintroduced in this subsection.

Solution: Figure 6.7 illustrates how the five maximum support values $m \in \Omega_M$ are obtained. Using equation 6.3, the CDF of M and its corresponding PDF are

$$F_M(m) = \begin{cases} 0.12 & m = 4 \\ 0.35 & m = 5 \\ 0.5 & m = 6 \\ 0.9 & m = 7 \\ 1 & m = 9, \end{cases} \quad f_M(m) = \begin{cases} 0.12 & m = 4 \\ 0.23 & m = 5 \\ 0.15 & m = 6 \\ 0.4 & m = 7 \\ 0.1 & m = 9. \end{cases}$$

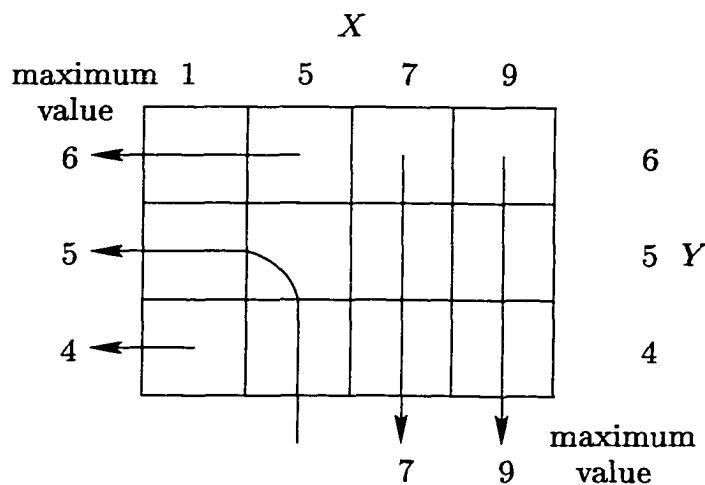


Figure 6.7: The maximum values $\{4, 5, 6, 7, 9\}$ for X and Y in Example 6.10.

The PDF of M is determined with the APPL statements

```
> X := [[0.2, 0.3, 0.4, 0.1], [1, 5, 7, 9], ["Discrete", "PDF"]];
> Y := [[0.6, 0.1, 0.3], [4, 5, 6], ["Discrete", "PDF"]];
> M := Maximum(X, Y);
```

□

One or Both Ω_X and Ω_Y Infinite

We assume that the supports of Ω_X and Ω_Y are integer valued. When both Ω_X and Ω_Y are infinite, it is important to determine whether or not $\min\{\Omega_X\} = \min\{\Omega_Y\}$. If

- $\min\{\Omega_X\} = \min\{\Omega_Y\}$, then $\Omega_M = \Omega_X$ (or Ω_Y);
- $\min\{\Omega_X\} > \min\{\Omega_Y\}$, then $\Omega_M = \Omega_X$;
- $\min\{\Omega_X\} < \min\{\Omega_Y\}$, then $\Omega_M = \Omega_Y$.

The maximum CDF in all cases is determined using equation 6.3 and can then be converted to its PDF representation.

Example 6.11. Let X be a negative binomial random variable with parameters $r = 2$ and $p = 1/2$, and Y be a geometric random variable with parameter $p = 1/4$. Determine the PDF of $M = \max\{X, Y\}$.

Solution: In APPL, the PDF is computed with the statements

```
> X := NegativeBinomialRV(2, 1 / 2);
> Y := GeometricRV(1 / 4);
> M := Maximum(X, Y);
```

The resulting PDF is

$$f_M(m) = -\frac{5m3^{m-1}}{8^m} + \frac{m}{2^m} + \left(\frac{3}{8}\right)^m - \left(\frac{1}{2}\right)^m + \frac{3^{m-1}}{4^m} \quad m = 2, 3, \dots$$

Notice that $\Omega_M = \Omega_Y$, since $\min\{\Omega_X\} < \min\{\Omega_Y\}$. □

When either Ω_X or Ω_Y is infinite (and the other is finite), the relationship between $\min\{\Omega_X\}$ and $\min\{\Omega_Y\}$ plays an important role in determining Ω_M . For discussion purposes, choose X to have an infinite support and Y to have a finite support, though they could be swapped without consequence. There are three cases to again consider: (1) $\min\{\Omega_X\} = \min\{\Omega_Y\}$, (2) $\min\{\Omega_X\} > \min\{\Omega_Y\}$, and (3) $\min\{\Omega_X\} < \min\{\Omega_Y\}$.

- $\min\{\Omega_X\} = \min\{\Omega_Y\}$

Example 6.12. Let $X \sim \text{geometric}(1/2)$ with PDF $f_X(x) = (1/2)^x$ for $x = 1, 2, \dots$, and Y have PDF

$$f_Y(y) = \begin{cases} 1/4 & y = 1 \\ 1/8 & y = 3 \\ 1/2 & y = 4 \\ 1/8 & y = 6. \end{cases}$$

Determine the PDF of $M = \max\{X, Y\}$.

Solution: In the case where $\min\{\Omega_X\} = \min\{\Omega_Y\}$, $\Omega_M = \{\min\{\Omega_X\}, \min\{\Omega_X\} + 1, \dots\}$, i.e., $|\Omega_M|$ is infinite. The probability value $F_M(m)$ is computed (by brute force) for each $m \in \{\min\{\Omega_X\}, \min\{\Omega_X\} + 1, \dots, \max\{\Omega_Y\}\}$ using equation 6.1. In APPL, M has a *NoDot* format for $m \in \{\min\{\Omega_X\}, \min\{\Omega_X\} + 1, \dots, \max\{\Omega_Y\}\}$. For each $m \in \{\max\{\Omega_Y\} + 1, \max\{\Omega_Y\} + 2, \dots\}$, $f_M(m) = f_X(m)$, and M has a *Dot* format for these m 's. Figure 6.8 illustrates how the maximum values are determined in this example.

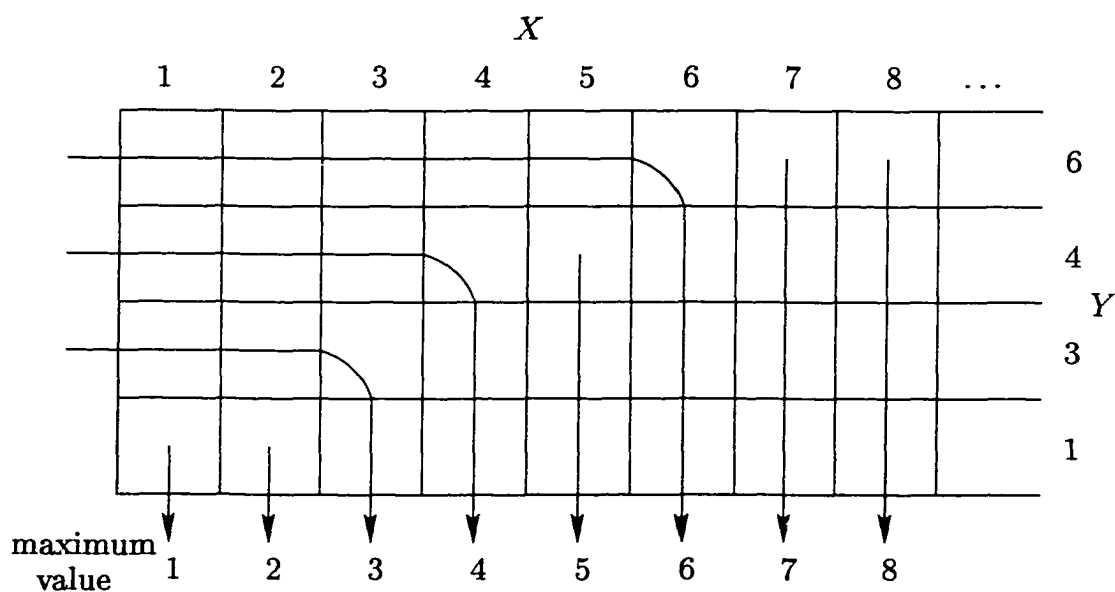


Figure 6.8: The support values for the random variable X with infinite support and the random variable Y with finite support, where $\min\{\Omega_X\} = \min\{\Omega_Y\}$.

The CDF of X is $F_X(x) = 1 - (1/2)^x$ for $x = 1, 2, \dots$, and the CDF of Y is

$$F_Y(y) = \begin{cases} 1/4 & y = 1 \\ 3/8 & y = 3 \\ 7/8 & y = 4 \\ 1 & y = 6. \end{cases}$$

For $m = 5$, for example, $F_M(5) = (31/32) \cdot (7/8) = 217/256$. The other CDF values are obtained similarly and the PDF of M is

$$f_M(m) = \begin{cases} 1/8 & m = 1 \\ 1/16 & m = 2 \\ 9/64 & m = 3 \\ 63/128 & m = 4 \\ 7/256 & m = 5 \\ 35/256 & m = 6 \\ (1/2)^m & m = 7, 8, \dots \end{cases}$$

The APPL statements

```
> X := GeometricRV(1 / 2);
> Y := [[1 / 4, 1 / 8, 1 / 2, 1 / 8], [1, 3, 4, 6],
        ["Discrete", "PDF"]];
> M := Maximum(X, Y);
```

prints an error message about the random variable's alien format and prints the PDF of M in a split *NoDot/Dot* format as

$$M := \left[\left[\frac{1}{8}, \frac{1}{16}, \frac{9}{64}, \frac{63}{128}, \frac{7}{256}, \frac{35}{256}, x \rightarrow \left(\frac{1}{2}\right)^x \right], \right. \\ \left. [1, 2, 3, 4, 5, 6, 7.. \infty], ["Discrete", "PDF"] \right]. \quad \square$$

- $\min\{\Omega_X\} > \min\{\Omega_Y\}$

Example 6.13. Let X be a negative binomial(4, 1/2) random variable with

PDF $f_X(x) = \frac{(x-1)!(1/2)^{x-4}}{96(x-4)!}$, $x = 4, 5, \dots$ and Y be a random variable with PDF

$$f_Y(y) = \begin{cases} 1/4 & y = 1 \\ 1/4 & y = 3 \\ 1/4 & y = 5 \\ 1/4 & y = 7. \end{cases}$$

Find the PDF of $M = \max\{X, Y\}$.

Solution: The support values yielded by X and Y are $\{4, 5, 6, \dots\}$. When $\min\{\Omega_X\} > \min\{\Omega_Y\}$, the support of M is the same as the support of X , i.e., $\Omega_M = \Omega_X$. For $m \in \{x \in \Omega_X | x \leq \max\{\Omega_Y\}\}$, the CDF of M is computed using equation 6.1 and the CDFs of X and Y . For $m \in \{\max\{\Omega_Y\} + 1, \max\{\Omega_Y\} + 2, \dots\}$, $F_M(m) = F_X(m)$. Finally the CDF of M is converted to its PDF representation.

In APPL, the statements

```
> X := NegativeBinomialRV(4, 1 / 2);
> Y := UniformDiscreteRV(1, 7, 2);
> M := Maximum(X, Y);
```

print the PDF of M as

$$M := \left[\left[\frac{1}{32}, \frac{7}{64}, \frac{15}{128}, \frac{31}{128}, x \rightarrow \frac{(x-1)!(1/2)^{x-4}}{96(x-4)!} \right], \right. \\ \left. [4, 5, 6, 7, 8 .. \infty], ["Discrete", "PDF"] \right].$$

□

- $\min\{\Omega_X\} < \min\{\Omega_Y\}$

Example 6.14. Let X be a geometric(1/4) random variable with PDF $f_X(x) = \frac{1}{4} \left(\frac{3}{4}\right)^{x-1}$, $x = 1, 2, \dots$ and Y be a random variable with PDF

$$f_Y(y) = \begin{cases} 1/4 & y = 4 \\ 1/4 & y = 6 \\ 1/4 & y = 7 \\ 1/4 & y = 9. \end{cases}$$

Find the PDF of $M = \max\{X, Y\}$.

Solution: Figure 6.9 illustrates the support values of X and Y . When $\min\{\Omega_X\} < \min\{\Omega_Y\}$, $\Omega_M = \{\min\{\Omega_X\}, \min\{\Omega_X\}+1, \dots\}$. For $m \in \{\min\{\Omega_X\}, \min\{\Omega_X\}+1, \dots, \max\{\Omega_X\}\}$, the CDF of M is computed using equation 6.1 and the CDFs of X and Y . For $m \in \{\max\{\Omega_X\} + 1, \max\{\Omega_X\} + 2, \dots\}$, $F_M(m) = F_X(m)$. Finally the CDF of M is converted to its PDF representation.

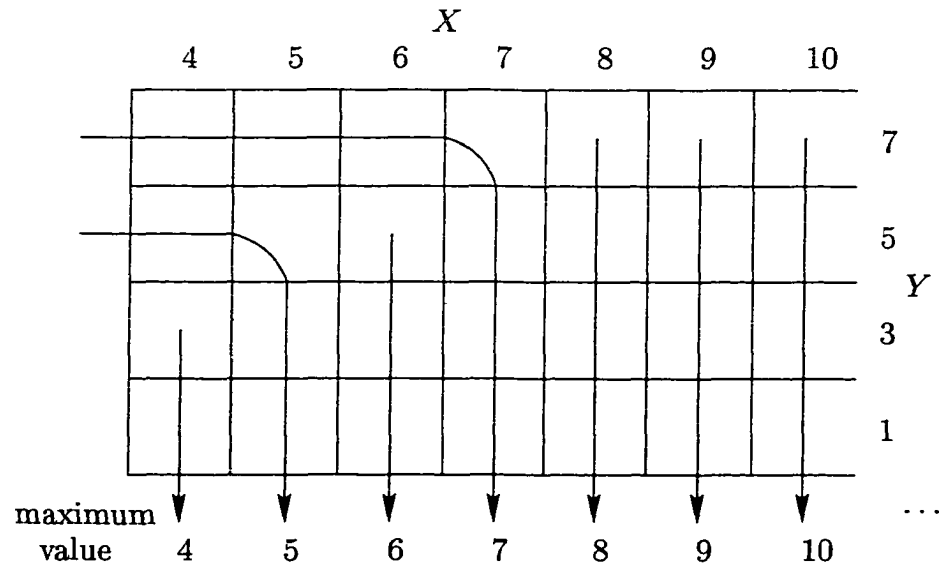


Figure 6.9: The support values for the random variable X with infinite support and the random variable Y with finite support, where $\min\{\Omega_X\} < \min\{\Omega_Y\}$.

In APPL, the statements

```
> X := GeometricRV(1 / 4);
> Y := [[1 / 4, 1 / 4, 1 / 4, 1 / 4], [4, 6, 7, 9],
        ["Discrete", "PDF"]];
> M := Maximum(X, Y);
```

return the PDF of M as

$$f_M(m) = \begin{cases} 175/1024 & m = 4 \\ 81/4096 & m = 5 \\ 1805/8192 & m = 6 \\ 15655/65536 & m = 7 \\ 6561/262144 & m = 8 \\ 1/4 & m = 9 \\ (1/4) \cdot (3/4)^{m-1} & m = 10, 11, \dots \end{cases}$$

□

Chapter 7

Algorithms for Operations on Continuous Distributions

This chapter contains work on algorithms associated with the manipulation of mainly continuous random variables. The first section describes how the algorithmic procedure `VerifyPDF`, which was originally written by Dr. Glen for his dissertation, was rewritten to check the validity of a random variable's probability density function. The second section finds method of moments estimators for a real or symbolic data set associated with a particular distribution. The third section finds maximum likelihood estimators for a complete or right-censored data set. The fourth section introduces the `APPL Mixture` and `Truncate` procedures.

7.1 Existence Conditions for PDFs

For a continuous random variable X , its PDF $f(x)$ must satisfy

- $f(x) \geq 0$ for $-\infty < x < \infty$, and
- $\int_{-\infty}^{\infty} f(x) dx = 1$.

The latter condition is typically straightforward to verify using a symbolic mathematical software package since integral evaluation is a built-in procedure in these packages. Showing that $f(x)$ is nonnegative over its support is more complicated though since there is no easy way to check that $f(x)$ is nonnegative for each and every x in an uncountable range. This section describes how the VerifyPDF procedure was rewritten in order to check that $f(x) \geq 0$ for all x in its domain. Also, VerifyPDF was extended to confirm the validity of probability mass functions for discrete random variables.

In order to show why it is important to verify that $f(x)$ is nonnegative over its support, consider the continuous random variable X with PDF $f(x) = 3|x| - 1$ for $-1 \leq x \leq 1$. Previously, the procedure VerifyPDF reported that $f(x)$ was a valid PDF since $f(x)$ integrates to 1 on the interval $[-1, 1]$. The PDF $f(x)$ is not valid, however, since, for example, $f(0) = -1$. At that time, the VerifyPDF procedure checked that $f(x)$ was positive at its endpoints. Although the integration and endpoint check correctly identified the validity of *most* PDFs, including the PDFs of standard distributions (e.g., exponential, gamma, normal, uniform), it incorrectly verified some distributions with invalid PDFs as valid, as the one illustrated in this paragraph.

Graphically, it is easy to tell if the PDF $f(x)$ of a random variable X is nonnegative. If the graph of $f(x)$ dips below the x -axis for any value of x in the random variable's support, then X does not have a valid PDF. Figure 7.1, for example, illustrates that the random variable X with PDF $f(x) = 3|x| - 1$ is not valid for $-1 \leq x \leq 1$ since its graph clearly extends below the x -axis. Since gleaning results interactively from a graphical display of a PDF is mechanically impossible in many mathematical software packages (including Maple), an analytic method to indicate negative PDF values was developed. The analytic method involves checking that both

$\int_{-\infty}^{\infty} f(x) dx$ and $\int_{-\infty}^{\infty} |f(x)| dx$ integrate to the value 1. By the following proposition, if these conditions are satisfied, then $f(x) \geq 0$ for all real x .

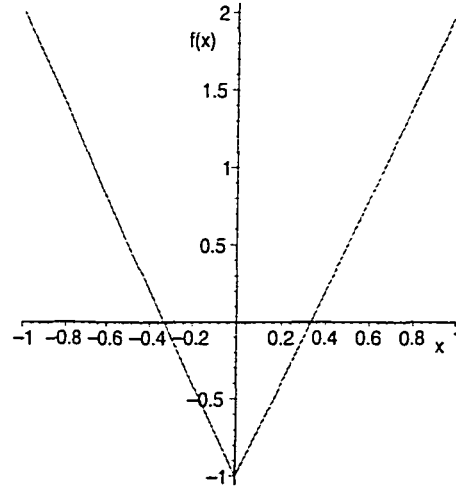


Figure 7.1: The graph of $f(x) = 3|x| - 1$ for $-1 \leq x \leq 1$.

Proposition: Let $f(x)$ be an integrable function on Ω_X . If $\int_{\Omega_X} f(x) dx = 1$ and $\int_{\Omega_X} |f(x)| dx = 1$, then $f(x) \geq 0$ almost everywhere.

Proof: Let $g(x) = |f(x)| - f(x) \geq 0$ for all $x \in \Omega_X$. Then $g(x)$ is a nonnegative integrable function on Ω_X . Since $\int_{\Omega_X} g(x) dx = \int_{\Omega_X} (|f(x)| - f(x)) dx = 0$, then by a standard measure theory theorem (Halmos, 1950, page 104), $g(x) = 0$ almost everywhere. Thus, $f(x) = |f(x)| \geq 0$ almost everywhere.

The following examples illustrate how **VerifyPDF** handles various continuous random variables.

Example 7.1. (Casella & Berger, 1990, page 43) Prove that the PDF of the random variable X with CDF given by

$$F(x) = \frac{1}{2} + \frac{\tan^{-1}(x)}{\pi} \quad -\infty < x < \infty$$

is a valid PDF.

Solution: It is not necessary that X is in its PDF representation for the `VerifyPDF` procedure to correctly determine if its PDF is valid since the procedure calls `PDF` to convert the random variable's representation initially. The APPL statements

```
> X := [[x -> 1 / 2 + arctan(x) / Pi], [-infinity, infinity],
        ["Continuous", "CDF"]];
> VerifyPDF(X);
```

print the following:

The area under $f(x)$ is 1,

$f(x)$ is nonnegative.

The PDF of the given random variable

$$\left[\left[x \rightarrow \frac{1}{\pi(1+x^2)} \right], [-\infty, \infty], ["Continuous", "PDF"] \right]$$

is valid.

□

Example 7.2. (Bain & Engelhardt, 1992, page 85) Determine whether each of the following functions is a valid CDF over the indicated part of the domain.

(a) $F(x) = e^{-x}$ for $0 \leq x < \infty$;

(b) $F(x) = 1 - e^{-x}$ for $-1 \leq x < \infty$.

Solution: If a random variable's PDF is not valid, then its CDF is not valid. The `VerifyPDF` procedure can be used to "weed out" random variables with invalid PDFs, which also have invalid CDFs.

For example (a), the APPL statements

```
> X := [[x -> exp(-x)], [0, infinity], ["Continuous", "CDF"]];
> VerifyPDF(X);
```

print the following:

*The PDF of the given random variable
is NOT valid because $f(x)$ is negative for some value x
in its support.*

For example (b), the APPL statements

```
> X := [[x -> 1 - exp(-x)], [-1, infinity], ["Continuous", "CDF"]];
> VerifyPDF(X);
```

print the following:

The area under $f(x)$ is 2.718281828.

The PDF of the given random variable is NOT valid. □

There are instances in which $\int_{-\infty}^{\infty} f(x) dx = 1$ and $\int_{-\infty}^{\infty} |f(x)| dx = 1$, but due to roundoff error `VerifyPDF` incorrectly determines that the random variable X has a valid PDF. As computer algebra systems become more sophisticated and powerful in their numerical evaluation methods, these types of errors will occur less frequently.

