Future of Business and Finance

Moshe Arye Milevsky

How to Build a Modern Tontine

Algorithms, Scripts and Tips

Springer

Moshe Arye Milevsky

How to Build a Modern Tontine

Algorithms, Scripts and Tips

Springer

Moshe Arye Milevsky
Toronto, ON, Canada

IFID Centre has provided funding for this publication.

ISSN 2662-2467 ISSN 2662-2475 (electronic)
Future of Business and Finance
ISBN 978-3-031-00927-3 ISBN 978-3-031-00928-0 (eBook)
https://doi.org/10.1007/978-3-031-00928-0

It is no miracle that a man seemingly in good health should suddenly die because such a kind of death, though more unusual than any other has been frequently observed to happen. But it is a miracle that a dead man should come to life, because that has never been observed in any age or country.

David Hume (1748) on Miracles

Preface

Like a tontine itself, this book is somewhat of a gamble for me. I have placed a few months of diligent work and writing on what is essentially the **green zero** of a spinning roulette wheel in a Las Vegas casino. Allow me to explain.

If you can remember what a modern roulette wheel looks like, it has 36 numbers alternating between red and black, but also a single green zero on the outside edge of the numbers. This helps give *the house* its own unique edge. Most gamblers like to bet on sets of numbers between 1 to 36, either red or black, or a mixture and combination of rows and columns. Some players pick even numbers while other picks are odd, scattered and diversified across the entire felt table. And then, *when* the croupier spins the wheel and *if* it happens to land on the green zero, well everyone loses. All the chips get whisked away, the table is cleared in one large swoop and the round is over for everyone. New players are shocked to learn that a green zero is even "a thing." But the odds of living it are exactly 1-in-37, at least in an honest and fair table. This is yet another reason the casino always wins in the long run, with many thanks to the *law of large numbers.* Recall the casino pays out 35-to-1 for any given number, but there are 37 different things that can happen in a spin. If you are reading this book, then you should be able to do the math and realize that "selling" roulette is a good business.

But here is the curious fact, you can also bet on the green zero if you want. No, it's not very common, and if you hang around casinos and roulette tables, it's quite rare to see anyone place their chips on the green zero. That's usually the play of the oddballs and weirdos. Yet, it too pays out 35-to-1, no different than any other normal number on the table. Here is a fact: Zero is bet on less frequently. Watch and observe this behavioural factoid of human superstition. How many people would say zero is their lucky number? Born on the zeroth day of the month? The zeroth month? You get the point? Occasionally, and I do mean infrequently, someone does wander up and bets on the green zero. And then given the compound odds, rarely will you see a green zero winner. But, if it happens – and I have seen this only once or twice – everyone else at the table (who lost) suddenly takes notice of the oddball. They look at him or her just a bit differently, perhaps with a tad more envy and respect than reserved for the black and red winners. They think to themselves: "Well, that was clever to bet on zero" (no, just lucky.) It's all rather odd and asymmetric, but that's irrationality in action.

Well, back to this book. There are two colours of gambling chips in the retirement income and decumulation business. The black chips are the traditional investment and asset allocation strategies – such as stocks, bonds, ETFs, etc. – and the red chips are life insurance and annuity products. Some advisors and their clients bet exclusively on black, others bet only on red. Both groups have sophisticated rationales and even religions for the beliefs in how they have placed their chips. Perhaps a third modern and erudite group bet on both red and black numbers at the same time, diversifying their colours on the retirement table. They too have their pseudo-scientific reasons, beliefs and superstitions.

The many years of retirement consist of multiple spins of the roulette wheel where sometimes the black appears to have been the right choice – perhaps a *bullish* outcome in the stock market – and sometimes the red was the winner, think of the opposite or a *bullish* outcome in the longevity market. The *product allocation* diversifiers can harvest psychic benefits from both.

Ok now, I may be reaching the limits of this analogy, but this is what I mean by betting on the green zero. I'm sensing the retirement and decumulation universe might need a third option, one that doesn't only combine red and black chips but goes outside the normal and predictable numbers. The logic of an outside option might be as old as zero itself. According to historians of science, the number zero was "discovered" in Mesopotamia around the time BC became AD, and it has been a part of counting ever since. In the same spirit, the economic idea of survivors sharing, pooling and distributing a fixed known sum of money among themselves was familiar going back to mediaeval times. This is the essence of *tontine thinking.*

Aging wealthy and devout Christian benefactors would bequeath to their local church a fixed annual sum of money that was to be distributed among the poor on the anniversary of their death. Think of the rents from a farm or the modern-day coupons from a bond, split among the local needy survivors. Their medieval wills stipulated that in exchange for the periodic payments the indigent would commit to pray for the souls of their wealthy benefactors, ensuring eternal salvation. This donation or clause which was enshrined in their Latin wills and legacies was called *equalitos dividendos.* That's a *medieval tontine*, and predates Lorenzo's version by many centuries.

As we progress through the third decade of the twenty-first century, the world appears to be experiencing a resurgence of *tontine thinking.* Some global pension and retirement funds have launched elements of tontines, various participants in the financial sector have introduced products with mortality credits – which is the essence of tontines – and the eponymous *Financial Times* penned an editorial in mid-2021 urging regulators to "give tontines a chance."