Example 7.3. (APPL trap) Let X be a continuous random variable with PDF

$$f(x) = 1.00002|x - 1| - 0.00001 \quad 0 \leq x \leq 2.$$

Show graphically that X has an invalid PDF, although `VerifyPDF` does not indicate this.

Solution: Figure 7.2, which displays the PDF of X for $0.9999 \leq x \leq 1.0001$, is created with the APPL statements

```
> X := [[x -> 1.00002 * abs(x - 1) - 0.00001], [0, 2],
        ["Continuous", "PDF"]];
> PlotDist(X, 0.9999, 1.0001);
```

As can be seen in the figure, $f(0) < 0$, and thus X does not have a valid PDF. Since Maple evaluates $\int_0^2 |1.00002 \cdot |x - 1| - 0.00001| dx$ as one, then the APPL statement

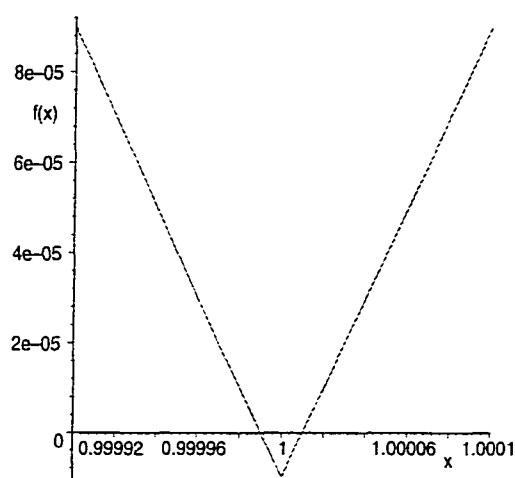


Figure 7.2: The graph of $f(x) = 1.0002|x - 1| - 0.0001$ for $0.9999 \leq x \leq 1.0001$.

> VerifyPDF(X)

incorrectly determines that X has a valid PDF. As Maple's abilities progress, APPL's will also! □

7.2 Method of Moments Estimation

This section presents the MOM procedure for estimating parameters via the method of moments. This method calculates the estimates of the unknown parameters by equating the first k theoretical moments of a random variable X to their corresponding sample moments, where k is the number of unknown parameters.

Let X_1, X_2, \dots, X_n be a random sample of size n from a distribution with PDF $f_X(x; \theta_1, \theta_2, \dots, \theta_k)$. The first k moments of a random variable X , if they exist, are found by computing the expectation

$$\mu_{(j)} = E(X^j), \quad j = 1, 2, \dots, k.$$

The first k sample moments are found by computing

$$M_{(j)} = \sum_{i=1}^n (x_i^j/n), \quad j = 1, 2, \dots, k,$$

where x_1, x_2, \dots, x_n are the data values.

Given that $f_X(x; \theta_1, \theta_2, \dots, \theta_k)$ is a suitable model for estimating the population distribution, then $M_{(j)}$ should be approximately equal to the corresponding $\mu_{(j)}$ for $j = 1, 2, \dots, k$. Thus, a general procedure for estimating the parameters $\theta_1, \theta_2, \dots, \theta_k$ is to solve the system of equations

$$\mu_{(j)} = M_{(j)}$$

for $j = 1, 2, \dots, k$. The solutions to these equations are called the *method of moments estimates*.

7.2.1 Implementation

The APPL procedure `MOM(X, Sample, Parameters)` used to compute the method of moments estimates is implemented as follows:

- The procedure is presented with three arguments:
 - **X**: A random variable (written in the APPL list-of-sublists format) with PDF $f_X(x; \theta_1, \theta_2, \dots, \theta_k)$,
 - **Sample**: A list of sample data points drawn from the distribution with PDF $f_X(x; \theta_1, \theta_2, \dots, \theta_k)$, and
 - **Parameters**: A list of parameters to be estimated.
- The procedure checks that the appropriate number of arguments are entered in their indicated formats. That is, **X** must be entered as a list-of-sublists, the

Sample data must be entered as a list, and Parameters must be entered as a list.

- After converting X to its PDF form, if necessary, the procedure checks that the Parameter list contains the same variable names assigned to the random variable X . That is, if $X := \text{GammaRV}(a, b)$, then the Parameter list must be entered as $[a, b]$. The variables being estimated in the Parameter list must match the distribution's parameters for the Maple `solve` procedure to correctly equate and solve the sample and theoretical moments for the appropriate parameters.
- In order to return exact solutions instead of floating point approximations (whenever possible), the procedure converts the values in the list `Sample` to rational numbers.
- The procedure computes and simplifies the sample and theoretical distribution moments. In order to compute the theoretical moments, the procedure calls the APPL `ExpectedValue` procedure, which was presented in Chapter 2.
- If possible, the procedure uses `solve` to find the exact solution(s) to the simultaneous system of equations obtained from equating the theoretical moments with their corresponding sample moments. If Maple cannot determine the exact solution with `solve`, then the procedure sends the equations to Maple's numeric solver, `fsolve`.
- Finally, the procedure returns the method of moment parameter estimates as a list. If the estimates have been solved by Maple's numeric solver, a message is displayed along with the estimates to indicate that `fsolve` was used.

7.2.2 Examples

This subsection contains two applications of the MOM procedure. The first example estimates the parameters for a continuous distribution, the gamma distribution. The second example finds parameter estimates for an exponential distribution and a Weibull distribution that are fit to the same data set. This example takes advantage of Maple's numeric solver, `fsolve`. One pitfall of `fsolve` is encountered in this example, though a correct parameter estimation can be found by making a slight adjustment in the MOM procedure. Example 1.3 in Chapter 1 used MOM to estimate the single parameter for a Poisson distribution.

Example 7.4. (Larsen & Marx, 2001, pages 319–322) Although hurricanes generally strike only the eastern and southern coastal regions of the United States, they do occasionally sweep inland before completely dissipating. The U.S. Weather Bureau confirms that in the period from 1900 to 1969 a total of 36 hurricanes moved as far as the Appalachians. Table 7.1 lists the maximum 24-hour precipitation levels recorded from those 36 storms during the time they were over the mountains.

A histogram of the data suggests that the random variable X , which is the maximum 24-hour precipitation, might be well approximated by the gamma distribution with PDF

$$f_X(x; \lambda, \kappa) = \frac{\lambda(\lambda x)^{\kappa-1} e^{-\lambda x}}{\Gamma(\kappa)} \quad x > 0; \lambda > 0; \kappa > 0.$$

In this example, λ and κ are the parameters to be estimated.

The following APPL statements define X as a gamma random variable, assign the hurricane data to the list `Hurricane`, and assign the parameters to be estimated to the list `Pars`. Then, `MOM(X, Hurricane, Pars)` assigns the method of moments estimates for the parameters λ and κ as a list to `HurricanePars`.

```
> X := GammaRV(lambda, kappa);
> Hurricane := [31.00, 2.82, 3.98, 4.02, 9.50, 4.50, 11.40, 10.71,
```


Table 7.1: Maximum 24-hour precipitation for 36 inland hurricanes (1900–1969).

Year	Name	Location	Maximum Precipitation (inches)
1969	Camille	Tye River, Va.	31.00
1968	Candy	Hickley, N.Y.	2.82
1965	Betsy	Haywood Gap, N.C.	3.98
1960	Brenda	Cairo, N.Y.	4.02
1959	Gracie	Big Meadows, Va.	9.50
1957	Audrey	Russels Point, Ohio	4.50
1955	Connie	Slide Mt., N.Y.	11.40
1954	Hazel	Big Meadows, Va.	10.71
1954	Carol	Eagles Mere, Pa.	6.31
1952	Able	Bloserville 1-N, Pa.	4.95
1949		North Ford #1, N.C.	5.64
1945		Crossnore, N.C.	5.51
1942		Big Meadows, Va.	13.40
1940		Rhodhiss Dam, N.C.	9.72
1939		Caesars Head, S.C.	6.47
1938		Hubbardston, Mass.	10.16
1934		Balcony Falls, Va.	4.21
1933		Peekamoose, N.Y.	11.60
1932		Caesars Head, S.C.	4.75
1932		Rockhouse, N.C.	6.85
1929		Rockhouse, N.C.	6.25
1928		Roanoke, Va.	3.42
1928		Caesars Head, S.C.	11.80
1923		Mohonk Lake, N.Y.	0.80
1923		Wappingers Falls, N.Y.	3.69
1920		Landrum, S.C.	3.10
1916		Altapass, N.C.	22.22
1916		Highlands, N.C.	7.43
1915		Lookout Mt., Tenn.	5.00
1915		Highlands, N.C.	4.58
1912		Norcross, Ga.	4.46
1906		Horse Cove, N.C.	8.00
1902		Sewanee, Tenn.	3.73
1901		Linville, N.C.	3.50
1900		Marrobone, Ky.	6.20
1900		St. Johnsbury, Vt.	0.67

6.31, 4.95, 5.64, 5.51, 13.40, 9.72, 6.47, 10.16, 4.21, 11.60,
 4.75, 6.85, 6.25, 3.42, 11.80, 0.80, 3.69, 3.10, 22.22, 7.43,
 5.00, 4.58, 4.46, 8.00, 3.73, 3.50, 6.20, 0.67];
 > Pars := [lambda, kappa];
 > HurricanePars := MOM(X, Hurricane, Pars);

The resulting estimates for the parameters are $\hat{\lambda} = \frac{954000}{4252153} \cong 0.224$ and $\hat{\kappa} = \frac{6952275}{4252153} \cong 1.64$. □

Example 7.5. (Leemis, 1995, page 190) A complete data set of $n = 23$ ball bearing failure times to test the endurance of deep-groove ball bearings has been extensively

studied. The ordered set of failure times measured in 10^6 revolutions is

17.88	28.92	33.00	41.52	42.12	45.60	48.48	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	173.40.	

Let X be a random variable denoting the ball bearing failure times.

- (a) Assume X is an exponential random variable with failure rate λ . Use MOM to estimate λ .
- (b) Assume X is a Weibull distribution with parameters λ and κ . Use MOM to estimate the parameters' values.

Solution: For part (a), the following APPL statements unassign the variable name λ , define X as an exponential random variable, and assign the parameters to be estimated to the list `Pars`. The data set for the failure times, `BallBearing`, is a pre-defined list in APPL. `MOM(X, BallBearing, Pars)` assigns the method of moments estimate for the parameter λ as a list to `ExpBallBearingPar`.

```
> unassign('lambda');
> X := ExponentialRV(lambda);
> Pars := [lambda];
> ExpBallBearingPar := MOM(X, BallBearing, Pars);
```

The resulting method of moments estimate for the parameter is $\hat{\lambda} = \frac{575}{41534} \cong 0.0138$ failures per million revolutions.

The graphs of the empirical CDF along with the CDF for the fitted exponential distribution are displayed in Figure 7.3. The graphs suggest that the exponential distribution provides a poor fit for this data set. The APPL procedure for producing the graph in this figure is discussed in Chapter 9.

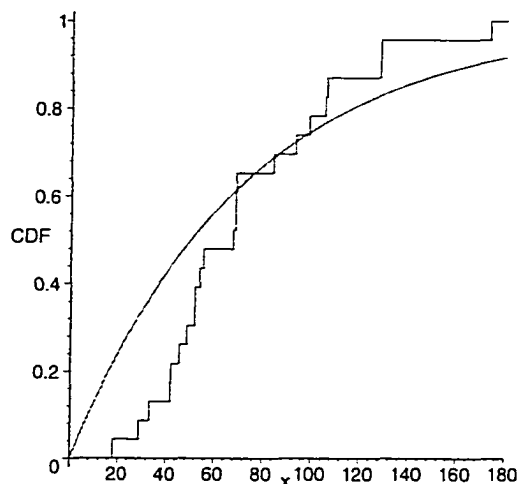


Figure 7.3: Empirical and fitted exponential CDFs for the ball bearing data set.

The Weibull distribution provides a much better approximation for this data. The PDF of the Weibull distribution is

$$f_X(x; \lambda, \kappa) = \kappa \lambda^\kappa x^{\kappa-1} e^{-(\lambda x)^\kappa} \quad x > 0; \lambda > 0; \kappa > 0.$$

Again using the ball bearing data set, we can find the method of moments estimates for the parameters λ and κ using the following APPL statements

```
> unassign('lambda'); unassign('kappa');
> X := WeibullRV(lambda, kappa);
> WeibBallBearingPar := MOM(X, BallBearing, [lambda, kappa]);
```

In this case, MOM informs the user that a numerical method was used to solve for the values of the parameters λ and κ . The numerical approximations are $\hat{\lambda} \cong 0.0176$ and $\hat{\kappa} \cong -3.55$. Although these are correct solutions to the simultaneous system of equations obtained from equating the theoretical moments with their corresponding sample moments, they are incorrect parameter estimates for the Weibull distribution since we need $\lambda > 0$ and $\kappa > 0$. Unfortunately, the Maple `fsolve` procedure searches for the first real root for a general equation, then quits (Heal et al., 1998, page 69). Often a plot of the simultaneous equations can suggest the general vicinity of other

real roots, and then `fsolve` can be used again with a specified range. To obtain the correct parameter estimates, the `fsolve` command line in MOM was changed to specify the range of another real root. Instead of using the general `fsolve` procedure: `fsolve(EqnSet, ParamSet)`, the `fsolve` procedure was used with a solution range: `fsolve(EqnSet, ParamSet, lambda = 0 .. 0.015)`. The new correct parameter estimates returned are $\hat{\lambda} \cong 0.0123$ and $\hat{\kappa} \cong 2.07$.

7.3 Maximum Likelihood Estimation with Right Censoring

This section presents the MLE procedure for estimating parameters via maximum likelihood estimation. Let X be a random variable with PDF $f_X(x; \theta)$, where θ is a vector of k unknown parameters, i.e., $\theta = (\theta_1, \theta_2, \dots, \theta_k)$. Suppose X_1, X_2, \dots, X_n is a random sample drawn from the population with PDF $f_X(x; \theta)$. Maximum likelihood estimation estimates the unknown parameter θ with a value $\hat{\theta}$ that maximizes the “likelihood” of obtaining that particular random sample.

The *likelihood function*, $L(\theta)$, for a given set of observations, x_1, x_2, \dots, x_n , from the population with PDF $f_X(x; \theta)$ is the product of the PDF $f_X(x; \theta)$ evaluated at the n sample data points, i.e.,

$$L(\theta) = \prod_{i=1}^n f_X(x_i; \theta). \quad (7.1)$$

The *maximum likelihood estimator* $\hat{\theta}$ is found by maximizing $L(\theta)$ with respect to θ . The parameter estimate $\hat{\theta}$ is the value that is most likely to have produced the sample data points x_1, x_2, \dots, x_n .

In practice, it is often easier to maximize the log likelihood function $\log L(\theta)$ to

determine $\hat{\theta}$, which is valid since the logarithm function is strictly increasing. The *log likelihood function* is

$$\log L(\theta) = \sum_{i=1}^n \log f_X(x_i; \theta). \quad (7.2)$$

If $L(\theta)$ is differentiable and assumes a maximum on the parameter space, then the MLE is a solution to

$$\frac{d}{d\theta} \log L(\theta) = 0.$$

Example 7.6. (Leemis, 1995, page 190) Returning to the ball bearing data set in Example 7.5, the ordered set of failure times measured in 10^6 revolutions is

17.88 28.92 33.00 41.52 42.12 45.60 48.48 51.84
 51.96 54.12 55.56 67.80 68.64 68.64 68.88 84.12
 93.12 98.64 105.12 105.84 127.92 128.04 173.40.

Let X be a random variable denoting the ball bearing failure times and assume X is an exponential random variable with failure rate λ . Estimate the parameter λ by maximum likelihood estimation.

Solution: The log likelihood function for λ is

$$\log L(\lambda) = 23 \log \lambda - \lambda \sum_{i=1}^{23} x_i.$$

Differentiating both sides of this equation and solving for λ , the maximum likelihood estimator $\hat{\lambda}$ is

$$\begin{aligned} \hat{\lambda} &= \frac{23}{\sum_{i=1}^{23} x_i} \\ &= \frac{23}{1661.16} \\ &\cong 0.0138, \end{aligned}$$

which is same as the method of moments estimator for this data set. \square

7.3.1 Right-Censored Data Sets

Suppose n items are being tested and their failure times t_1, t_2, \dots, t_n are being recorded. If the test stops before an item has failed, only a lower bound for the failure time is known. These failure times are *right-censored* data values. Right censoring occurs frequently in lifetime data sets since it is often impossible, impractical, and/or infeasible (because of time, money, energy, etc.) to continue running a test until all items on the test have failed. If a data set contains one or more censored observations, it is called a *censored data set*. Otherwise, if all the failure times are known, it's called a *complete data set*.

Following the notation and language used by Leemis (1995, pages 184–186), let t_1, t_2, \dots, t_n be the independent failure times collected during a test. Let c_1, c_2, \dots, c_n be the associated right-censored times. Let $x_i = \min\{t_i, c_i\}$, $i = 1, 2, \dots, n$. We can then split the indexes of the data items $1, 2, \dots, n$ into two disjoint sets: U and C . The set U contains the indexes of the items that are observed to fail during the test, and the set C contains the indexes of the items whose failure times are right-censored.

If θ is the vector of unknown parameters, we can rewrite the likelihood function in equation 7.1 with respect to the indexes of the observed failures and the right-censored observations:

$$L(\theta) = \left(\prod_{i \in U} f_X(x_i; \theta) \right) \cdot \left(\prod_{i \in C} S_X(x_i; \theta) \right),$$

where $S_X(x_i; \theta)$ is the probability that item i survives to time x_i . The log likelihood function is

$$\log L(\theta) = \sum_{i \in U} \log f_X(x_i; \theta) + \sum_{i \in C} \log S_X(x_i; \theta).$$

Since the PDF $f_X(x)$ is the product of the hazard function $h_X(x)$ and survivor func-

tion $S_X(x)$, the log likelihood function can be simplified to

$$\begin{aligned}\log L(\theta) &= \sum_{i \in U} \log h_X(x_i; \theta) + \sum_{i \in U} \log S_X(x_i; \theta) + \sum_{i \in C} \log S_X(x_i; \theta) \\ &= \sum_{i \in U} \log h_X(x_i; \theta) + \sum_{i=1}^n \log S_X(x_i; \theta).\end{aligned}$$

We can rewrite the log likelihood function in terms of the hazard and cumulative hazard functions only:

$$\log L(\theta) = \sum_{i \in U} \log h_X(x_i, \theta) - \sum_{i=1}^n H_X(x_i, \theta). \quad (7.3)$$

Example 7.7. (Leemis, 1995, page 190) The set of remission times for the treatment group in the study concerning the drug 6-MP (Gehan, 1965) is a right-censored data set. Letting an asterisk denote a right-censored observation, the remission times (in weeks) are

6 6 6 6* 7 9* 10 10* 11* 13 16
17* 19* 20* 22 23 25* 32* 32* 34* 35*.

Let $X \sim \text{exponential}(\lambda)$ be used to model the remission time data. Use maximum likelihood estimation to determine the value of the parameter λ .

Solution: Since there are $n = 21$ individuals on the test, nine uncensored observations, $h_X(x) = \lambda$, and $H_X(x) = \lambda x$, then the log likelihood function for λ is

$$\log L(\lambda) = 9 \log \lambda - \lambda \sum_{i=1}^{21} x_i.$$

Taking the derivative of the log likelihood function with respect to λ , equating it to zero, and solving for λ , we obtain $\lambda \cong 0.0251$. □

7.3.2 Implementation

The APPL procedure, `MLE(X, Data, Parameters, [Rightcensor])`, is used to compute the maximum likelihood estimators for the parameters (in the list `Parameters`) of a random variable X given a sample data set (`Data`) from the distribution's population. An optional argument, `Rightcensor`, allows for data values to be right-censored. The argument `Rightcensor` is a list of ones and zeros, corresponding to the data values in the list `Data`. The value one in position i of the `Rightcensor` list indicates that the data value in position i of the `Data` list is an observed value. A zero indicates a right-censored value. The procedure is implemented as follows:

- The procedure is called with either three or four arguments. If there are three arguments, the MLEs are determined using the log likelihood formula in equation 7.2.
- If there are four arguments, then the procedure assumes there are right-censored values in the `Data` list. (If there are no right-censored observations, i.e., there are only zeros in the list `Rightcensor`, the MLEs are just computed using the log likelihood function in equation 7.3. The `HF` and `CHF` procedures are used to determine the hazard and cumulative hazard functions of X . The log likelihood function in equation 7.3 is used to determine the MLEs.
- As in the `MOM` procedure, `MLE` uses `solve` to find the exact solution(s) to the simultaneous system of differentiated log likelihood functions (with respect to the unknown parameters in the `Parameters` list) and the unknown parameters. If Maple cannot determine the exact solution with `solve`, then the procedure sends the equations to Maple's numeric solver, `fsolve`.
- Finally, the procedure returns the maximum likelihood parameter estimates as a list. If the estimates have been solved by Maple's numeric solver, a message

is printed along with the estimates to indicate that `fsolve` was used.

Example 7.6. Revisited (APPL solution) The APPL statements

```
> X := ExponentialRV(lambda);
> lamhat := MLE(X, BallBearing, [lambda]);
```

return $\hat{\lambda} \cong 0.0138$ as the maximum likelihood estimator. □

Example 7.7. Revisited (APPL solution) Both `MP6` and `MP6Censor` are pre-defined data sets in APPL. The list `MP6` is simply a list of the 21 data values given in Example 7.7, and `MP6Censor` is the list

```
MP6Censor := [1, 1, 1, 0, 1, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0]
```

where 0 represents a censored value and 1 represents an uncensored value. The statements used to determine the MLE for the exponential distribution are

```
> X := ExponentialRV(lambda);
> hat := MLE(X, MP6, [lambda], MP6Censor);
```

The statements yield $\hat{\lambda} = \frac{9}{359}$. □

7.4 Mixture and Truncate Procedures

Chapter 8 is about Benford's law and determining which probability distributions conform to it. The two procedures introduced in this section, along with the Benford procedure described in Chapter 8, were originally written to aid in making this determination. The `Mixture` and `Truncate` procedures are described in the following subsections.

7.4.1 Mixture

A population may contain items gathered from several different populations, each with a distinct lifetime distribution. A car mechanic, for example, may have a part that is manufactured in one of four facilities, but he is not certain in which one. In a *finite mixture model*, items are assumed to come from one of n populations. The failure time distribution of the item can be expressed in terms of a mixture of each item's population distribution.

The PDFs of *mixture distributions*, also called *compound distributions*, can be expressed as weighted sums of the PDFs of the component distributions. This APPL procedure, `Mixture`, is written for *finite mixtures*, as described in the previous paragraph. The PDF of a general finite mixture random variable X is

$$f_X(x) = \sum_{i=1}^n p_{X_i} f_{X_i}(x|\theta_{X_i}),$$

where $f_{X_i}(x|\theta_{X_i})$ is the PDF for the random variable X_i from population i , θ_{X_i} is a vector of parameters for the distribution of X_i , and p_{X_i} is the *mix parameter* for population X_i , $i = 1, 2, \dots, n$. Note that $p_{X_i} > 0$, for $i = 1, 2, \dots, n$ and $\sum_{i=1}^n p_{X_i} = 1$.

The `Mixture(MixParameters, MixRVs)` procedure “mixes” the random variables X_1, X_2, \dots, X_n defined in the list `MixRVs` by taking weighted sums defined in the list `MixParameters`. Two examples of the `Mixture` procedure follow.

Example 7.8. (Leemis, 1995, page 118) If $n = 2$ facilities produce items with `exponential(1)` and `exponential(2)` lifetimes, respectively, and one-third of the items come from facility one and two-thirds come from facility 2, determine the PDF of the time to failure of an item whose manufacturing site is unknown.

Solution: Let $X_1 \sim \text{exponential}(\lambda_1)$ and $X_2 \sim \text{exponential}(\lambda_2)$, where $\lambda_1 = 1$ and $\lambda_2 = 2$. Let X be the time to failure for the item from the unknown manufacturing

site. The PDF of X is

$$\begin{aligned} f_X(x) &= p_{X_1}f_{X_1}(x|\lambda_1) + p_{X_2}f_{X_2}(x|\lambda_2) \\ &= \frac{1}{3}e^{-x} + \frac{4}{3}e^{-2x} \quad x \geq 0, \end{aligned}$$

which is a finite mixture of the populations from distributions X_1 and X_2 . This model is a special case of the *hyperexponential* distribution, which is the finite mixture of n exponential populations.

The following APPL statements return the PDF of the above model

```
> X := ExponentialRV(1);
> Y := ExponentialRV(2);
> p := [1 / 3, 2 / 3];
> Mixture(p, [X, Y]);
```

□

Example 7.9. Let $X_1 \sim \text{triangular}(1, 2, 3)$, $X_2 \sim \text{triangular}(1, 2, 4)$, and $X_3 \sim \text{triangular}(2, 5, 7)$. Let $p_{X_1} = \frac{1}{8}$, $p_{X_2} = \frac{5}{8}$, and $p_{X_3} = \frac{1}{4}$ be the probabilities of selecting an item from the distributions associated with the random variables X_1 , X_2 , and X_3 . Find the PDF of Z , the finite mixture of the three distributions.

Solution: This example forces the **Mixture** procedure to return a PDF defined on more than a single segment of support. The following APPL statements

```
> X1 := TriangularRV(1, 2, 3);
> X2 := TriangularRV(1, 2, 4);
> X3 := TriangularRV(2, 5, 7);
> Z := Mixture([1 / 8, 5 / 8, 1 / 4], [X1, X2, X3]);
```

returns the PDF of Z as

$$f_Z(z) = \begin{cases} \frac{13}{24}z - \frac{13}{24} & 1 \leq z < 2 \\ \frac{137}{129} - \frac{3}{10}z & 2 \leq z < 3 \\ \frac{23}{30} - \frac{7}{40}z & 3 \leq z \leq 4 \\ \frac{1}{30}z - \frac{1}{15} & 4 \leq z < 5 \\ \frac{7}{20} - \frac{1}{20}z & 5 \leq z < 7. \end{cases}$$

7.4.2 Truncate

Let X be a random variable with PDF $f_X(x)$ on Ω_X . Then for $a, b \in \Omega_X$, the PDF of the *doubly truncated* (i.e., truncated below at a and above at b) random variable T is

$$f_T(t) = \frac{f_X(t)}{F_X(b) - F_X(a)} \quad a \leq t \leq b,$$

provided $F_X(b) - F_X(a) \neq 0$. The `Truncate(X, low, high)` procedure returns the PDF of the random variable X truncated below at `low` and above at `high`. Three examples of the `Truncate` procedure follow.

Example 7.10. (Rohatgi, 1976, page 119) Let X be a random variable with PDF $f(x) = 1$ if $0 \leq x \leq 1$, and 0 otherwise. Let T be the random variable formed by truncating X below at $1/3$ and above at $1/2$. Find the PDF of the truncated distribution T , its mean, and its variance.

Solution: The PDF of T is

$$f_T(t) = 6 \quad \frac{1}{3} \leq t \leq \frac{1}{2}.$$

Its mean is $\frac{5}{12}$, while its variance is $\frac{1}{432}$. The following APPL statements determine the PDF, mean, and variance of T :

```
> U := UniformRV(0, 1);
```

```
> T := Truncate(U, 1 / 3, 1 / 2);
> Mean(T);
> Variance(T);
```

□

Example 7.11. (Barr & Sherrill, 1999, page 359) (Truncated normal distribution)
 Let $T \sim N(1, 2)$ random variable truncated below at 2. Determine the mean of T .
 (The standard deviation, not the variance, of T is two.)

Solution: Let $N(\mu, \sigma)$ denote a normal random variable with mean μ and standard deviation σ . If T is a $N(\mu, \sigma)$ random variable truncated below by a , then $(T - \mu)/\sigma = Z$ is a standard normal random variable truncated below by $a^* = (a - \mu)/\sigma$. The expected value of T in terms of the expected value of Z is

$$E(T) = \sigma E(Z) + \mu.$$

For our example, $T \sim N(1, 2)$ truncated below at $a = 2$, and thus

$$E(T) = 2 \cdot E(Z) + 1,$$

where $Z \sim N(0, 1)$ truncated below at $a^* = 1/2$. The density of Z is

$$f_Z(z) = \frac{e^{(-z^2/2)}\sqrt{2}}{\sqrt{\pi}(-1 + \operatorname{erf}(\sqrt{2}/4))}$$

where “erf” is the Maple procedure defined by $\operatorname{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2} dt$. The mean of Z is

$$E(Z) = \frac{\sqrt{2}e^{(-1/8)}}{\sqrt{\pi}(-1 + \operatorname{erf}(\sqrt{2}/4))},$$

which is approximately 1.141078. Hence, $E(T) = 2 \cdot E(Z) + 1 \cong 3.282156$. This mean is computed in APPL with the statements

```
> X := NormalRV(1, 2);
> T := Truncate(X, 2, infinity);
> Mean(T);
```

□

Example 7.12. (Hogg & Craig, 1995, page 146) Let $\phi(x)$ and $\Phi(x)$ be the PDF and the CDF of a standard normal random variable X . Let Y be the random variable formed by truncating X below at -2 and above at 3 . Show that $E(Y) = \frac{\phi(-2) - \phi(3)}{\Phi(3) - \Phi(-2)}$.

Solution: We can use the Truncate procedure to determine $E(Y)$ with the following statements:

```
> X := StandardNormalRV();
> Y := Truncate(X, -2, 3);
> ExpectedValue(Y);
```

which returns the value $\frac{\sqrt{2}(e^{-2} - e^{-9/2})}{\sqrt{\pi}(\operatorname{erf}(\frac{3}{2}\sqrt{2}) + \operatorname{erf}(\sqrt{2}))} \cong 0.0508$, where erf is the Maple procedure defined by $\operatorname{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2} dt$. Writing this expression in terms of ϕ and Φ , first notice that $\phi(-2) - \phi(3) = \frac{e^{-2} - e^{-9/2}}{\sqrt{2\pi}}$. Then, converting the Maple expression in the denominator, $\operatorname{erf}(\frac{3}{2}\sqrt{2}) + \operatorname{erf}(\sqrt{2})$, into its integral form, we have:

$$\begin{aligned} \operatorname{erf}\left(\frac{3}{2}\sqrt{2}\right) + \operatorname{erf}(\sqrt{2}) &= \frac{2}{\sqrt{\pi}} \left[\int_0^{\frac{3}{2}\sqrt{2}} e^{-w^2} dw + \int_0^{\sqrt{2}} e^{-w^2} dw \right] \\ &= 2 \left(\frac{1}{\sqrt{\pi}} \int_0^3 \frac{e^{-w^2/2}}{\sqrt{2}} dw + \frac{1}{\sqrt{\pi}} \int_0^2 \frac{e^{-w^2/2}}{\sqrt{2}} dw \right) \\ &= 2 \left(\frac{1}{\sqrt{2\pi}} \int_0^3 e^{-w^2/2} dw + \frac{1}{\sqrt{2\pi}} \int_0^2 e^{-w^2/2} dw \right) \\ &= 2 \left(\frac{1}{\sqrt{2\pi}} \int_{-2}^3 e^{-w^2/2} dw \right) \\ &= 2 \left(\frac{1}{\sqrt{2\pi}} \int_{-\infty}^3 e^{-w^2} dw - \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{-2} e^{-w^2/2} dw \right) \\ &= 2(\Phi(3) - \Phi(-2)). \end{aligned}$$

Thus, the expected value of Y is

$$\begin{aligned} \frac{\sqrt{2}(e^{-2} - e^{-9/2})}{\sqrt{\pi}(\operatorname{erf}(\frac{3}{2}\sqrt{2}) + \operatorname{erf}(\sqrt{2}))} &= \left(\frac{\sqrt{2} [\sqrt{2\pi}(\phi(-2) - \phi(3))]}{\sqrt{\pi} [2(\Phi(3) - \Phi(-2))]} \right) \\ &= \frac{\phi(-2) - \phi(3)}{\Phi(3) - \Phi(-2)}. \end{aligned}$$

□

Chapter 8

Survival Distributions Satisfying Benford's Law

Benford's law has traditionally concerned the distribution of the leading digit for a data set. This chapter quantifies compliance with Benford's law for several popular survival distributions. The traditional analysis of Benford's law considers its applicability to *data sets*. This chapter switches the emphasis to *probability distributions* that obey Benford's law.

8.1 Benford's Law

Astronomer and mathematician Simon Newcomb noticed “how much faster the first pages (of tables of logarithms) wear out than the last ones” leading to the counter-intuitive conclusion that the first significant digit in the values in a logarithm table is not uniformly distributed between 1 and 9. Using a heuristic argument, he found that ones occur most often (more than 30 percent of the time) and nines least often (less than 5 percent of the time). More specifically, if the random variable X denotes

the first significant digit, then

$$\Pr(X = x) = \log_{10} (1 + 1/x) \quad x = 1, 2, \dots, 9.$$

He published this “logarithm law” in the *American Journal of Mathematics* in 1881.

General Electric physicist Frank Benford (1938) apparently independently arrived at the same conclusion as Newcomb concerning logarithm tables. He proceeded to “collect data from as many fields as possible” to see if natural and sociological data sets would also obey the logarithm law. He often found good agreement between the logarithm law for his 20,229 total observations, including data sets as diverse as the areas of rivers, American League baseball statistics, atomic weights of elements, death rates, and numbers appearing in *Reader’s Digest*.

What has become known as “Benford’s law” has found applications in the distribution of the one-day return on stock market indices (Ley, 1996), the distribution of the populations of 3141 counties in the 1990 U.S. Census, and the detection of accounting fraud (Nigrini, 1996).

A mathematically rigorous proof of Benford’s law has proven elusive. This is in part due to the fact that certain data sets (e.g., random numbers) do not follow Benford’s law. Recent attempts have considered the effect of scale invariance (e.g., dollars vs. yen), base invariance (e.g., octal vs. base ten), and mixtures (i.e., sample data drawn from several population distributions that are selected at random), as indicated in Hill (1995, 1998).

The purpose here is to switch the emphasis from the examination of *data sets* that obey Benford’s law to *probability distributions* that obey Benford’s law. Survival distributions (i.e., random variables with positive support) will be emphasized here, although more general distributions can be examined in the same fashion.

8.2 Parametric Survival Distributions

Hill (1995, pages 361–362) states that “An interesting open problem is to determine which common distributions (or mixtures thereof) satisfy Benford’s law ...”. This section quantifies compliance with Benford’s law for several popular survival distributions.

As before, let X denote a random variable having Benford’s distribution, and let T denote a random lifetime with SF $S(t) = \Pr(T \geq t)$. If Y is the value of the first significant digit in the lifetime T , then

$$\Pr(Y = y) = \sum_{i=-\infty}^{\infty} [S(y \cdot 10^i) - S((y+1)10^i)]$$

for $y = 1, 2, \dots, 9$. Thus $\Pr(Y = 7)$, for example, is found by summing the appropriate probabilities on the intervals

$$\dots, (0.07, 0.08), (0.7, 0.8), (7, 8), (70, 80), \dots$$

More detailed examples on the derivation of the probability mass function of Y are given in Section 8.3.

For a particular random variable T having prescribed survivor function $S(t)$, it is desired to measure the goodness-of-fit between Benford’s distribution and the distribution of the first significant digit. Two such measures are the chi-square goodness-of-fit statistic

$$c = \sum_{x=1}^9 \frac{[\Pr(Y = x) - \Pr(X = x)]^2}{\Pr(X = x)}$$

and

$$m = \max_{x=1,2,\dots,9} \{|\Pr(Y = x) - \Pr(X = x)|\}.$$

These measures are calculated for several popular lifetime distributions in Table 8.1

Table 8.1: Conformance to Benford's law for parametric survival distributions.

Distribution	λ	κ	Class	c	m
Exponential	1		IFR/DFR	$0.61 \cdot 10^{-2}$	$0.29 \cdot 10^{-1}$
Exponential	5		IFR/DFR	$0.54 \cdot 10^{-2}$	$0.18 \cdot 10^{-1}$
Muth		0.1	IFR	$0.13 \cdot 10^{-1}$	$0.41 \cdot 10^{-1}$
Gompertz	5	1.1	IFR	$0.62 \cdot 10^{-2}$	$0.20 \cdot 10^{-1}$
Weibull	1	0.3	DFR	$0.37 \cdot 10^{-10}$	$0.16 \cdot 10^{-5}$
Weibull	1	2	IFR	0.19	0.11
Gamma	1	0.3	DFR	$0.15 \cdot 10^{-3}$	$0.29 \cdot 10^{-2}$
Gamma	1	2	IFR	$0.48 \cdot 10^{-1}$	$0.50 \cdot 10^{-1}$
Log logistic	1	0.3	DFR	$0.86 \cdot 10^{-21}$	$0.67 \cdot 10^{-11}$
Log logistic	1	2	UBT	$0.24 \cdot 10^{-1}$	$0.35 \cdot 10^{-1}$
Exponential Power Distribution	1	0.3	BT	$0.48 \cdot 10^{-4}$	$0.17 \cdot 10^{-2}$

as parameterized in Leemis (1995, Chapter 4). Appendix F contains APPL code for computing the distribution of Y for the unit exponential distribution and the Benford distribution X for one significant digit.

The following observations were made while constructing the table:

- The results for the exponential distribution for $\lambda = 5$, for example, are also good for $\lambda = 5 \cdot 10^k$, for $k = \pm 1, \pm 2, \dots$
- For all distributions considered with a shape parameter κ , the goodness-of-fit measures c and m increased in κ for the values of κ considered.
- For all two-parameter distributions, the goodness-of-fit measures c and m were more sensitive to changes in the shape parameter κ than the scale parameter λ .

Notice that for the log logistic distribution with $\lambda = 1$ and $\kappa = 0.3$, there is an astonishing 11-digit agreement with Benford's law. The fact that the PDF of the logarithm of a log logistic random variable is symmetric might provide a clue as to why it matched Benford's law so closely.

General conditions associated with the distribution of the random variable T will now be derived in order to determine when Benford's law applies.

8.3 Conditions for Conformance to Benford's Law

As stated earlier, the PDF of a Benford random variable X is

$$f_X(x) = \Pr(X = x) = \log_{10}(1 + 1/x),$$

for $x = 1, 2, \dots, 9$. The associated CDF is

$$F_X(x) = \Pr(X \leq x) = \log_{10}(1 + x),$$

for $x = 1, 2, \dots, 9$. Inverting the CDF, a Benford variate X can be generated by

$$X \leftarrow \lceil 10^U - 1 \rceil,$$

or

$$X \leftarrow \lfloor 10^U \rfloor,$$

where $U \sim U(0, 1)$.

As before, let T be the random lifetime whose first significant digit is of interest. Let the integer-valued random variable D satisfy

$$10^D \leq T < 10^{D+1}$$

(e.g., $T = 365 \Rightarrow D = 2$ and $T = 1/10 \Rightarrow D = -1$). This definition of D allows the first significant digit Y to be written in terms of T and D as

$$Y = \lfloor T \cdot 10^{-D} \rfloor = \lfloor 10^{\log_{10} T - D} \rfloor$$

(e.g., $T = 365 \Rightarrow Y = \lfloor 365 \cdot 10^{-2} \rfloor = \lfloor 3.65 \rfloor = 3$).

Referring back to the variate generation algorithm, it is clear that if the random variable $Z = \log_{10} T - D \sim U(0, 1)$, which represents the result from the logarithm table, then the first significant digit Y has the Benford distribution. Using conditioning, the CDF of Z is given by

$$\begin{aligned} F_Z(z) &= \Pr(Z \leq z) \\ &= \sum_{d=-\infty}^{\infty} \Pr(10^d \leq T < 10^{d+1}) \cdot \Pr(\log_{10} T - d \leq z | 10^d \leq T < 10^{d+1}), \end{aligned}$$

for $0 < z < 1$. Thus conformance to Benford's law implies that the weights (the first term in the product) associated with each order of the magnitude and the distribution of $Z = \log_{10} T - D$ (the second term in the product) are such that the infinite sum produces a linear function in z .

Why was Newcomb surprised? He expected each page of a logarithm table to be equally worn; i.e., he surmised that the values that people used as arguments in logarithm tables would be uniformly distributed between 1.0 and 10.0. Although the left-hand column of a logarithm table is arranged in a linear fashion so that 1.0 to 2.0 requires $\frac{1}{9}$ of the pages, Newcomb correctly observed that the people using the tables in 1881 did not use them in a uniform fashion (e.g., over 30% of the table look-ups were from the first $\frac{1}{9}$ of the pages). In summary, Newcomb expected uniformity in the inputs to the logarithm tables, but uniformity was actually achieved in the resultant logarithms, represented by Z .

We now proceed to investigate distributions that satisfy these conditions.

Example 8.1. A distribution can be created that satisfies Benford's law exactly. Let $W \sim U(0, 2)$. Let $T = 10^W$. The PDF of T is

$$f_T(t) = \frac{1}{2t \log 10}$$

for $1 < t < 100$. The probability mass function of D is

$$f_D(0) = \Pr(D = 0) = \Pr(1 < T < 10) = \int_1^{10} f_T(t) dt = \frac{1}{2}$$

and

$$f_D(1) = \Pr(D = 1) = \Pr(10 < T < 100) = \int_{10}^{100} f_T(t) dt = \frac{1}{2}.$$

The probability mass function of the leading digit Y is

$$\begin{aligned} f_Y(y) &= \Pr(Y = y) \\ &= \Pr(y < T < y + 1) + \Pr(10y < T < 10(y + 1)) \\ &= \int_y^{y+1} f_T(t) dt + \int_{10y}^{10(y+1)} f_T(t) dt \\ &= \int_y^{y+1} \frac{1}{2t \log 10} dt + \int_{10y}^{10(y+1)} \frac{1}{2t \log 10} dt \\ &= \log_{10} \left(\frac{y+1}{y} \right) \quad y = 1, 2, \dots, 9. \end{aligned}$$

This probability mass function matches Benford's distribution exactly.

Alternatively, one can proceed by determining the distribution of $Z = \log_{10} T - D$, where $W = \log_{10} T$.

$$\begin{aligned} F_Z(z) &= \Pr(Z \leq z) \\ &= \sum_{d=-\infty}^{\infty} \Pr(10^d \leq T < 10^{d+1}) \cdot \Pr(\log_{10} T - d \leq z | 10^d \leq T < 10^{d+1}) \\ &= \sum_{d=0}^1 \Pr(d \leq W < d+1) \cdot \Pr(W - d \leq z | d \leq W < d+1) \\ &= \Pr(0 \leq W < 1) \cdot \Pr(W \leq z | 0 \leq W < 1) + \\ &\quad \Pr(1 \leq W < 2) \cdot \Pr(W - 1 \leq z | 1 \leq W < 2) \\ &= \frac{1}{2}z + \frac{1}{2}z \\ &= z \end{aligned}$$

for $0 < z < 1$. Since this is the CDF for a $U(0, 1)$ random variable, Benford's law is satisfied exactly. \square

The previous example can be generalized as follows. Let $W \sim U(a, b)$, where a and b are real numbers satisfying $a < b$. If the interval $(10^a, 10^b)$ covers an integer number of orders of magnitude, then the first significant digit of the random variable $T = 10^W$ satisfies Benford's law exactly. Equivalently, if $b - a$ is a positive integer, then the first significant digit of $T = 10^W$ satisfies Benford's law. Examples include $a = -2, b = 1$ and $a = \log_{10}(3/2), b = \log_{10}(150)$.

There is no need for the support of the distribution of $T = 10^W$ to span several orders of magnitude as is the case for many of the data sets that conform to Benford's law. Example 8.1 shows that a single order of magnitude (e.g., $a = 5, b = 6$) is sufficient.

The next example considers a non-uniform distribution for W .

Example 8.2. Let $W \sim \text{triangular}(0, 1, 2)$. The PDF for W is

$$f_W(w) = \begin{cases} w & 0 < w < 1 \\ 2 - w & 1 \leq w < 2. \end{cases}$$

As before, let $T = 10^W$ and $Z = W - D$. The cumulative distribution function of Z is

$$\begin{aligned} F_Z(z) &= \sum_{d=0}^1 \Pr(d \leq W < d+1) \cdot \Pr(W - d \leq z | d \leq W < d+1) \\ &= \frac{1}{2}z^2 + \frac{1}{2}(2z - z^2) \\ &= z \quad 0 < z < 1. \end{aligned}$$

Thus the first significant digit of T satisfies Benford's law exactly. \square

This example can also be generalized. Let $W \sim \text{triangular}(a, b, c)$, where a, b , and c are real numbers satisfying $a < b < c$. The first significant digit of the random variable $T = 10^W$ satisfies Benford's law exactly if a, b , and c are integers.

The symmetric, integer-parameter triangular distribution's conformance to Benford's law may provide some insight into the log logistic's stellar performance in Table 1. If the PDF of W is symmetric about an integer and the variance of W is large, then it is often the case that the PDF of W is approximately linear between consecutive integers. The symmetric portions of the PDF of W will nearly cancel one another when computing the distribution of Z . A normal random variable W with integer mean μ and large standard deviation σ , for example, corresponds to a lognormal $T = 10^W$ whose first significant digit closely approximates Benford's law.

The next example considers a non-symmetric distribution for W .

Example 8.3. Let W have PDF

$$f_W(w) = \begin{cases} 1 - w^2 & -1 < w < 0 \\ (w - 1)^2 & 0 \leq w < 1. \end{cases}$$

As before, let $T = 10^W$ and $Z = W - D$. The cumulative distribution function of Z is

$$\begin{aligned} F_Z(z) &= \sum_{d=-1}^0 \Pr(d \leq W < d+1) \cdot \Pr(W - d \leq z | d \leq W < d+1) \\ &= \frac{2}{3} \left[-\frac{z^3}{2} + \frac{3z^2}{2} \right] + \frac{1}{3} [1 + (z-1)^3] \\ &= z \quad 0 < z < 1. \end{aligned}$$

Thus the first significant digit of T satisfies Benford's law exactly. \square

This example can be generalized for W with probability density function

$$f_W(w) = \begin{cases} 1 - w^n & -1 < w < 0 \\ (w - 1)^n & 0 \leq w < 1, \end{cases}$$

where n is a positive, even integer.

We wanted to experiment with several other probability distributions in order to evaluate conformance to Benford's law. In order to automate this process, we wrote the APPL `Benford` procedure, whose argument is the distribution of W and whose returned value is the distribution of Z . The algorithm is shown below.

```

Ω ← Support(W)           [The set Ω is the support of the random variable W]
Lo ← ⌊Ω⌋                 [Lower loop limit]
Hi ← ⌈Ω⌉ - 1             [Upper loop limit]
Weight ← Array[1 .. Hi - Lo + 1] [Weight holds the mixture probabilities]
TransfW ← Array[1 .. Hi - Lo + 1] [TransfW holds W's transformed segments]

For d ← Lo to Hi
  Weight[d] ← Fw(d + 1) - Fw(d) [Calculate weights for the mixture]
  TruncW[d] ← Truncate(W, d, d + 1) [Truncate W between d and d + 1]
  TransfW[d] ← Transform(TruncW, w - d) [Horizontally shift W by d units]
Z ← Mixture(Weight, TransfW) [Compute the distribution of the mixture]

```

The statements required to return the distribution of Z for the triangular distribution in Example 8.2, for instance, are

```

W := TriangularRV(0, 1, 2);
Z := Benford(W);

```

After experimenting with `Benford` on other distributions, we have come to the following conclusions:

1. Distributions of W with a single mode that occurs at either extreme of their support will never satisfy Benford's law (e.g., $W \sim$ exponential).

2. Using a geometric argument, certain limiting distributions of W (e.g., $W \sim N(\mu, \sigma^2)$, where μ is an integer and $\sigma \rightarrow \infty$) will satisfy Benford's law.
3. Other distributions (e.g., Weibull) may come very close to satisfying Benford's law for various parameter values. Our experimentation revealed that compliance with Benford's law depends on parameter values within one particular parametric family. Thus using Benford's law to detect accounting fraud, for example, is dubious due to an unacceptably high rate of false positives.
4. For a random variable T that can assume negative values, all of the conclusions drawn here apply since the first digit of $|T|$ equals the first digit of T .
5. If W is a distribution such that the first significant digit of 10^W satisfies Benford's law, then the first significant digit of b^W satisfies Benford's law for base $b = 2, 3, \dots$
6. The distribution associated with the more general form of Benford's law

$$\Pr(\text{mantissa} < t) = \log_{10} t \quad 1 \leq t < 10,$$

where the mantissa of a real number is the number obtained from shifting the decimal point to the place immediately following the first significant (non-zero) digit, is *sum-invariant* (Allaart, 1997). A short proof of a generalization of Allaart's result appears below.

Result: Using our earlier notation, let $W \sim U(0, 1)$ and $T = 10^W$. Then the random variable T , with CDF given by $F_T(t) = \log_{10} t$ for $1 \leq t < 10$, is sum-invariant; i.e., if the interval $[1, 10)$ is equally partitioned by $h > 0$, then the expected sum of n random variates from this distribution in any given partitioned interval is the same for all intervals.

Proof: Let k be any natural number and set $h = \frac{9}{k}$. Without loss of generality, fix k . Let $A_j = [1 + (j - 1) \cdot h, 1 + j \cdot h) \subset \mathfrak{R}$ for $j = 1, 2, \dots, k$. The probability that T is in the interval A_j and the conditional expected value of T on the interval A_j for any $j = 1, 2, \dots, k$ are, respectively,

$$\begin{aligned} \Pr(1 + (j - 1) \cdot h \leq T < 1 + j \cdot h) &= \int_{1+(j-1) \cdot h}^{1+j \cdot h} \frac{1}{x \log(10)} dx \\ &= \frac{\log(1 + (j - 1) \cdot h) - \log(1 + j \cdot h)}{\log(10)} \end{aligned}$$

and

$$\begin{aligned} \mathbb{E}(T | 1 + (j - 1) \cdot h \leq T < 1 + j \cdot h) &= \int_{1+(j-1) \cdot h}^{1+j \cdot h} \frac{x}{x(\log(1 + (j - 1) \cdot h) - \log(1 + j \cdot h))} dx \\ &= \frac{h}{\log(1 + (j - 1) \cdot h) - \log(1 + j \cdot h)}. \end{aligned}$$

Thus, the expected sum of n variates in the interval A_j for any j is

$$n \cdot \mathbb{E}(T | 1 + (j - 1) \cdot h \leq T < 1 + j \cdot h) \cdot \Pr(1 + (j - 1) \cdot h \leq T < 1 + j \cdot h) = \frac{n \cdot h}{\log(10)},$$

Since this expected sum depends on k and is independent of j , the distribution of T is sum-invariant.

7. Any mixture of distributions that individually obey Benford's law will obey Benford's law. The case of two random variables satisfying Benford's law is proven below.

Result: Let T_1 and T_2 be nonnegative random variables whose first significant digits satisfy Benford's law. Let the random variable T have PDF

$$f_T(t) = p f_{T_1}(t) + (1 - p) f_{T_2}(t) \quad t > 0,$$

for $0 < p < 1$. Then T also satisfies Benford's law.

Proof: Let $Z_1 = \log_{10} T_1 - D$, $Z_2 = \log_{10} T_2 - D$, and $Z = \log_{10} T - D$. Since T_1 and T_2 satisfy Benford's law, then $F_{Z_1}(z) = z$ and $F_{Z_2}(z) = z$. In order to prove that T also satisfies Benford's law, we need to show that $F_Z(z) = z$. By conditioning on z , we have

$$\begin{aligned} F_Z(z) &= pF_{Z_1}(z) + (1-p)F_{Z_2}(z) \\ &= pz + (1-p)z \\ &= z \quad 0 < z < 1. \end{aligned}$$

8.4 Variate Generation

As stated earlier, variates from the Benford distribution can be generated via

$$X \leftarrow \lfloor 10^U \rfloor,$$

where $U \sim U(0, 1)$. Two variations of this algorithm can be developed by allowing different bases and multiple significant digits as described in the next two paragraphs.

Benford's law for the first significant digit in base b is associated with the PDF

$$f_X(x) = \Pr(X = x) = \log_b(1 + 1/x) \quad x = 1, 2, \dots, b-1$$

for $b = 2, 3, \dots$. Since the CDF is

$$F_X(x) = \Pr(X \leq x) = \log_b(1 + x) \quad x = 1, 2, \dots, b-1,$$

variates can be generated via

$$X \leftarrow \lfloor b^U \rfloor,$$

where $U \sim U(0, 1)$. When b is 2 (the binary case), for example, the X value generated is always 1, as expected.

When the first r digits are considered, Benford's law generalizes to

$$f_X(x) = \Pr(X = x) = \log_{10}(1 + 1/x) \quad x = 10^{r-1}, 10^{r-1} + 1, \dots, 10^r - 1$$

for $r = 1, 2, \dots$ [Note that this rather relaxed notation implies that $x = 365$ when $r = 3$ corresponds to a first digit $R_1 = 3$, second digit $R_2 = 6$, and third digit $R_3 = 5$, which occurs with probability $\Pr(X = 365) = \Pr(R_1 = 3, R_2 = 6, R_3 = 5) = \log_{10}(1 + 1/365)$.] The CDF is

$$\begin{aligned} F_X(x) &= \Pr(X \leq x) \\ &= \sum_{i=10^{r-1}}^x \log_{10}(1 + 1/i) \\ &= \sum_{i=10^{r-1}}^x \log_{10}(i + 1) - \log_{10} i \\ &= \log_{10}(x + 1) - \log_{10}(10^{r-1}) \\ &= \log_{10}\left(\frac{x + 1}{10^{r-1}}\right) \quad x = 10^{r-1}, 10^{r-1} + 1, \dots, 10^r - 1. \end{aligned}$$

Variates can be generated by inversion via

$$X \leftarrow \lfloor 10^{U-r+1} \rfloor,$$

where $U \sim U(0, 1)$.

Combining the previous two cases, a discrete Benford variate X associated with the first r significant digits in base b is generated by inversion via

$$X \leftarrow \lfloor b^{U-r+1} \rfloor,$$

where $U \sim U(0, 1)$.

8.5 Conclusions

Benford's law holds exactly for certain parametric survival distributions introduced in Section 8.3, holds to varying degrees for many other parametric distributions as shown in Section 8.2, and holds very poorly [e.g., for the number of children in a family in the U.S. or $T \sim U(3, 7)$ since the digits 1, 2, 7, 8, 9 never occur] for other distributions. The reason that Benford's law applies to so many data sets may simply be due to the fact that many popular parametric lifetime models also closely follow his law for particular values of their parameters.

Chapter 9

Input Modeling

Input modeling that involves fitting standard univariate parametric probability distributions is typically performed using an input modeling package, such as Arena, AweSim, Unifit, BestFit, or Stat::Fit (Swain, 2001). These packages typically fit several distributions to a data set, then determine the distribution with the best fit by comparing goodness-of-fit statistics. But what if an appropriate input model is not included in one of these packages? The modeler must resort to deriving estimators by hand for an appropriate input model. The purpose of this chapter is to investigate the use of APPL for input modeling. APPL allows an analyst to specify a standard or non-standard distribution for an input model, and have the derivations performed automatically. Input modeling serves as an excellent arena for illustrating the applicability and usefulness of APPL. It contains input modeling procedures for parameter estimation (as described in Chapter 7), plotting empirical and fitted CDFs, and performing goodness-of-fit tests. In this chapter, examples are used to exhibit APPL's utility for input modeling. Limitations of some procedures when applied to certain distributions (i.e., applying maximum likelihood to the Weibull distribution) and strategies for overcoming these obstacles are also discussed in this chapter.

9.1 Examples

Both APPL and Maple can easily be adapted for use in input modeling. This section provides seven examples of cases where a symbolic language is of use in analyzing a data set.

Example 9.1. (Model selection) One of the tools for selecting a suitable input model is a plot of the coefficient of variation ($\gamma = \sigma/\mu$) versus the skewness

$$\gamma_3 = E \left[\left(\frac{X - \mu}{\sigma} \right)^3 \right].$$

After constructing this plot, the sample coefficient of variation and sample skewness can be plotted for a particular data set or data sets to determine an appropriate distribution for modeling the data.

The APPL statements that produce the plot in Figure 9.1 for the Weibull, gamma, log normal, and log logistic distributions use the additional APPL `CoefOfVar` and `Skewness` procedures. The statements necessary to plot the gamma distribution's coefficient of variation versus skewness are shown below. The plots for the other distributions are calculated similarly. The Maple statement used to display all four plots in one graphic is also provided.

```
> unassign('kappa');
> lambda := 1;
> X := GammaRV(lambda, kappa);
> c := CoefOfVar(X);
> s := Skewness(X);
> GammaPlot := plot([c, s, kappa = 0.5 .. 999], labels = [cv, skew]):
.
.
.
> plots[display]({GammaPlot, WeibullPlot, LogNormalPlot,
  LogLogisticPlot}, scaling = unconstrained);
```

The `unassign` command in Maple is used to unassign any previous value given to an

existing variable name, such as κ . □

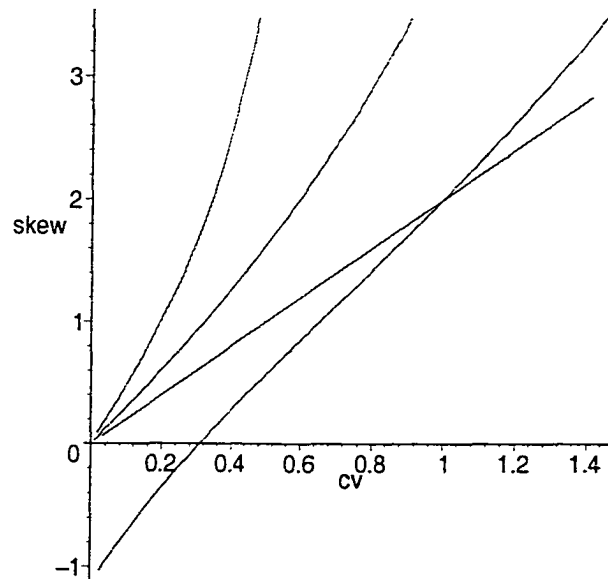


Figure 9.1: Coefficient of variation, γ , versus skewness, γ_3 , for the gamma, Weibull, log normal, and log logistic distributions.

Example 9.2. The $n = 23$ ball bearing failure times were collected to determine an input model in a discrete-event simulation of a reliability system. The failure times in 10^6 revolutions are

17.88	28.92	33.00	41.52	42.12	45.60	48.48	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	173.40.	

[Although these ball bearing failure times are from the life testing literature (Lawless 1982, page 228), the same analysis would apply to service times, for example.] In Example 7.5 we used the MOM procedure to determine the parameter estimates for fitting an exponential distribution and a Weibull distribution to this data set. Determine the model adequacy for these two distributions.

Solution: Figure 7.3 is a plot of the empirical and fitted CDFs for the ball bearing failure times and the exponential distribution. The `PlotEmpVsFittedCDF` procedure was written to provide a graphical means for comparing a data set's empirical CDF and its fitted CDF for various distributions. The APPL statements used to plot the empirical and fitted CDFs for the ball bearing failure times and the Weibull distribution (where the parameters for the Weibull distribution $\hat{\lambda} = 0.0123$ and $\hat{\kappa} = 2.07$ were computed in Example 7.5) in Figure 9.2 are

```
> X := WeibullRV(lambda, kappa);
> PlotEmpVsFittedCDF(X, BallBearing, [lambda = 0.0123, kappa = 2.07],
    0, 180);
```

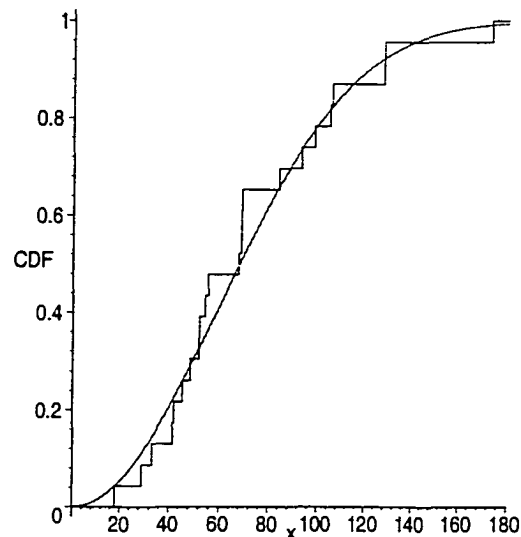


Figure 9.2: Empirical and fitted Weibull CDFs (using the method of moments) for the ball bearing data set.

In order to assess the model adequacy, either a formal goodness-of-fit test can be performed, or goodness-of-fit statistics can be compared for competing models. The Kolmogorov–Smirnov test statistic, for example, can be computed for both the fitted exponential and Weibull distributions. The `KSTest` procedure determines the

maximum vertical difference between the empirical distribution function and the fitted cumulative distribution function. For the fitted exponential distribution (where the parameter $\hat{\lambda} = 575/41534$ was computed in Example 7.5) and the ball bearing failure times, the APPL statements

```
> X := ExponentialRV(lambda);
> KSTest(X, BallBearing, [lambda = 575 / 41534]);
```

return 0.3068, indicating a rather poor fit. Similar APPL statements return the Kolmogorov–Smirnov test statistic for the fitted Weibull distribution as 0.1511. \square

As an alternative to fitting the exponential or Weibull distributions to the ball bearing failure times, one might consider fitting the *reciprocal* of an exponential random variable to the ball bearing failure times, as suggested in the following example. Part of the appeal in using APPL for input modelling is being able define non-standard distributions to fit to data sets.

Example 9.3. Fit the reciprocal of an exponential random variable to the ball bearing failure times in the previous example.

Solution: The APPL statements required to find the distribution of the reciprocal of an exponential random variable and find the MLE for the unknown parameter are

```
> X := ExponentialRV(lambda);
> g := [[x -> 1 / x], [0, infinity]];
> Y := Transform(X, g);
> lamhat := MLE(Y, BallBearing, [lambda]);
```

which return the PDF of Y as

$$f_Y(y) = \frac{\lambda}{y^2} e^{-\lambda/y} \quad y > 0$$

and calculate the MLE $\hat{\lambda} \cong 55.06$. The function g is used to find the distribution of

$$Y = g(X) = 1/X. \quad \square$$

As can be seen in Figure 9.3, the reciprocal of the exponential also provides a poor fit to the ball bearing failure times. Although the Weibull distribution provides a fairly good fit for the ball bearing failure times, it is not the best parametric model available in terms of the Kolmogorov–Smirnov goodness-of-fit statistic. It seems appropriate to consider another two-parameter distribution as a potential model, as shown in the next example.

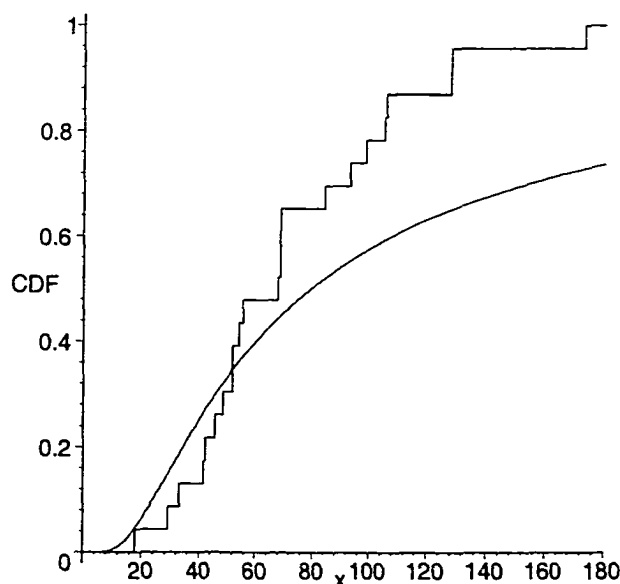


Figure 9.3: Empirical and reciprocal exponential fitted CDFs for the ball bearing failure times.

Example 9.4. Fit the inverse Gaussian distribution to the ball bearing failure times.

Solution: Again using the APPL MLE and KSTest procedures, the statements

```
> X := InverseGaussianRV(lambda, mu);
> hat := MLE(X, BallBearing, [lambda, mu]);
> KSValue := KSTest(X, BallBearing, [lambda = hat[1], mu = hat[2]]);
```

yields an improved fit with $\hat{\lambda} \cong 231.67$, $\hat{\mu} \cong 72.22$, and a Kolmogorov–Smirnov test statistic of 0.088. The statements `lambda = hat[1]` and `mu = hat[2]` assign the

values in the list `hat` to `lambda` and `mu`, respectively. The procedure `MLE` is able to return the appropriate values because the maximum likelihood estimators are in closed form for this particular distribution. This will not always be the case, as illustrated in Example 9.5 with the Weibull distribution. \square

Besides the procedures `PlotEmpVsFittedCDF` and `KSTest`, fit can be assessed visually using a Q–Q or P–P plot (Law and Kelton, 2000, pages 352–358). The APPL statements used to produce the Q–Q and P–P plots for the Weibull distribution and the ball bearing failure times data set displayed in Figures 9.4 and 9.5 are

```
> Y := WeibullRV(lambda, kappa);
> QQPlot(Y, BallBearing, [lambda = 0.0123, kappa = 2.07]);
> PPPlot(Y, BallBearing, [lambda = 0.0123, kappa = 2.07]);
```

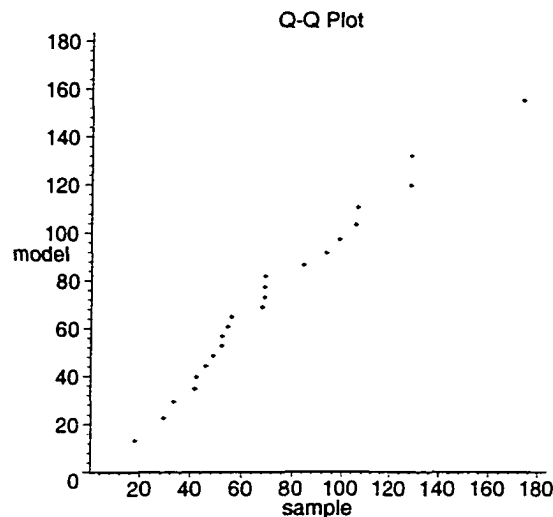


Figure 9.4: Q–Q plot of ball bearing failure times with fitted (method of moments) Weibull distribution.

To conclude the ball bearing failure times data analysis, Table 9.1 (on page 241) summarizes the Kolmogorov–Smirnov test statistic values for various distributions that were fit to the data via maximum likelihood estimation.

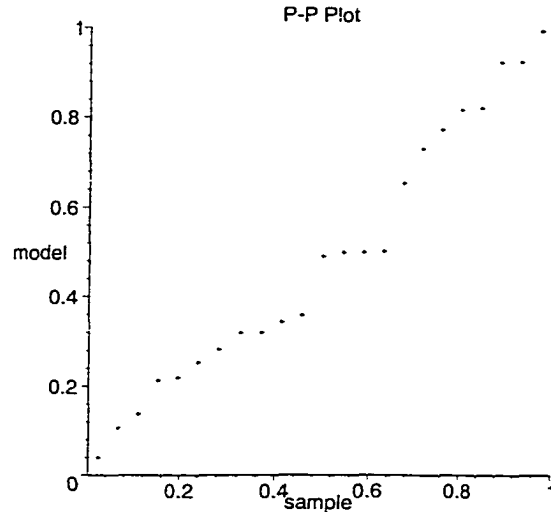


Figure 9.5: P–P plot of ball bearing failure times with fitted (method of moments) Weibull distribution.

Another wrinkle that can present itself in input modeling is the presence of censoring. A right-censored data set, for example, often occurs in reliability and biostatistical applications. Examples likely to arise in discrete-event input modeling situations include machine failure times (when some machines have not yet failed) and the analysis of rare events.

Example 9.5. Consider again the problem (introduced in Example 7.7) of determining an input model for the remission time for the treatment group in the study concerning the drug 6-MP (Gehan, 1965). Letting an asterisk denote a right-censored observation, the remission times (in weeks) are

6 6 6 6* 7 9* 10 10* 11* 13 16
17* 19* 20* 22 23 25* 32* 32* 34* 35*.

In this example, fit a Weibull distribution to the 6-MP data.

Solution: Both `MP6` and `MP6Censor` are pre-defined lists in APPL. `MP6` is simply the 21 data values given above, and `MP6Censor` is the list

[1, 1, 1, 0, 1, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0]

Table 9.1: Kolmogorov–Smirnov test statistic values for various distributions that were fit to the ball bearing failure times in APPL via maximum likelihood estimation.

Model	Test statistic
Exponential	0.307
Reciprocal of Exponential	0.306
Weibull	0.151
Gamma	0.123
Arctangent	0.094
Log normal	0.090
Inverse Gaussian	0.088

where 0 represents a censored value and 1 represents an uncensored value. Unfortunately, the statements

```
> Y := WeibullRV(lambda, kappa);
> hat := MLE(Y, MP6, [lambda, kappa], MP6Censor);
```

fail to return the MLEs in APPL. The Maple numerical equation solving procedure `fsolve` is not clever enough to exploit some of the structure in the score vector that is necessary to find the MLEs. Therefore a special routine, `MLEWeibull`, has been written that computes MLEs for the Weibull distribution. The additional statement

```
> hat := MLEWeibull(MP6, MP6Censor);
```

yields the MLE estimates $\hat{\lambda} \cong 0.03$ and $\hat{\kappa} \cong 1.35$ for the Weibull distribution. The Kaplan–Meier product-limit survivor function estimate for the MP6 data set, along with the fitted Weibull survivor function are plotted in Figure 9.6 using the additional APPL statement

```
> PlotEmpVsFittedSF(Y, MP6, [lambda = hat[1], kappa = hat[2]],
    MP6Censor, 0, 23);
```

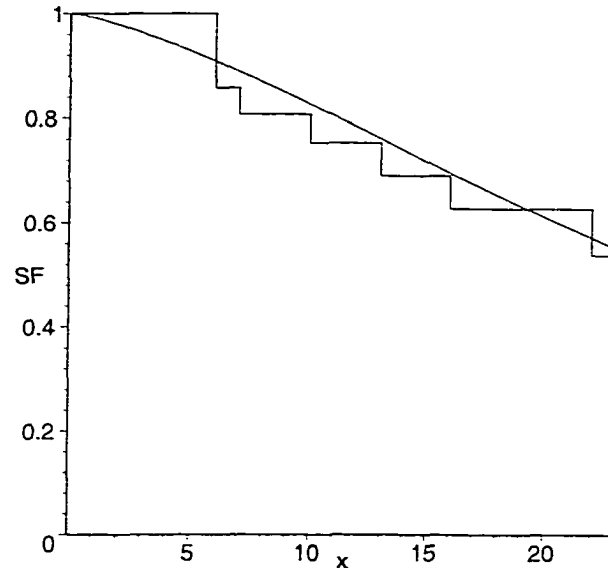


Figure 9.6: Product-limit survivor function estimate and fitted Weibull survivor function for the 6-MP treatment group.

The downward steps in the estimated survivor function occur only at observed remission times. The six parameters to the plotting function `PlotEmpVsFittedSF` are the random variable whose SF is to be plotted, the data values in a list, the parameters associated with the random variable, the right-censoring vector in a list, and the lower and upper plotting limits. Note that the product-limit estimator cuts off after the largest observed remission time (Lawless, 1982). \square

All of the input modeling examples thus far have been limited to continuous data. The next example fits the geometric distribution as a model for daily demand at a vending machine.

Example 9.6. A vending machine has capacity for 24 cans of “Purple Passion” grape drink. The machine is restocked to capacity every day at noon. Restocking time is negligible. The last five days have produced the following Purple Passion sales:

14 24 18 20 24.

The *demand* for Purple Passion at this particular vending machine can be estimated from the data by treating the 24-can sales figures as *right-censored* demand observations. If demand has the geometric distribution, with PDF

$$f(t) = p(1 - p)^t \quad t = 0, 1, 2, \dots$$

find the MLE for \hat{p} .

Solution: Although not discussed in Chapter 7, the MLE procedure can also handle discrete distributions. Since the pre-defined geometric distribution in APPL is parameterized for $t = 1, 2, \dots$, we need to define a geometric random variable with the different parameterization (used above) in the list-of-sublists data structure. No new APPL procedures are needed to compute the MLE for \hat{p} . The statements

```
> X := [[x -> p * (1 - p) ^ x], [0 .. infinity], ["Discrete", "PDF"]];
> PurplePass := [14, 24, 18, 20, 24];
> PurplePassCensor := [1, 0, 1, 1, 0];
> MLE(X, PurplePass, [p], PurplePassCensor);
```

yield $\hat{p} = \frac{3}{103}$. Model adequacy is not considered for this particular example. \square

All previous examples have considered time-independent observations. There are occasions when a series of event times may be time dependent, and a more complicated input model may be appropriate.

Example 9.7. Ignoring preventive maintenance, twelve odometer readings (from a certain model of car) associated with failures appearing over the first 100,000 miles are

12,942	28,489	65,561	78,254	83,639	85,603
88,143	91,809	92,360	94,078	98,231	99,900

Fit a nonhomogeneous Poisson process to the above data set, where the ending time of the observation interval is assumed to be 100,000 miles.

Solution: The data can be approximated by a power law process (i.e., the intensity function has the same parametric form as the hazard function for a Weibull random variable). The following APPL statements, including the additional MLENHPP procedure, return $\hat{\lambda} \cong 0.000026317$ and $\hat{\kappa} \cong 2.56800$:

```
> CarFailures := [12942, 28489, 65561, 78254, 83639, 85603, 88143,
  91809, 92360, 94078, 98231, 99900];
> X := WeibullRV(lambda, kappa);
> hat := MLENHPP(X, CarFailures, [lambda, kappa], 100000);
```

The last argument in MLENHPP tells the procedure that the failures were observed over the interval $[0, 100,000]$ miles. The additional APPL statement

```
> PlotEmpVsFittedCIF(X, Sample, [lambda = hat[1], kappa = hat[2]],
  0, 100000);
```

produces a plot of the empirical cumulative intensity function and the power law cumulative intensity function as shown in Figure 9.7. □

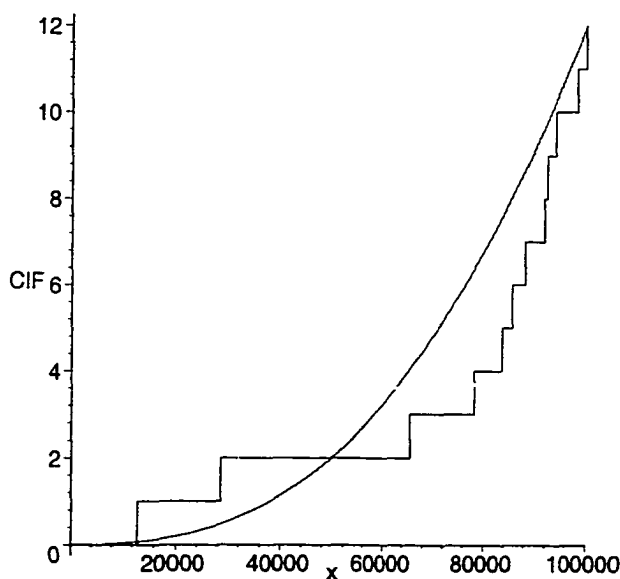


Figure 9.7: Cumulative intensity function estimate and fitted power law intensity function for the CarFailures data.

9.2 Further work

Some ongoing work in the area of input modeling in APPL is described here. First, most distributions containing 3 or 4 unknown parameters (e.g., the Johnson distributions) do not have closed-form maximum likelihood estimators. Based on our experience with the Weibull distribution in Example 9.5, it will be necessary to write custom code for many of these distributions. This is precisely what is required from the batch and interactive software packages that perform input modeling. Fortunately, there is significant literature concerning the numerical methods required to arrive at these estimators.

Second, some distributions, such as the Erlang distribution, have both a discrete and a continuous parameter. In order to compute parameter estimates, it is necessary to prove results that will expedite their calculation. In using maximum likelihood on the Erlang, for example, it would not be possible to calculate the MLEs for the scale parameter for all shape parameters in the parameter space. Thus some results concerning the monotonicity of the likelihood function as the shape parameter varies are necessary to provide an algorithm for calculating the MLEs.

Third, some distributions have their unknown parameters as part of their support. Consider finding the MLEs for the triangular(a, b, c) distribution for a sample size of $n = 2$. Without loss of generality, assume $x_1 < x_2$. Symmetry dictates that

$$\hat{b} = \frac{x_1 + x_2}{2}$$

and that $\hat{b} - \hat{a} = \hat{c} - \hat{b}$. Thus the problem of finding the MLE for a , for example, reduces to maximizing

$$f(x_1; a) = \frac{2(x_1 - a)}{(c - a)(b - a)} = \frac{x_1 - a}{(b - a)^2}.$$

Differentiating with respect to a yields

$$\frac{\partial f}{\partial a} = \frac{-(b-a)^2 + 2(x_1 - a)(b-a)}{(b-a)^4}.$$

When the derivative is equated to zero and the resulting equation is solved for a , the MLE is

$$\hat{a} = 2x_1 - \hat{b}.$$

Likewise,

$$\hat{c} = 2x_2 - \hat{b}.$$

Moving to the case of $n = 3$ is more complicated since it is not clear whether the middle data value should have its likelihood function considered part of the left or the right support of the PDF. An algorithm must be developed in order to compute the MLEs for general n .

APPL is a platform which can be used for input modeling in an interactive, as opposed to a batch platform. Its ability to interface with probability theory presents some advantages for calculating exact probability measures.

Chapter 10

APPLications

10.1 Kolmogorov–Smirnov Test Statistic for Estimated Parameters

The Kolmogorov–Smirnov (K–S) test compares a hypothetical or fitted CDF $\hat{F}(x)$ with an empirical CDF $F_n(x)$ in order to assess fit. The empirical CDF $F_n(x)$ is defined as

$$F_n(x) = \frac{\text{number of } X_i\text{'s } \leq x}{n},$$

where n is the size of the random sample, which means $F_n(x)$ is the proportion of the observations that are less than or equal to x . The K–S test statistic D_n is the *largest* vertical distance between $F_n(x)$ and $\hat{F}(x)$ for all values of x , i.e.,

$$D_n = \sup_x \{|F_n(x) - \hat{F}(x)|\}.$$

The statistic D_n can be computed by calculating (Law & Kelton, 2000, page 364)

$$D_n^+ = \max_{1 \leq i \leq n} \left\{ \frac{i}{n} - \hat{F}(X_{(i)}) \right\}, \quad D_n^- = \max_{1 \leq i \leq n} \left\{ \hat{F}(X_{(i)}) - \frac{i-1}{n} \right\}$$

and letting

$$D_n = \max\{D_n^+, D_n^-\}.$$

Although the test statistic D_n is easy to calculate, its distribution is mathematically intractable. Drew et al. (2000) provide an algorithm for calculating the CDF of D_n when all the parameters of the hypothetical CDF $\hat{F}(x)$ are known (referred to as the “all parameters known” case). Assuming that \hat{F} is continuous, the distribution of D_n is a function of n , but does not depend on \hat{F} . This “all parameters known” case has been previously coded into APPL as the KSRV procedure.

The more common and practical situation occurs when the parameters are unknown and are estimated from the sample data, using a technique such as maximum likelihood estimation. In this case, the distribution of D_n depends upon both n and the particular distribution that is being fit to the data. This section presents the derivation of the distribution of D_n for the case of exponential sampling for $n = 1$ and $n = 2$. Future work involves extending the pattern established for $n = 1$ and $n = 2$ for the exponential distribution to larger samples and other population distributions.

Let X be an exponential random variable with PDF $f(x) = \frac{1}{\theta}e^{-x/\theta}$ and CDF $F(x) = 1 - e^{-x/\theta}$ for $x > 0$. If x_1, x_2, \dots, x_n are the sample data values, then the MLE $\hat{\theta}$ is

$$\hat{\theta} = \frac{1}{n} \sum_{i=1}^n x_i.$$

10.1.1 D_1 for the Exponential Distribution

If there is only $n = 1$ sample data value, which we will call x_1 , then $\hat{\theta} = x_1$. Thus, the fitted CDF is

$$\hat{F}(x) = 1 - e^{-x/\hat{\theta}} = 1 - e^{-x/x_1} \quad x > 0.$$

As can be seen in Figure 10.1, the *largest* vertical distance between $F_1(x)$ and $\hat{F}(x)$ occurs at x_1 and it the value $D_1 = 1 - 1/e$. The PDF for D_1 is degenerate at $1 - 1/e$:

$$f_{D_1}(d_1) = 1 \quad d_1 = 1 - 1/e.$$

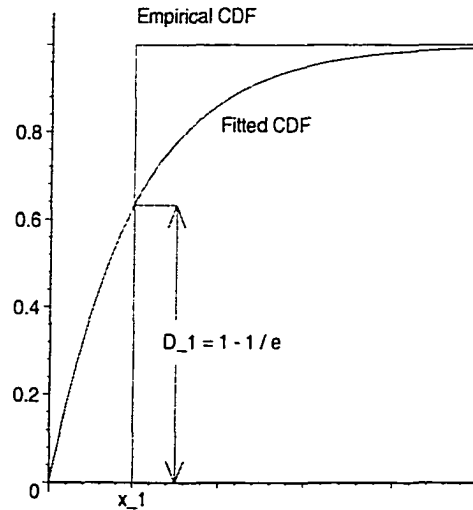


Figure 10.1: The empirical and fitted exponential distribution for one data value x_1 . The K-S test statistic value $D_1 = 1 - 1/e$ is pictured.

10.1.2 D_2 for the Exponential Distribution

Order the $n = 2$ sample data values and let $x_{(1)} = \min\{x_1, x_2\}$ and $x_{(2)} = \max\{x_1, x_2\}$.

The MLE is $\hat{\theta} = (x_{(1)} + x_{(2)})/2$ and the fitted CDF is

$$\hat{F}(x) = 1 - e^{-x/\hat{\theta}} = 1 - e^{-2x/(x_{(1)}+x_{(2)})} \quad x > 0.$$

Let

$$y = \frac{x_{(1)}}{x_{(2)}}$$

where $0 < y \leq 1$ since $0 < x_{(1)} \leq x_{(2)}$. The fitted CDF $\hat{F}(x)$ at the values $x_{(1)}$ and $x_{(2)}$ is

$$\hat{F}(x_{(1)}) = 1 - e^{-2x_{(1)}/(x_{(1)}+x_{(2)})} = 1 - e^{-2y/(y+1)} \quad 0 < y \leq 1,$$

and

$$\hat{F}(x_{(2)}) = 1 - e^{-2x_{(2)}/(x_{(1)}+x_{(2)})} = 1 - e^{-2/(y+1)} \quad 0 < y \leq 1.$$

The fitted CDF $\hat{F}(x)$ always intersects the second riser of the empirical CDF $F_2(x)$ since $\hat{F}(x_{(2)})$ ranges from $1 - \frac{1}{e} \cong 0.6321$ (when $y = 1$) to $1 - \frac{1}{e^2} \cong 0.8647$ (when $y = 0$). The fitted CDF *may* intersect the first riser of the empirical CDF depending on the value of y . For $0 < y \leq \frac{\log 2}{2 - \log(2)} \cong 0.5304$, the first riser is intersected. For $\frac{\log 2}{2 - \log(2)} < y \leq 1$, the fitted CDF lies entirely above the first riser.

Define lengths A , B , C and D according to the diagram in Figure 10.2. With respect to $y = x_{(1)}/x_{(2)}$, the lengths A , B , C and D (as functions of y) are

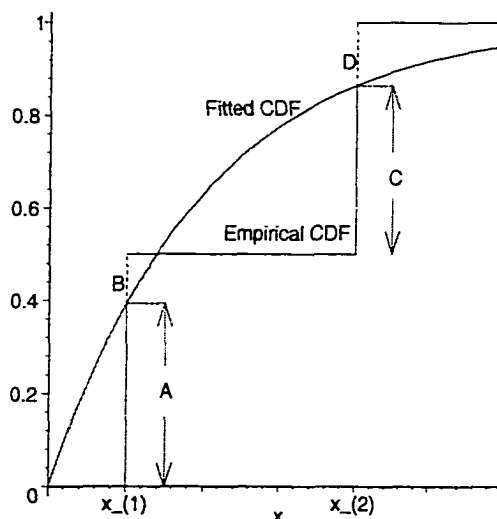


Figure 10.2: The empirical and fitted exponential distribution for two data values $x_{(1)}$ and $x_{(2)}$. In this particular plot, $0 < y \leq \frac{\log 2}{2 - \log(2)}$, so the first riser of the empirical CDF $F_2(x)$ is intersected by $\hat{F}(x)$.

$$A(y) = |(1 - e^{-2y/(y+1)}) - 0| = |1 - e^{-2y/(y+1)}| = 1 - e^{-2y/(y+1)} \quad 0 < y \leq 1;$$

$$B(y) = \left| \frac{1}{2} - (1 - e^{-2y/(y+1)}) \right| = \left| e^{-2y/(y+1)} - \frac{1}{2} \right|.$$

The length B is defined piecewise as

$$B(y) = \begin{cases} e^{-2y/(y+1)} - \frac{1}{2} & 0 < y \leq \frac{\log(2)}{2-\log(2)} \\ \frac{1}{2} - e^{-2y/(y+1)} & \frac{\log(2)}{2-\log(2)} < y \leq 1; \end{cases}$$

$$C(y) = \left| (1 - e^{-2/(y+1)}) - \frac{1}{2} \right| = \frac{1}{2} - e^{-2/(y+1)} \quad 0 < y \leq 1;$$

$$D(y) = |1 - (1 - e^{-2/(y+1)})| = e^{-2/(y+1)} \quad 0 < y \leq 1.$$

Figure 10.3 is a graph of the lengths A , B , C and D plotted with respect to $0 < y \leq 1$. For any given y on $(0, 1]$, the K-S test statistic is $D_2 = \max\{A, B, C, D\}$. Thus, only $C(y)$, $A(y)$, and the first piece of $B(y)$ are needed to define D_2 in terms of y . In addition, there are two y values of interest in Figure 10.3:

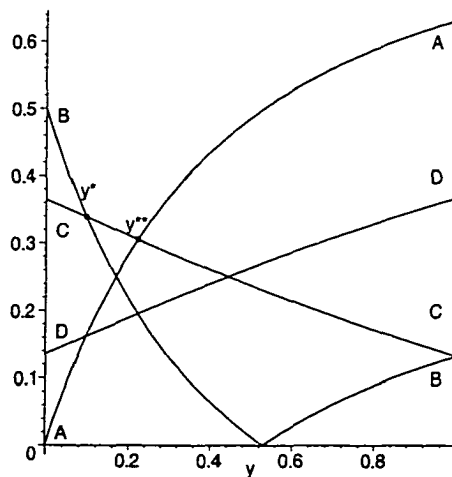


Figure 10.3: Lengths A , B , C , and D from Figure 10.2 for $0 < y \leq 1$.

- The value y^* such that $B(y) = C(y)$. Using Maple's solve procedure, we determine that

$$y^* = -\frac{2 + \log\left(\frac{1}{2} - \frac{1}{2}\sqrt{1 - 4/e^2}\right)}{\log\left(\frac{1}{2} - \frac{1}{2}\sqrt{1 - 4/e^2}\right)} \cong 0.0965,$$

and

$$C(y^*) = \exp\left(-2 - \log\left(\frac{1}{2} - \frac{1}{2}\sqrt{1 - 4/e^2}\right)\right) - \frac{1}{2} \cong 0.3386.$$

- The value y^{**} such that $A(y) = C(y)$. Using Maple's solve procedure, we determine that

$$y^{**} = -\frac{2 + \log\left(-\frac{1}{4} + \frac{1}{4}\sqrt{1 + 16/e^2}\right)}{\log\left(-\frac{1}{4} + \frac{1}{4}\sqrt{1 + 16/e^2}\right)} \cong 0.2226,$$

and

$$C(y^{**}) = 1 - \exp\left(-2 - \log\left(-\frac{1}{4} + \frac{1}{4}\sqrt{1 + 16/e^2}\right)\right) \cong 0.3052.$$

Thus, the largest vertical distance D_2 is either computed using the length formula for $A(Y)$, $B(Y)$, or $C(Y)$ depending on the value $Y = X_{(1)}/X_{(2)}$, i.e.,

$$D_2 = \begin{cases} B(Y) & 0 < Y \leq -\frac{2 + \log\left(\frac{1}{2} - \frac{1}{2}\sqrt{1 - 4/e^2}\right)}{\log\left(\frac{1}{2} - \frac{1}{2}\sqrt{1 - 4/e^2}\right)} \\ C(Y) & -\frac{2 + \log\left(\frac{1}{2} - \frac{1}{2}\sqrt{1 - 4/e^2}\right)}{\log\left(\frac{1}{2} - \frac{1}{2}\sqrt{1 - 4/e^2}\right)} < Y \leq -\frac{2 + \log\left(-\frac{1}{4} + \frac{1}{4}\sqrt{1 + 16/e^2}\right)}{\log\left(-\frac{1}{4} + \frac{1}{4}\sqrt{1 + 16/e^2}\right)} \\ A(Y) & -\frac{2 + \log\left(-\frac{1}{4} + \frac{1}{4}\sqrt{1 + 16/e^2}\right)}{\log\left(-\frac{1}{4} + \frac{1}{4}\sqrt{1 + 16/e^2}\right)} < Y \leq 1, \end{cases}$$

or, equivalently,

$$D_2 = \begin{cases} B(Y) & 0 < Y \leq y^* \\ C(Y) & y^* < Y \leq y^{**} \\ A(Y) & y^{**} < Y \leq 1. \end{cases}$$

Determining the Distribution of $Y = X_{(1)}/X_{(2)}$

Let X_1, X_2 be a random sample drawn from a population having PDF

$$f(x) = \frac{1}{\theta} e^{-x/\theta} \quad x > 0,$$

for $\theta > 0$. In order to determine the distribution of D_2 , we must determine the distribution of $X_{(1)}/X_{(2)}$, where

$$X_{(1)} = \min\{X_1, X_2\}, \text{ and}$$

$$X_{(2)} = \max\{X_1, X_2\}.$$

Using an order statistic result (Hogg & Craig, 1995, page 199) the joint PDF of $X_{(1)}$ and $X_{(2)}$ is

$$\begin{aligned} f(x_{(1)}, x_{(2)}) &= 2! \cdot \frac{1}{\theta} e^{-x_{(1)}/\theta} \cdot \frac{1}{\theta} e^{-x_{(2)}/\theta} \\ &= \frac{2}{\theta^2} e^{-(x_{(1)}+x_{(2)})/\theta} \quad 0 < x_{(1)} \leq x_{(2)}. \end{aligned}$$

The CDF technique is used to determine the CDF of Y . Let $Y = X_{(1)}/X_{(2)}$ and define the dummy transformation $Z = X_{(2)}$. The random variables Y and Z define a one-to-one transformation that maps $\mathcal{A} = \{(x_{(1)}, x_{(2)}) \mid 0 < x_{(1)} \leq x_{(2)}\}$ to $\mathcal{B} = \{(y, z) \mid 0 < y \leq 1, z > 0\}$. Since $y = x_{(1)}/x_{(2)}$ and $z = x_{(2)}$ (i.e., $x_{(1)} = yz$ and $x_{(2)} = z$) and the Jacobian of the inverse transformation is $J = z$, then the PDF of

Y is

$$\begin{aligned}
 f_Y(y) &= \int_0^\infty \frac{2}{\theta^2} e^{-(y+z)/\theta} |z| dz \\
 &= \frac{2}{\theta^2} \int_0^\infty e^{-z(y+1)/\theta} z dz \\
 &= \frac{2}{\theta^2} \left[z \left(-\frac{\theta}{y+1} e^{-z(y+1)/\theta} \right) \Big|_0^\infty - \int_0^\infty \left(-\frac{\theta}{y+1} e^{-z(y+1)/\theta} dz \right) \right] \\
 &= \frac{2}{\theta^2} \left[0 - 0 + \frac{\theta}{y+1} \int_0^\infty e^{-z(y+1)/\theta} dz \right] \\
 &= \frac{2}{\theta^2} \left[-\frac{\theta^2}{(y+1)^2} e^{-z(y+1)/\theta} \Big|_0^\infty \right] \\
 &= \frac{2}{\theta^2} \left[\frac{\theta^2}{(y+1)^2} \right] \\
 &= \frac{2}{(y+1)^2} \quad 0 < y \leq 1.
 \end{aligned}$$

The final step in determining the distribution of D_2 is to project the maximum of A , B , C , and D in Figure 10.3 onto the vertical axis. In order to determine the CDF for D_2 , we need to find the functions F_α , F_β , and F_γ associated with the following limits for the CDF of D_2 :

$$F_{D_2}(d) = \begin{cases} 0 & 0 < d \leq C(y^{**}) \\ F_\alpha(d) & C(y^{**}) < d \leq C(y^*) \\ F_\beta(d) & C(y^*) < d \leq \frac{1}{2} \\ F_\gamma(d) & \frac{1}{2} < d \leq 1 - \frac{1}{e} \\ 1 & d > 1 - \frac{1}{e}. \end{cases}$$

Determining formulas for F_α , F_β , and F_γ

In order to determine the three functions F_α , F_β , and F_γ associated with the CDF of D_2 , it will be necessary to find the point of intersection of a horizontal line at height d in Figure 10.4 with $C(y)$, $A(y)$, and the first piece of $B(y)$. These points

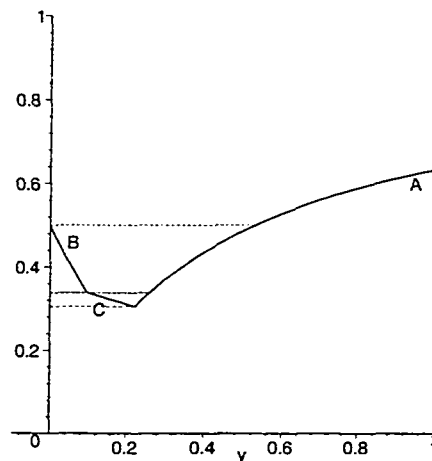


Figure 10.4: $D_2 = \max\{A, B, C, D\}$.

of intersection will be necessary in order to set up the appropriate integration limits when using the CDF technique to determine the distribution of D_2 .

First, consider the intersection of the first piece of $B(y)$ with a horizontal line at height d in Figure 10.4:

$$e^{-2y/(y+1)} - \frac{1}{2} = d.$$

Solving this equation for y yields

$$y = -\frac{\log(d + 1/2)}{2 + \log(d + 1/2)}.$$

Next, consider the intersection of $C(y)$ with a horizontal line at height d in Figure 10.4:

$$\frac{1}{2} - e^{-2/(y+1)} = d.$$

Solving this equation for y yields

$$y = -\frac{2 + \log(1/2 - d)}{\log(1/2 - d)}.$$

Finally, consider the intersection of $A(y)$ with a horizontal line at height d in Figure 10.4:

$$1 - e^{-2y/(y+1)} = d.$$

Solving this equation for y yields

$$y = -\frac{\log(1-d)}{2 + \log(1-d)}.$$

The following three paragraphs give the limits of integration associated with the functions F_α , F_β , and F_γ .

For $C(y^{**}) < d \leq C(y^*)$, $F_{D_2}(d)$, i.e., expression for $F_\alpha(d)$, is

$$\begin{aligned} F_\alpha(d) &= \Pr(D_2 \leq d) \\ &= \int_{-\frac{2+\log(1/2-d)}{\log(1/2-d)}}^{y^{**}} f_Y(y) dy + \int_{y^{**}}^{-\frac{\log(1-d)}{2+\log(1-d)}} f_Y(y) dy \\ &= -\log(1/2-d) - 2 - \log(1-d) \\ &= -2 - \log[(1/2-d)(1-d)]. \end{aligned}$$

For $C(y^*) < d \leq 0.5$, $F_{D_2}(d)$, i.e., expression for $F_\beta(d)$, is

$$\begin{aligned} F_\beta(d) &= \Pr(D_2 \leq d) \\ &= 1 - \Pr(D_2 > d) \\ &= 1 - \left[\int_0^{-\frac{\log(d+1/2)}{2+\log(d+1/2)}} f_Y(y) dy + \int_{-\frac{\log(1-d)}{2+\log(1-d)}}^1 f_Y(y) dy \right] \\ &= \log\left(\frac{d+1/2}{1-d}\right). \end{aligned}$$

For $0.5 < d \leq 1 - 1/e$, $F_{D_2}(d)$, i.e., expression for $F_\gamma(d)$, is

$$\begin{aligned}
 F_\gamma(d) &= \Pr(D_2 \leq d) \\
 &= 1 - \Pr(D_2 > d) \\
 &= 1 - \left[\int_{-\frac{\log(1-d)}{2+\log(1-d)}}^1 f_Y(y) dy \right] \\
 &= -\log(1-d).
 \end{aligned}$$

Putting the pieces together, the CDF for D_2 is

$$F_{D_2}(d) = \begin{cases} 0 & 0 < d \leq C(y^{**}) \\ -2 - \log[(1/2 - d)(1 - d)] & C(y^{**}) < d \leq C(y^*) \\ \log\left(\frac{d+1/2}{1-d}\right) & C(y^*) < d \leq \frac{1}{2} \\ -\log(1-d) & \frac{1}{2} < d \leq 1 - \frac{1}{e} \\ 1 & d > 1 - \frac{1}{e}. \end{cases}$$

This CDF is consistent with the tabled values from Leemis (1995, page 274) which were generated using Monte Carlo simulation with 500,000 replications.

Example 10.1. Let $x_{(1)} = 95$ and $x_{(2)} = 100$. The maximum likelihood estimator $\hat{\theta}$ is

$$\hat{\theta} = \frac{95 + 100}{2} = 97.5$$

The ratio of the data values

$$y = \frac{x_{(1)}}{x_{(2)}} = 0.95,$$

which indicates, from Figure 10.4, that the test statistic

$$D_2 = A(0.95) = 1 - e^{-2(0.95)/(1.95)} \cong 0.6226$$

falls in the right-hand tail of the distribution of D_2 and hence provides evidence to reject the null hypothesis for the goodness-of-fit test. Since large values of the test statistic lead to rejecting H_0 , the p -value associated with this particular data set is

$$p = 1 - F_{D_2}(0.6226) = 1 + \log(1 - 0.6226) = 0.02556. \quad \square$$

The procedure `ExponentialKSRV(Data)` returns the PDF of D_n for the exponential distribution when given a list of data values `Data`. For data sets containing more than two elements, the procedure currently prints an error message.

10.2 Others

This section contains various other applications of APPL procedures.

Example 10.2. This example considers the use of the Kolmogorov–Smirnov test for assessing model adequacy (goodness-of-fit) for the prime modulus multiplicative linear congruential random number generator:

$$z_{i+1} = az_i \pmod{m}$$

for $i = 0, 1, \dots$, where z_0 is a seed, $a = 7^5 = 16,807$, and $m = 2^{31} - 1 = 2,147,483,647$ (Park and Miller, 1988). The random numbers generated are z_1/m , z_2/m , etc. If the seed $z_0 = 987,654,321$ is used, then the first five random numbers generated are

$$\begin{array}{r} \underline{1,605,065,384} \\ 2,147,483,647 \end{array} \quad \begin{array}{r} \underline{1,791,818,921} \\ 2,147,483,647 \end{array} \quad \begin{array}{r} \underline{937,423,366} \\ 2,147,483,647 \end{array}$$

$$\begin{array}{r} \underline{1,334,477,970} \\ 2,147,483,647 \end{array} \quad \begin{array}{r} \underline{252,032,522} \\ 2,147,483,647 \end{array}$$

or, approximately

0.7474168 0.8343807 0.4365218
 0.6214147 0.1173618.

Since these five data values are being evaluated for their uniformity, there should be a reasonable match between their empirical cumulative distribution function and the cumulative distribution function for a $U(0, 1)$ random variable. If we let the list `Sample` contain the five random numbers generated above, then the APPL statements required to plot these two functions over the interval $(0, 1)$, shown in Figure 10.5, are

```
> n := 5;
> a := 7 ^ 5;
> seed := 987654321;
> m := 2 ^ 31 - 1;
> Sample := [];
> for j from 1 to n do
>   seed := a * seed mod m;
>   Sample := [op(Sample), seed / m];
> od;
> U := UniformRV(0, 1)
> PlotEmpVsFittedCDF(U, Sample, [], 0, 1);
```

The five parameters to the plotting function are the random variable whose CDF is to be plotted, the data values in a list, the parameters associated with the random variable (empty in this case of $U(0, 1)$), and the optional lower and upper plotting limits.

Let $F(x)$ be the hypothesized CDF and $F_5(x)$ be the empirical CDF. In order to determine the Kolmogorov–Smirnov test statistic,

$$D_5 = \sup_x |F(x) - F_5(x)|,$$

which measures the largest vertical distance between the two cumulative distribution

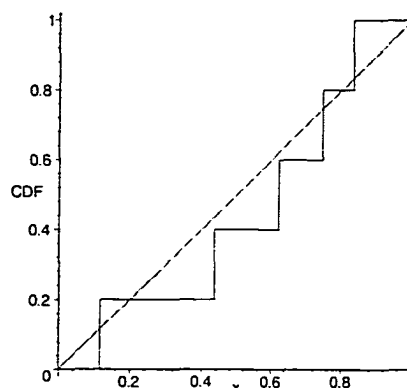


Figure 10.5: The empirical CDF of Sample and the theoretical $U(0, 1)$ CDF.

functions, the following additional command must be issued

```
> TestStat := KSTest(U, Sample, []);
```

The approximate value of the test statistic for the five random numbers is 0.2365, which occurs just to the left of the random number 0.4365.

Since large values of the test statistic indicate a poor fit and the cumulative distribution function $F_{D_5}(y)$ of the test statistic is (Drew et al., 2000)

$$F_{D_5}(y) = \begin{cases} 0 & y < \frac{1}{10} \\ \frac{24}{625} (10x - 1)^5 & \frac{1}{10} \leq y < \frac{1}{5} \\ -288x^4 + 240x^3 - \frac{1464}{25}x^2 + \frac{672}{125}x - \frac{96}{625} & \frac{1}{5} \leq y < \frac{3}{10} \\ 160x^5 - 240x^4 + \frac{424}{5}x^3 + 12x^2 - \frac{168}{25}x + \frac{336}{625} & \frac{3}{10} \leq y < \frac{2}{5} \\ -20x^5 + 74x^4 - \frac{456}{5}x^3 + \frac{224}{5}x^2 - \frac{728}{125}x & \frac{2}{5} \leq y < \frac{1}{2} \\ 12x^5 - 6x^4 - \frac{56}{5}x^3 + \frac{24}{5}x^2 + \frac{522}{125}x - 1 & \frac{1}{2} \leq y < \frac{3}{5} \\ -20y^6 + 32y^5 - \frac{185}{9}y^3 + \frac{175}{36}y^2 + \frac{3371}{648}y - 1 & \frac{1}{2} \leq y < \frac{3}{5} \\ -8x^5 + 22x^4 - \frac{92}{5}x^3 + \frac{12}{25}x^2 + \frac{738}{125}x - 1 & \frac{3}{5} \leq y < \frac{4}{5} \\ 2x^5 - 10x^4 + 20x^3 - 20x^2 + 10x - 1 & \frac{4}{5} \leq y < 1 \\ 1 & y \geq 1, \end{cases}$$

the p -value for this particular test is found with the additional APPL statement

```
> p := SF(KSRV(5), TestStat);
```

which yields $p \cong 0.8838$.

If this process is repeated for a total of 1000 groups of nonoverlapping consecutive sets of five random numbers, the empirical CDF of the Kolmogorov–Smirnov statistics should be close to the theoretical from APPL if the random number generator is valid. Figure 10.6 is a plot of the empirical CDF of the 1000 Kolmogorov–Smirnov statistics versus the theoretical Kolmogorov–Smirnov CDF with $n = 5$. The empirical CDF lies slightly above the theoretical. If this experiment were performed repeatedly, the empirical CDFs should fluctuate around the theoretical CDF. \square

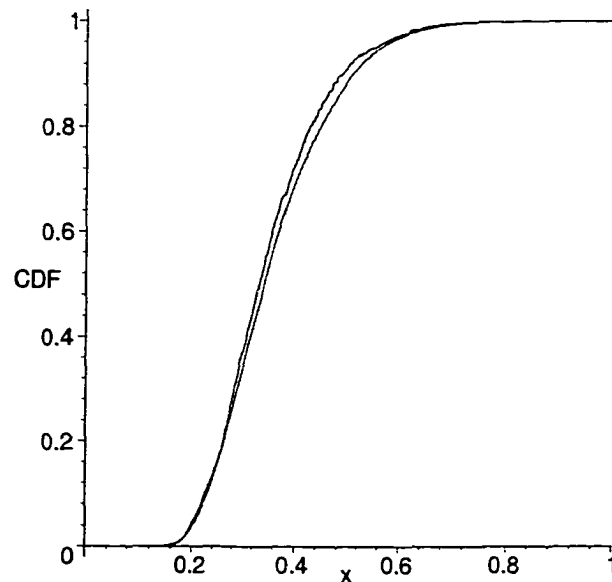


Figure 10.6: Empirical CDF of 1000 Kolmogorov–Smirnov statistics and the theoretical Kolmogorov–Smirnov CDF for $n = 5$.

Example 10.3. Consider a 2×2 matrix of $U(0, 1)$ random variables. Find the distribution of the determinant of the matrix.

Solution: Let $X_1, X_2, X_3,$ and X_4 be iid $U(0, 1)$, and let $Y_1 = X_1X_2$ and $Y_2 = X_3X_4$. The distribution of Y_i for $i = 1, 2$ is $f_{Y_i}(y) = -\log(y)$ for $0 < y < 1$ (which can be determined in APPL). Now we must find the distribution of $Y_1 - Y_2$.

The joint PDF of Y_1 and Y_2 is

$$f_{Y_1, Y_2}(y_1, y_2) = (\log(y_1))(\log(y_2))$$

for $0 < y_1 < 1, 0 < y_2 < 1$. Consider the one-to-one transformation ϕ such that

$$\phi: \begin{array}{l} w_1 = y_1 - y_2 \\ w_2 = y_1 + y_2 \end{array} \quad \phi^{-1}: \begin{array}{l} y_1 = \frac{w_1 + w_2}{2} \\ y_2 = \frac{w_2 - w_1}{2} \end{array}.$$

The Jacobian of the inverse transformation is

$$J = \begin{vmatrix} \frac{1}{2} & \frac{1}{2} \\ -\frac{1}{2} & \frac{1}{2} \end{vmatrix} = \frac{1}{2}.$$

The joint PDF of Y_1 and Y_2 is

$$\begin{aligned} f_{W_1, W_2}(w_1, w_2) &= f_{Y_1, Y_2}(\phi^{-1}(w_1, w_2)) \cdot |J| \\ &= \frac{1}{2} \log\left(\frac{w_1 + w_2}{2}\right) \log\left(\frac{w_2 - w_1}{2}\right), \end{aligned}$$

where w_1, w_2 satisfy the inequalities $-w_1 < w_2, w_2 < 2 - w_1, w_2 < w_1,$ and $w_1 - 2 < w_2$. The marginal distribution of W_1 is

$$f_{W_1}(w_1) = \begin{cases} \int_{-w_1}^{2+w_1} \frac{1}{2} \log\left(\frac{w_1 + w_2}{2}\right) \log\left(\frac{w_2 - w_1}{2}\right) dw_2, & -1 < w_1 < 0 \\ \int_{w_1}^{2-w_1} \frac{1}{2} \log\left(\frac{w_1 + w_2}{2}\right) \log\left(\frac{w_2 - w_1}{2}\right) dw_2, & 0 < w_1 < 1. \end{cases}$$

Simplifying the above expression yields

$$f_{W_1}(w_1) = \begin{cases} 2 - w_1 \log(w_1 + 1) - \log(w_1 + 1) + 2w_1 + w_1 \operatorname{dilog}\left(\frac{w_1+1}{w_1}\right) - \\ \quad w_1 \log(-w_1) \log(w_1^{-1}) - (1/6)w_1 \pi^2 - w_1 \log(-w_1) & -1 < w_1 < 0 \\ 2 + w_1 \log(1 - w_1) - \log(1 - w_1) + w_1 \log(w_1) - w_1 \operatorname{dilog}(w_1) - \\ \quad (1/2)w_1 (\log(w_1))^2 - 2w_1 + w_1 \log(1 - w_1) \log(w_1^{-1}) & 0 < w_1 < 1, \end{cases}$$

where $\operatorname{dilog}(x) = \int_1^x \frac{\log(t)}{1-t} dt$.

Using the APPL `Determinant` procedure, which returns the PDF of the determinant of a 2×2 matrix with random variables as elements, we determine the PDF of this example with the statements:

```
> X11 := UniformRV(0, 1);
> X12 := UniformRV(0, 1);
> X21 := UniformRV(0, 1);
> X22 := UniformRV(0, 1);
> M := array(1 .. 2, 1 .. 2, [[X11, X12], [X21, X22]]);
> Determinant(M);
```

□

Example 10.4. Find the distribution of the distance between two points chosen randomly in the unit square.

Solution: Let (X_1, Y_1) and (X_2, Y_2) be two pairs of independent and identically distributed $U(0, 1)$ random variables. The distribution of the distance between the two points can be found by hand using a similar process to the one exhibited in the last example.

The APPL statements used to determine this distribution are

```
> X1 := UniformRV(0, 1);
> X2 := UniformRV(0, 1);
> Y1 := UniformRV(0, 1);
> Y2 := UniformRV(0, 1);
> DX := Difference(X1, X2);
```

```

> DY := Difference(Y1, Y2);
> g := [[x -> x ^ 2, x -> x ^ 2], [-infinity, 0, infinity]];
> h := [[x -> sqrt(x)], [0, infinity]];
> SDX := Transform(DX, g);
> SDY := Transform(DY, g);
> SSQ := Convolution(SDX, SDY);
> Z := Transform(SSQ, h);

```

The resulting PDF is

$$f_Z(z) = 2 (z^2 + \pi - 4 |z|) z \quad 0 \leq z < 1$$

and

$$f_Z(z) = -2 \left(2 \sqrt{z^2 - 1} + 2 \arcsin \left(\frac{z^2 - 2}{z^2} \right) \sqrt{z^2 - 1} + z^2 \sqrt{z^2 - 1} - 4z^2 + 4 \right) \frac{z}{\sqrt{z^2 - 1}}$$

for $1 \leq z \leq \sqrt{2}$. The APPL statement `DX := Difference(X1, X2)` assigns the PDF of $X_1 - X_2$ to the Maple variable `DX`. Similarly, $Y_1 - Y_2$ gets assigned to the variable `DY`. The `Transform` function transforms the random variable `DX` by the function $g(x) = x^2$. Thus, the statement `SDX := Transform(DX, g)` assigns the PDF of $(X_1 - X_2)^2$ to the variable `SDX`, while `SDY` is assigned the PDF of $(Y_1 - Y_2)^2$. The `Convolution` procedure calculates the PDF of the sum of the random variables `SDX` and `SDY`. Last, the resulting convolution, `SSQ` is transformed by the function $h(x) = \sqrt{x}, x > 0$ to produce the PDF $f_Z(z)$. \square

Example 10.5. Let $X_1, X_2,$ and X_3 be a random sample drawn from an exponential distribution with failure rate λ and PDF $f_X(x) = \lambda e^{-\lambda x}$ for $x > 0$. Test

$$H_0 : \lambda = 5$$

$$H_1 : \lambda < 5$$

at significance level $\alpha = 0.01$ using the test statistics

(a) $X_1 + X_2 + X_3$, and

(b) $X_{(3)}$.

Find the critical values and the power curves for each test statistic.

Solution (a): Let $Y = X_1 + X_2 + X_3$ and c_1 be the critical value of the test. Since large values of the test statistic lead to rejecting H_0 , we need to compute the critical value c_1 such that $\Pr(Y > c_1) = 0.01$ under H_0 . We can first find the survivor function (SF) of the sum Y with the APPL statements

```
> X := ExponentialRV(5);
> Y := ConvolutionIID(X, 3);
> SY := SF(Y);
```

which returns

$$S_Y(y) = \frac{25}{2} y^2 e^{-5y} + 5 y e^{-5y} + e^{-5y} \quad y > 0.$$

In order to determine $\Pr(Y > c_1) = 0.01$, we first use the Maple procedures `op` and `unapply` to extract the survivor function from the list of sublists Y and set it equal to 0.01. We then use Maple's numeric solver, `fsolve`, to solve the resulting equation

$$\frac{25}{2} c_1^2 e^{-5c_1} + 5 c_1 e^{-5c_1} + e^{-5c_1} = 0.01$$

for c_1 . The Maple statement needed to solve this equation for c_1 is

```
> c1 := fsolve(op(unapply(SY[1](c1))(c1)) = 0.01);
```

which yields $c_1 \cong 1.681189383$. We can verify this value of c_1 with the APPL statement

```
> alpha := SF(Y, c1);
```

which returns $\alpha = 0.01$.

We will determine the power curve for part (a) after determining the critical value for the test statistic $X_{(3)}$ in part (b). We want to examine each test statistic's power curve on the same plot.

Solution (b): Let c_2 be the critical value of the test. Since large values of the test statistic again lead to rejecting H_0 , we need to compute the critical value c_2 such that $\Pr(Y > c_2) = 0.01$ under H_0 . We first find the survivor function of the third order statistic, X_3 , with the APPL statements

```
> X := ExponentialRV(5);
> X3 := OrderStat(X, 3, 3);
> SX3 := SF(X3);
```

where $SF(X_3)$ computes the survivor function of the third order statistic when three items are drawn randomly from the given exponential population. The survivor function of the third order statistic is

$$S_{X_{(3)}}(x) = e^{-15x} - 3e^{-10x} + 3e^{-5x} \quad x > 0.$$

The $\Pr(X_{(3)} > c_2) = 0.01$ is solved with the APPL statement

```
> c2 := fsolve(op(unapply(SX3[1](c2))(c2)) = 0.01);
```

which yields $c_2 \cong 1.140087221$. We can verify this value of c_2 with the APPL statement

```
> alpha := SF(X3, c2);
```

which returns $\alpha = 0.009999999988$.

The additional statements needed to generate the power curves for the test statistics $Y = X_1 + X_2 + X_3$ and $X_{(3)}$ are

```

> X := ExponentialRV(lambda);
> Y := SF(ConvolutionIID(X, 3));
> X3 := SF(OrderStat(X, 3, 3));
> PCa := subs(y = c1, Y[1](y));
> PCb := subs(x = c2, X3[1](x));
> PCPlota := plot(subs(lambda = i, op(PCa)), i = 0 .. 6):
> PCPlotb := plot(subs(lambda = i, op(PCb)), i = 0 .. 6):
> plots[display]([PCPlota, PCPlotb]);

```

As can be seen in Figure 10.7, the sum test statistic, $Y = X_1 + X_2 + X_3$, is more *powerful* than the order statistic test statistic, $X_{(3)}$, for this particular hypothesis test.

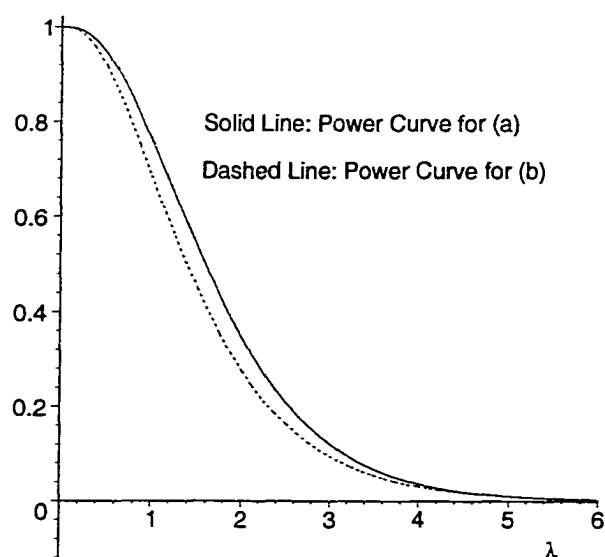


Figure 10.7: Power curves for the test statistic $Y = X_1 + X_2 + X_3$ (solid line) and test statistic $X_{(3)}$ (dashed line) for Example 10.5.

Example 10.6. (Barr & Sherrill, 1999, pages 357–358) (Army selection boards)

The Army uses centralized Army-wide selection boards to select officers for promotion and advanced military schooling. Each selection board determines a performance-based “order of merit” ranking of the officers under its consideration. Generally, officers under consideration for promotion

or advanced schooling who are not selected for the new grades or schools either leave or are separated from the Army. Officers being considered for promotion to Lieutenant Colonel (LTC), for example, have successfully passed five such selection boards. Data from recent years shows the selection rates of these boards averages about 78%. Thus, very roughly, the fraction of officers remaining after five boards is about 30% of the original population, i.e., $0.78^5 = 0.29$. If we assume the original population of officers has normally distributed “performance,” and selection boards select officers with the highest performance, then a LTC selection board is effectively considering a truncated normal population of “performance,” with a truncation point corresponding to the 70th percentile of the original normal population. Determine the variance of the population under consideration by the LTC board. (Barr & Sherrill, 1999, pages 357–358)

Solution: For a standard normal distribution, truncation at the 70th percentile would correspond to a lower truncation point of $b = 0.53$. To determine the variance of a truncated standard normal distribution with a lower truncation point $b = 0.53$, the following APPL statements are used

```
> X := StandardNormalRV();
> b := IDF(X, 7 / 10);
> T := Truncate(X, b, infinity);
> var := Variance(T);
```

The statements return the variance $\text{var} \cong 0.2636$, meaning the variance of the population under consideration by the LTC board is only about one-fourth that of the original population. Barr & Sherrill state:

“... that this relatively small variance makes it more difficult for the board to discriminate among officers under consideration. Members of

Table 10.1: Variance of a truncated standard normal distribution T for increasing values of the lower truncation point t .

Lower truncation value t	Variance of T truncated below at t
0.53	0.2636
0.65	0.2452
0.75	0.2309
0.85	0.2176
0.95	0.2050

selection boards for the higher ranks are sometimes quoted as saying, ‘All the officers look about the same.’”

Barr & Sherrill also note that as t increases, there is a rapid decrease in variation. Table 10.1 displays the variance of the truncated standard normal distribution for increasing values of t as determined in APPL. They conclude that, at higher ranks, there is relatively little difference in performance scores. \square

Example 10.7. (Maple animation of a continuous order statistic) The PDF of a standardized inverse Gaussian (IG) random variable X is

$$f_X(x) = \frac{1}{\sqrt{2\pi}} \left(\frac{3}{3+kx} \right)^{3/2} e^{-3x^2/(6+2kx)} \quad -\frac{3}{k} < x < \infty.$$

Balakrishnan and Chen’s (1997) text, *CRC Handbook of Tables for Order Statistics from Inverse Gaussian Distributions with Applications*, contains hundreds of pages of tables and plots for IG distributions. A plot of the $N(0, 1)$ and the standardized $IG(0.8)$ PDFs overlaid in a single plot, for example, is on page 50. Not only can APPL reproduce many of these tables and plots (with ease), it can also produce animations (with Maple’s `animate` procedure) to make comparisons of plots for various parameter

values. For example, The two plots for the $N(0, 1)$ and $IG(k)$ may be overlaid and animated for k increasing from zero to one as follows:

```
> Z := NormalRV(0, 1);
> X := [[x -> (3 / (3 + k * x)) ^ (3 / 2) *
        exp(-3 * x ^ 2 / (6 + 2 * k * x)) / sqrt(2 * Pi)],
        [-3 / k, infinity], ["Continuous", "PDF"]];
> NormalExpr := op(unapply(Z[1](x))(x));
> InvGaussExpr := op(unapply(X[1](x))(x));
> unassign('k');
> plots[animate]({NormalExpr, InvGaussExpr}, x = -4 .. 4, k = 0 .. 1);
```

The plot is shown in Figure 10.8 for $k = 0.8$. To execute the animation, first select the plot by clicking on it. Then choose “Play” from the “Animation” menu.

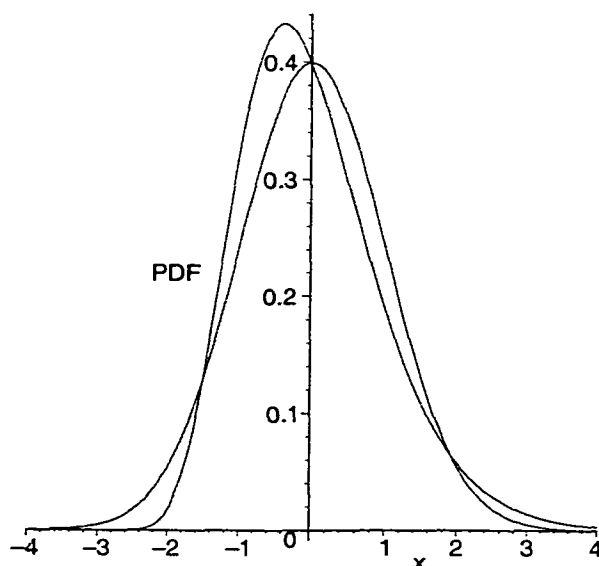


Figure 10.8: Overlaid plots of the standard normal and standard $IG(0.8)$ distributions.

Chapter 11

Future Work

As new algorithms are devised and implemented as APPL procedures, it is clear that the discrete data structure will need to become more general. There are instances when manipulating one or more random variables results in a random variable with a mixed *Dot* and *NoDot* format. We do not want these random variables to be alien to the existing APPL procedures, and the best way to remedy this situation is to adjust the data structure format first. Since we eventually want APPL procedures to be able to work with random variables that have a mixed discrete and continuous support, developing a mixed *Dot/NoDot* format for random variables with only discrete supports is a good place to begin construction on a new or revised format.

Not only would we like to see the data structure become more general in the future, but we would like to begin eliminating certain restrictions on random variables in various procedures. In the **Maximum** procedure, for example, we assume that discrete random variables have infinite supports consisting of consecutive integer values. This would exclude, for example, determining the PDF of the maximum of the random variable $Y = 2X$, where $X \sim \text{Geometric}(p)$. In other procedures, the restrictions are not implemented as part of the procedure, but are intrinsic Maple restrictions, e.g., sizes of lists. Some Maple restrictions are possible to work around, such as

using Maple's numeric equation solver `fsolve` (in MOM and MLE) when `solve` cannot determine an analytic solution. In other procedures, such as CDF or Transform, it is unclear how to proceed when an analytic inverse cannot be determined.

One of our primary interests is finding additional application areas for APPL, especially those involving discrete random variables. Our article "APPL: A Probability Programming Language" introduced the statistical community to APPL in the May, 2001 issue of *The American Statistician*. This article was devoted primarily to continuous distributions. We are encouraged by the interest level in APPL, and we are continually seeking applications from other fields of interest. We are interested in situations where an "exact" probability calculation is needed. Not only is APPL a tool for extending the depth of probabilistic theory, but it has the potential to strengthen the analysis and design of problems from other scientific fields.

Some of the application areas we are considering for future work are

- **Reliability:** Finding the exact distribution of a system time to failure given the component time to failure distributions.
- **Networks:** Finding the exact distribution of the project duration in a stochastic activity network.
- **Mechanical Design:** Finding the exact distribution of clearance in a design with random tolerancing parameters.
- **Statistics:**
 - Analysis of outliers,
 - Critical values for hypothesis testing,
 - Distribution of point estimators,
 - Coverage of confidence intervals.

Besides research, APPL is a tool for learning probability. Theoretical aspects of probability are enhanced by the visualization of the manipulation of random variables in a computer algebra system. It is my intention to continue developing APPL for use by students and researchers at all levels in their scientific careers. What we give to others today may be that small piece of that something they need to to make their breakthrough tomorrow.

Appendix A

Algorithm for OrderStat

The algorithm for the APPL `OrderStat(X, n, r, ["wo"])` procedure returns the PDF of the r th order statistic of a random variable X (with support Ω) given that n items are sampled from the random variable's population, either with or without ("wo") replacement. `OrderStat` uses the PDF, CDF, SF, `NextPermutation`, and `NextCombination` procedures. Algorithms for `NextPermutation` and `NextCombination` are in Appendix B.

Procedure `OrderStat(X, n, r, ["wo"])`

If $r > n$ then

 Return("Error: Order statistic index larger than sample size")

$fX \leftarrow \text{PDF}(X)$

$FX \leftarrow \text{CDF}(X)$

If (X is Continuous) then

$$fXOS \leftarrow \frac{n! (FX)^{r-1} (1 - FX)^{n-r} fX}{(r-1)! (n-r)!}$$

Else if (X is Discrete) then

$N \leftarrow |\Omega|$

$Lo \leftarrow \min(\Omega)$

$SX \leftarrow \text{SF}(fX)$

 If (Number of arguments = 3) then

[Sampling with replacement]

 If (X has a numeric PDF) then

 For $k \leftarrow 1$ to N

If ($N = 1$) then

$$fXOS[1] \leftarrow 1$$

Else if ($k = 1$) and ($k \neq N$) then

$$fXOS[1] \leftarrow \sum_{w=0}^{n-r} \binom{n}{w} [fX(1)]^{n-w} [SX(2)]^w$$

Else if ($k \neq 1$) and ($k = N$) then

$$fXOS[N] \leftarrow \sum_{u=0}^{r-1} \binom{n}{u} [FX(N-1)]^u [fX(N)]^{n-u}$$

Else

$$fXOS[k] \leftarrow \sum_{u=0}^{r-1} \sum_{w=0}^{n-r} \binom{n}{u, n-u-w, w} [FX(k-1)]^u [fX(k)]^{n-u-w} [SX(k+1)]^w$$

Else if (X has a symbolic PDF) then

$$fXOS1 \leftarrow \sum_{w=0}^{n-r} \binom{n}{w} [fX(L_0)]^{n-w} [SX(L_0+1)]^w$$

[$fXOS1$ holds the numeric PDF value of the r th order statistic at $x = L_0$]

$$fXOS2 \leftarrow \sum_{u=0}^{r-1} \sum_{w=0}^{n-r} \binom{n}{u, n-u-w, w} [FX(x-1)]^u [fX(x)]^{n-u-w} [SX(x+1)]^w$$

[$fXOS2$ holds the symbolic PDF of the r th order statistic at $x = L_0 + 1, L_0 + 2, \dots$]

If $fXOS2(L_0) = fXOS1$ then

$$fXOS \leftarrow fXOS2 \quad [\text{Return single function } fXOS]$$

Else

$$fXOS \leftarrow fXOS1, fXOS2 \quad [\text{Return piecewise defined } fXOS]$$

Else if (Number of arguments = 4) then

[Sampling without replacement]

If (X has finite support) then

If ($n > N$) then

Return("Error: Sample size larger than population size")

$fX \leftarrow \text{ConvertToNumeric}(fX)$ [Converts PDF of X to a numeric representation]

If (Equally likely distribution) then

For $i \leftarrow r$ to $N - n + r$

$$fXOS[i] \leftarrow \frac{\binom{i-1}{r-1} \binom{N-i}{n-r}}{\binom{N}{n}}$$

Else if (Non-equally likely distribution) then


```

If ( $n = 1$ ) then                                     [One item is sampled]
     $fXOS \leftarrow fX$ 
Else if ( $n = N$ ) then                                 [Entire population is sampled]
     $fXOS[r] \leftarrow 1$                              [The  $r$ th position is assigned the value 1, others 0]
Else                                                  [Number of items sampled is 2, 3, ...,  $N - 1$ ]
     $ProbStorage \leftarrow Array[1..n, 1..N]$ 
    For  $i \leftarrow 1$  to  $n$ 
        For  $j \leftarrow 1$  to  $N$ 
             $ProbStorage[i, j] \leftarrow 0$            [Initialize array to hold zeroes]
     $Combo \leftarrow [1..n]$                            [Create the first ordering of values 1 through  $n$ ]
    For  $i \leftarrow 1$  to  $\binom{N}{n}$ 
         $Perm \leftarrow Combo$  [Assign the permutation as the current combination]
        For  $j \leftarrow 1$  to  $n!$  [Compute the probability of obtaining the permutation]
             $PermProb \leftarrow fX[Perm[1]]$ 
             $CumSum \leftarrow PermProb$ 
            For  $m \leftarrow 2$  to  $n$ 
                 $PermProb \leftarrow PermProb \cdot \frac{fX[Perm[m]]}{1 - CumSum}$ 
                 $CumSum \leftarrow CumSum + fX[Perm[m]]$ 
             $OrderedPerm \leftarrow sort(Perm)$          [Sort the permutation]
            For  $m \leftarrow 1$  to  $n$ 
                For  $k \leftarrow 1$  to  $N$ 
                    If ( $OrderedPerm[k] = m$ ) then
                         $ProbStorage[m, k] \leftarrow PermProb + ProbStorage[m, k]$ 
                     $Perm \leftarrow NextPermutation(Perm)$  [Return next permutation]
                     $Combo \leftarrow NextCombination(Combo, N)$  [Return next combination]
    Else if (Infinite Support) then
        If ( $n = 1$ ) then                               [One item is sampled]
             $fXOS \leftarrow fX$ 
        Else if ( $n = 2$ ) and ( $r = 1$ ) then
             $fXOS \leftarrow fX(x) \left[ \frac{SX(x+1)}{1 - fX(x)} + \sum_{y=x+1}^{\infty} \frac{fX(y)}{1 - fX(y)} \right]$ 
        Else
            Return("No formula for this infinite support case")
Return( $fXOS$ )

```

Appendix B

Maple Code for NextCombination and NextPermutation

These Maple codes were adapted from Nijenhuis and Wilf (1978) and Reingold et al. (1977). The code for `NextCombination` generates the next lexicographical (“alphabetical order”) combination of size n (where n is the size of the previous combination) of the integers $1, 2, \dots, N$. The procedure receives as arguments a list of integers, which is the previous combination, and the size of the underlying set of integers from which the next combination is to be formed. For example, if `Previous := [1, 2, 4, 7]` and `N = 10` are entered as arguments in `NextCombination`, then `NextCombination` returns the next lexicographical combination of four elements chosen from the set $\{1, 2, \dots, 10\}$ as `[1, 2, 4, 8]`. As another example, if `Previous := [1, 4, 9, 10]` and `N = 10`, then `NextCombination` returns `[1, 5, 6, 7]`.

The code for `NextPermutation` generates the next lexicographical permutation of the integers $1, 2, \dots, N$. The code receives as its only argument a list of integers, which is the previous permutation. For example, if `Previous := [1, 2, 4, 7]` is entered as an argument in `NextPermutation`, then the code returns the next lexicographical permutation as `[1, 2, 7, 4]`. Similarly, `[1, 2, 7, 4]` as an argument generates

the next permutation [1, 4, 2, 7].

The method being used for both codes requires the construction of the next combination (or permutation) by a small modification of the previous combination (or permutation). We chose the lexicographical approach because of its computational simplicity and straightforward construction.

```

NextCombination := proc(Previous :: list, N :: posint)
local Next, n, MoveLeft, i, j:

if (nargs <> 2) then
  print('ERROR(NextCombination): This procedure requires 2 arguments'):
  RETURN():
fi:
Next := Previous:
n := nops(Previous):
#
# If the value in the final position of the combination is not the
# maximum value it can attain, N, then increment it by 1.
#
if (Next[n] <> N) then
  Next[n] := Next[n] + 1:
#
# If the final position in the combination is already at its maximum
# value, N, then move left through the combination and find the next
# possible value that can be incremented. Index position i's maximum
# attainable value is N + i - n.
#
else
  MoveLeft := true:
  for i from (n - 1) by -1 to 1 while (MoveLeft = true) do
    if (Next[i] < N + i - n) then
      Next[i] := Next[i] + 1:
#
# Upon incrementing the rightmost element in position i, reset each
# value in the jth position (j = 1, 2, ... , n - i) to the right of
# the ith position to 1 more than the value in the preceding position.
#
      for j from 1 to (n - i) do
        Next[i + j] := Next[(i + j) - 1] + 1:
      od:

```

```

    MoveLeft := false:
  fi:
od:
fi:
RETURN(Next):
end:

```

Example B.1. Let $A := [1, 2, 4, 6]$ and $N = 10$. Find `NextCombination(A)`.

1. Error check: Two arguments supplied.
2. Assign `Next := [1, 2, 4, 6]`. Since the final position in the combination is not the value $N = 10$, then increment it by 1.
3. Return the next combination as $[1, 2, 4, 7]$.

Example B.2. Let $A := [1, 4, 9, 10]$ and $N = 10$. Find `NextCombination(A)`.

1. Error check: Two arguments supplied.
2. Assign `Next := [1, 4, 9, 10]`. Since the final position in the combination has already attained its maximum value, $N = 10$, then scan the combination from right to left to locate the rightmost element that has not yet attained its maximum value. Since position two has not attained its maximum value of 8, then increment it by 1. `Next` becomes $[1, 5, 9, 10]$.
3. Reset each value to the right of second position to one more than the value in the preceding position. Thus, position three's value becomes 6 and position four's value becomes 7. `Next` is now $[1, 5, 6, 7]$.
4. Return the next combination as $[1, 5, 6, 7]$.

```

NextPermutation := proc(Previous :: list)
local Next, n, flag, i, OrigVal, SwapIndex, j, Temp1, k, Temp2, m:

if (nargs <> 1) then
  print('ERROR(NextPermutation): This procedure requires 1 argument'):
  RETURN():
fi:
Next := Previous:
n := nops(Previous):
flag := false:
#
# Find the largest index value i for which Next[i] < Next[i + 1].
#
for i from n - 1 to 1 by -1 while not(flag) do
  if (Next[i] < Next[i + 1]) then
    flag := true:
    OrigVal := Next[i]:
    SwapIndex := i + 1:
#
# Find the smallest value Next[j] for which Next[i] < Next[j] and i < j.
#
    for j from n to SwapIndex by -1 do
      if ((Next[j] < Next[SwapIndex]) and (Next[j] > OrigVal)) then
        SwapIndex := j:
        fi:
      od:
      Temp1 := Next[SwapIndex]:
      Next[SwapIndex] := Next[i]:
      Next[i] := Temp1:
#
# Reverse the order of the values to the right of the leftmost swapped value
#
      for k from i + 1 to n do
        Temp2[k] := Next[k]:
      od:
      for m from i + 1 to n do
        Next[m] := Temp2[n + i + 1 - m]:
      od:
    fi:
  od:
RETURN(Next):
end:

```

Example B.3. Let $A := [1, 4, 3, 2]$. Find $\text{NextPermutation}(A)$.

1. Error check: One argument supplied.
2. Assign `Next := [1, 4, 3, 2]`. The largest index value i in which `Next[i] < Next[i + 1]` is $i = 1$.
3. The smallest value `Next[j]` such that `Next[i] < Next[j]` and $i < j$ is `Next[4] = 2`.
4. Swap the values in position $i = 1$ and $j = 4$. `Next` becomes `[2, 4, 3, 1]`.
5. Reverse the order of the values to the right of `Next[1] = 2`. `Next` becomes `[2, 1, 3, 4]`.
6. Return `[2, 1, 3, 4]` as the next permutation.

Appendix C

Determining Candidate Sums for the Heap

Theorem: Let $x_1 < x_2 < \dots < x_n$ and $y_1 < y_2 < \dots < y_m$ be finite real numbers. Let the $m \times n$ array A be arranged as shown in Figure C.1 and have entries $A_{i,j} = y_i + x_j$, $i = 1, 2, \dots, m$ and $j = 1, 2, \dots, n$. Let the set C contain all (i, j) pairs such that $x_i + y_j \leq c$, where $c > y_1 + x_1$ is a real number. Let P be the path from the northwest corner of A to the southeast corner of A that separates C from C' . The smallest element in C' must occur just to the northeast of a southward followed by an eastward change in direction of the path P .

Proof: For any cell in C , every cell to the southwest of C is also in C since $x_1 < x_2 < \dots < x_n$ and $y_1 < y_2 < \dots < y_m$. Thus C must be a finite union of rectangles in A , where each rectangle contains the $(1, 1)$ cell. The next cell to be included in C as c increases is the smallest element in C' . Since C will continue to be a finite union of rectangles when the next cell is added, the smallest element in C' must occur at intersections of the rectangles and the western and southern boundaries of A that occur on P .

Figure C.2 displays the array A for Example 4.3 with Chapter 4 with $c = 3$.

y_m	$y_m + x_1$	$y_m + x_2$	\dots	$y_m + x_n$
\vdots	\vdots	\vdots	\ddots	\vdots
y_2	$y_2 + x_1$	$y_2 + x_2$	\dots	$y_2 + x_n$
y_1	$y_1 + x_1$	$y_1 + x_2$	\dots	$y_1 + x_n$
	x_1	x_2		x_n

Figure C.1: Array A where $x_1 < x_2 < \dots < x_n$ and $y_1 < y_2 < \dots < y_m$.

8	5				
5	2	4			
1	-2	0	3		
-2	-5	-3	0	4	
	-3	-1	2	6	8

Figure C.2: Array A corresponds to Example 4.3 in Chapter 4. The path P from the northwest corner to the southeast corner of A that delimits the set $C = \{(i, j) \mid y_i + x_j \leq 3\}$ from C' is thickened. The circled entries lie just to the northeast of points in the path where there is a turn from a southward to an eastward direction. These three entries are contained in the cells in C' that hold the smallest entries in C' for $c = 3$.

Appendix D

Algorithm for BruteForceMethod

The algorithm for the APPL procedure `BruteForceMethod(X, Y)` computes the PDF of the convolution of the PDFs of the two random variables X and Y by the “brute force method” described in Chapter 4. The support list for the convolution is sorted by a heapsort in the APPL procedure `HeapSort`, which sorts the elements of its first argument, making corresponding swaps to the elements of its second argument. The variables Ω_X and Ω_Y are the supports of the random variables X and Y , respectively.

```
Procedure BruteForceMethod(X, Y)
   $n \leftarrow |\Omega_X|$ 
   $m \leftarrow |\Omega_Y|$ 
   $s \leftarrow \text{array}[1 .. n \cdot m]$ 
   $Probs \leftarrow \text{array}[1 .. n \cdot m]$ 
  For  $i \leftarrow 1$  to  $n$ 
    For  $j \leftarrow 1$  to  $m$ 
       $s \leftarrow y_i + x_j$ 
       $Probs \leftarrow f_Y(y_i) \cdot f_X(x_j)$ 
  return(HeapSort( $s, Probs$ ))
```

Appendix E

Algorithm for MovingHeapMethod

The algorithm for the APPL procedure `MovingHeapMethod(X, Y)` computes the PDF of the convolution of two random variables X and Y by the “moving heap method” described in Chapter 4. The additional APPL procedures `RebuildHeap`, `InsertHeap`, and `PercolateDownHeap` are standard heap programs for inserting and restructuring a heap so that it continues to fulfill the properties of a heap.

```
Procedure MovingHeapMethod(X, Y)
   $n \leftarrow |\Omega_X|$ 
   $m \leftarrow |\Omega_Y|$ 
  Dimension  $s[n \cdot m]$ 
  Dimension  $Probs[n \cdot m]$ 
   $s_1 \leftarrow y_1 + x_1$ 
   $Probs_1 \leftarrow f_Y(y_1) \cdot f_X(x_1)$ 
  Dimension  $r[m + 1]$ 
  Dimension  $c[n + 1]$ 
   $row1col2entry \leftarrow [y_1 + x_2, f_Y(y_1) \cdot f_X(x_2)]$ 
   $r_1 \leftarrow 1$ 
   $c_2 \leftarrow 1$ 
   $row2col1entry \leftarrow [y_2 + x_1, f_Y(y_2) \cdot f_X(x_1)]$ 
   $r_2 \leftarrow 1$ 
```

```

 $c_1 \leftarrow 1$ 
 $r_{n+1} \leftarrow 1$  [Keeps search for new entries inside north border of  $A$ ]
 $c_{m+1} \leftarrow 1$  [Keeps search for new entries inside east border of  $A$ ]
 $H \leftarrow [-1 \cdot 10^{20}, \text{row1col2entry}, \text{row2col1entry}]$ 
 $Mimic \leftarrow [[0, 0], [1, 2], [2, 1]]$  [Holds the positions of the entries]
PercolateDownHeap(2, 3) [Restructures  $H$  to fulfill the heap properties]
For  $q \leftarrow 2$  to  $n \cdot m$ 
   $RootItem \leftarrow H_2$ 
   $RootPosition \leftarrow Mimic_2$ 
   $s_q \leftarrow RootItem_1$  [Root entry placed in sums array  $s$ ]
   $Probs_q \leftarrow RootItem_2$  [Root's probability placed in probability array  $Probs$ ]
   $a \leftarrow RootPosition_1$ 
   $b \leftarrow RootPosition_2$ 
   $r_a \leftarrow 0$  [The root's row becomes inactive]
   $c_b \leftarrow 0$  [The root's column becomes inactive]
   $size \leftarrow |H|$ 
   $H_2 \leftarrow H_{size}$ 
   $Mimic_2 \leftarrow Mimic_{size}$ 
   $H \leftarrow [H_1 .. H_{size-1}]$ 
   $Mimic \leftarrow [Mimic_1 .. Mimic_{size-1}]$ 
  RebuildHeap(2,  $size - 1$ ) [Restores  $H$  as a heap]
  If ( $r_a = 0$ ) and ( $c_{b+1} = 0$ ) then [If the cell just east of
    the removed entry is inactive,
    insert its entry into the heap]
     $r_a \leftarrow 1$ 
     $c_{b+1} \leftarrow 1$ 
     $NewPosition \leftarrow [a, b + 1]$ 
    InsertHeap( $NewPosition$ )
  If ( $r_{a+1} = 0$ ) and ( $c_b = 0$ ) then [If the cell just south of
    the removed entry is inactive,
    insert its entry into the heap]
     $r_{a+1} \leftarrow 1$ 
     $c_b \leftarrow 1$ 
     $NewPosition \leftarrow [a + 1, b]$ 
    InsertHeap( $NewPosition$ )
return( $s, Probs$ )

```

Appendix F

APPL Code for Benford

This appendix contains APPL code for computing the probability mass functions of Y when T has the unit exponential distribution. Other distributions are handled analogously. The values of the loop indices `low` and `high` must be input in order to avoid summing an infinite number of terms. The parameter in the `ExponentialRV` procedure refers to the failure rate in the exponential distribution. The `SF` procedure gives the survivor function of the random variable given in the first argument evaluated at the second argument.

```
T := ExponentialRV(1);
pmf := [0, 0, 0, 0, 0, 0, 0, 0, 0];
for y from 1 to 9 do
  for i from low to high do
    pmf[y] := pmf[y] + SF(T, y * 10 ^ i) - SF(T, (y + 1) * 10 ^ i);
  od;
od;
```

Bibliography

- Allaart, P. C. (1997), "An Invariant-Sum Characterization of Benford's Law," *Journal of Applied Probability*, 34, 288–291.
- Bain, L., and Engelhardt, M. (1992), *Introduction to Probability and Mathematical Statistics* (2nd ed.), PWS-KENT Publishing Company, Boston, MA.
- Balakrishnan, N., and Chen, W. W. S. (1997), *CRC Handbook of Tables for Order Statistics from Inverse Gaussian Distributions with Applications*, CRC, New York, NY.
- Barr, D., and Sherrill, T. (1999), "Mean and Variance of Truncated Normal Distributions," *The American Statistician*, 53, 357–361.
- Benford, F. (1938), "The Law of Anomalous Numbers," *Proceedings of the American Philosophical Society*, 78, 551–572.
- Bortkiewicz, L. (1898), *Das Gesetz der Kleinen Zahlen*, Teubner.
- Carrano, F. M., Helman, P., and Veroff, R. (1998), *Data Abstraction and Problem Solving with C++: Walls and Mirrors* (2nd ed.), Addison-Wesley Longman, Inc., Reading, MA.
- Casella, G., and Berger, R. (1990), *Statistical Inference*, Wadsworth & Brooks/Cole Advanced Books and Software, Pacific Grove, CA.
- David, H. A. (1970), *Order Statistics*, John Wiley & Sons, Inc., New York, NY.
- Drew, J. H., Glen, A. G., and Leemis, L. M. (2000), "Computing the Cumulative Distribution Function of the Kolmogorov-Smirnov Statistic," *Computational Statistics and Data Analysis*, 34, 1–15.
- Efron, B., and Tibshirani, R. J. (1993), *An Introduction to the Bootstrap*, Chapman & Hall, New York, NY.

- Feller, W. (1971), *An Introduction to Probability and Its Applications, Volume II*, John Wiley & Sons, Inc., New York, NY.
- Gehan, E. A. (1965), "A Generalized Wilcoxon Test for Comparing Arbitrarily Singly-Censored Samples," *Biometrika*, 52, 203–223.
- Glen, A. G., Leemis, L. M., and Drew, J. H. (1997), "A Generalized Univariate Change-of-Variable Transformation Technique," *INFORMS Journal on Computing*, 9, 288–295.
- Glen, A., Leemis, L., and Drew, J. (2001), "Computing the Distribution of the Product of Two Continuous Random Variables," Technical report, Mathematics Department, The College of William & Mary.
- Glen, A., Evans, D., and Leemis, L. (2001), "APPL: A Probability Programming Language," *The American Statistician*, 55, 156–166.
- Grinstead, C. M., and Snell, J. L. (1997), *Introduction to Probability* (2nd rev. ed.), American Mathematical Society, Providence, RI.
- Halmos, P. (1950), *Measure Theory*, D. Van Nostrand Company, Inc., New York, NY.
- Hastings, K. (2001), *Introduction to Probability with Mathematica*, Chapman & Hall/CRC, Boca Rotan, FL.
- Heal, K. M., Hansen, M. L., and Rickard, K. M. (1998), *Maple V Learning Guide*, Springer-Verlag, New York, NY.
- Hill, T. P. (1995), "A Statistical Derivation of the Significant-Digit Law," *Statistical Science*, 86, 354–363.
- Hill, T. P. (1998), "The First Digit Phenomenon," *American Scientist*, 86, 358–363.
- Hogg, R. V., and Craig, A. T. (1995), *Mathematical Statistics* (5th ed.), Prentice-Hall, Englewood Cliffs, NJ.
- Hogg, R. V., and Tanis, E. A. (1993), *Probability and Statistical Inference* (4th ed.), Macmillan Publishing Company, New York, NY.
- Karian, Z. A., and Tanis, E. A. (1999), *Probability and Statistics: Explorations with Maple* (2nd ed.), Wiley, New York, NY.
- Kendall, W. S. (1993), "Computer Algebra in Probability and Statistics," *Statistica*

- Neerlandica*, 47, 9–25.
- Larsen, R. J., and Marx, M. L. (1986), *An Introduction to Mathematical Statistics and Its Applications* (2nd ed.), Prentice–Hall, Englewood Cliffs, NJ.
- Larsen, R. J., and Marx, M. J. (2001), *An Introduction to Mathematical Statistics and Its Applications* (3rd ed.), Prentice–Hall, Upper Saddle River, NJ.
- Law, A. M., and Kelton, W. D. (2000), *Simulation Modeling and Analysis* (3rd ed.), McGraw–Hill, New York, NY.
- Lawless, J. F. (1982), *Statistical Models and Methods for Lifetime Data*, John Wiley & Sons, Inc., New York
- Leemis, L. M. (1995), *Reliability: Probabilistic Models and Statistical Methods*, Prentice–Hall, Englewood Cliffs, NJ.
- Ley, E. (1996), “On the Peculiar Distribution of the U.S. Stock Indices Digits,” *American Statistician*, 50, 311–313.
- Lopez, R. (2001), *Advanced Engineering Mathematics*, Addison–Wesley, Boston, MA.
- Margolin, B. H., and Winokur, Jr., H. S. (1967), “Exact Moments of the Order Statistics of the Geometric Distribution and their Relation to Inverse Sampling and Reliability of Redundant Systems,” *Journal of the American Statistical Association*, 62, 915–925.
- McCarron, J. (2001), MUG newsgroup communication.
- Miller, I., and Miller, M. (1999) *John E. Freund's Mathematical Statistics*, (6th ed.), Prentice–Hall, Upper Saddle River, NJ.
- Nicol, D. (2000), Personal communication.
- Nigrini, M. (1996), “A Taxpayer Compliance Application of Benford’s Law,” *Journal of the American Taxation Association*, 18, 72–91.
- Nijenhuis, A., and Wilf, H. S. (1978), *Combinatorial Algorithms: For Computers and Calculators* (2nd ed.), Academic Press, Inc., New York, NY.
- Park, S. K., and Miller, K. W. (1988), “Random Number Generators: Good Ones Are Hard to Find,” *Communications of the ACM*, 31, 1192–1201.

- Parlar, M. (2000), *Interactive Operations Research with Maple: Methods and Models*, Birkhäuser, Boston, MA.
- Parzen, E. (1960), *Modern Probability Theory and Its Applications*, John Wiley & Sons, Inc., New York, NY.
- Reingold, R. M., Nievergelt, J., and Deo, N. (1977), *Combinatorial Algorithms: Theory and Practice*, Prentice-Hall, Englewood Cliffs, NJ.
- Rohatgi, V. K. (1976), *An Introduction to Probability Theory and Mathematical Statistics*, John Wiley and Sons, New York, NY.
- Rose, C., and Smith, M. D. (2001), *Mathematical Statistics and Mathematica*, forthcoming, Springer-Verlag, New York, NY.
- Ross, S. (1998), *A First Course in Probability* (5th ed.), Macmillan College Publishing Company, Inc., New York, NY.
- Stockmeyer, P. (2001), Personal communication.
- Sveshnikov, A. A., Ed. (1968), *Problems in Probability Theory, Mathematical Statistics and Theory of Random Functions*, Dover Publications, Inc., New York, NY.
- Swain, J. (2001), "Simulation Software Survey: Powerful Tools for Visualization and Decision-Making," *OR/MS Today*, 28, pages 52–63.
- Thompson, P. (2000), "Getting Normal Probability Approximations without Using Normal Tables," *The College of Mathematics Journal*, T. Farmer, Ed., 31, pages 51–54.
- Trosset, M. (2001), Personal communication.
- Vivaldi, F. (2001), *Experimental Mathematics with Maple*, Chapman & Hall/CRC, London.
- Weiss, M. A. (1994), *Data Structures and Algorithm Analysis in C++*, Addison-Wesley Publishing Company, Menlo Park, CA.
- Wilks, S. S. (1962), *Mathematical Statistics*, John Wiley & Sons, Inc., New York, NY.

Vita

Diane Lynn Evans

Diane was born in Akron, Ohio on February 11, 1968. She received her B.S. (1990) and M.A. (1992) degrees in Mathematics from The Ohio State University, and an M.S. (1998) degree in Operations Research from the Mathematics Department at The College of William & Mary. Diane was awarded a two-year Clare Boothe Luce Fellowship in 1998 from the Applied Science department at William & Mary to work on her Ph.D. She has taught as an instructor in the Mathematics and Computer Science Department at Wittenberg University in Springfield, Ohio (1992–1994) and in the Mathematics Department at Virginia Wesleyan College in Norfolk, Virginia (1994–1998). She is a member of ASA, AMS, and INFORMS. In Fall 2001, Diane will begin teaching at Rose-Hulman Institute of Technology in Terre Haute, Indiana as an assistant professor in the mathematics department. Her email and web addresses are <devans@evansmath.com> and <www.evansmath.com/~devans>.