I am betting that *tontine thinking* will in fact pay out over the next few years of the spinning retirement roulette wheel. Yes, it could very well be – and the odds do favour – an alternating combination of only red and black numbers. But, if indeed that little white ball ends up landing on the green zero, well then I'll be there to collect.

Toronto, ON, Canada
May 2022

Moshe Arye Milevsky

Acknowledgements

I would like to begin by thanking Joe Bisk for his extraordinary and dedicated assistance with this project. He served as a technical editor, R-script tester and co-wrote the solutions in Chap. 9. I can honestly say that without his help, this manuscript would have taken twice as long to write and would have been half as fun to create.

On the publishing side of the business, I would like to thank the editors at Springer Nature, namely Kirthika Selvaraju, Eva Hiripi, Amelie von Zumbusch and Laura Briskman (no longer at Springer), who convinced me to give them another chance (and then immediately left.) and especially Laura Briskman, who convinced me to give that publisher another chance.

I would also like to provide a shout-out to a number of people at *Guardian Capital* in Toronto, who expressed an interest in managing such a scheme and were the impetus for this book. Many of them provided feedback on my early ideas, read smaller segments and have been part of a laboratory experiment of sorts. In particular, I would like to thank Dino Bourdos, Barry Gordon, Denis Larose, Rohit Mehta, Adam Murl, Cesar Rivera, Humaira Omary and Candice Zhuang. I was challenged by each of them to explain things just a little bit better, and many of those (too) long email replies have migrated here.

In academia, a growing number of researchers, scholars and writers have taken an interest in *modern tontines* – albeit under different names – and I have benefited from conversations, seminars (or zoom sessions), with many of them. I'll give them proper credit in the literature review of Chap. 1, but at this point, I would like to highlight my friend, collaborator and colleague of over 30 years, Tom Salisbury. Although my original interest in tontines was kindled in the history archives, the mathematical foundations were set in his lectures on measure theory, rigorous probability and stochastic processes, in the early 1990s, at York University in Toronto.

Finally, no acknowledgement would be complete without a heartfelt *thank you* to my wife and partner of three decades, Edna Ida. It would have been impossible for me write a *17th* book without her support, encouragement and tolerance for 30 years in a row. In some sense, I got very lucky because she is my real green zero.

Contents

About the Author

Moshe Arye Milevsky, PhD is a tenured professor of finance at the Schulich School of Business and a member of the Graduate Faculty in Mathematics and Statistics at York University, in Toronto, Canada. He is an expert in the nascent field of **optimal decumulation**, and his book *King William's Tontine: Why the Retirement Annuity of the Future Should Resemble Its Past* was granted the Kulp-Wright Award from the *American Risk and Insurance Association* in 2017.

Moshe is an associate editor of *Insurance: Mathematics & Economics*, as well as the *Journal of Pension Economics & Finance*. He is also a fellow of the *Fields Institute for Research in Mathematical Sciences*, where he was previously a member of the board of directors and active in their scientific and commercial activities.

In addition to his scholarly work, Moshe is a well-known industry consultant, keynote speaker and fin-tech entrepreneur, with a number of US patents.

For more information about his recent work and interests, please visit his website at http://www.MosheMilevsky.com.

List of Figures

List of Tables

1 Why Tontines? Why Now?

In this chapter I provide some background on the reasons (I think) a traditional fund company might want to introduce a modern tontine as it relates to the unique challenges that people face managing their financial affairs towards the end of the human lifecycle. I also touch upon the difficulty retirees have in figuring out how to decumulate wealth and explain the difference between guaranteed income for life and the value of mortality and longevity credits.

1.1 Retirement vs. Decumulation

Australians retire with millions of dollars. It's not that Australia is necessarily wealthier than any other country, although they certainly rank high on a per capita basis. Rather, they happen to reach the traditional retirement age with millions of dollars in their retirement savings accounts. The source of (and credit for) their wealth is the Australian government who forces workers to save close to 10% of their salary in an investment account, and its employers mostly who are contributing to that pot. To put it in very simple terms, if your quoted salary is 100,000 AUD per year, your employer will guarantee and pay another 10,000 AUD which flows into your *super* (short for superannuation) *pot* as it's called. So, after a few decades of being forced to save that much money every year, and if the money is invested at a reasonable rate of return, it's not surprising it accumulates to millions of dollars at retirement. No other country has such a widespread system of forced retirement savings, also known as mandatory Defined Contribution (DC) Individual Account (IA) plans.

The problem with all these millions of AUD is that retiring Australians face a huge dilemma of what to do with all this money. Now sure that sounds like a super problem to have, but it's a scary one when you are looking at large sums that must finance your golden years of unknown length and duration. Australians can continue to invest the funds in the many different investment products they used during the

M. A. Milevsky, *How to Build a Modern Tontine*, Future of Business and Finance,
https://doi.org/10.1007/978-3-031-00928-0_1

accumulation phase, or withdraw their money and spend it slowly, or they can yank it out and buy a sailboat, which some do. Australians have lot of choices to make, with complex income-tax and old-age pension implications, which can be rather paralyzing and often leads to some very peculiar outcomes.

The Australian scheme has been in place for almost three decades now and has offered plenty of time to gauge how typical retirees behave—and what they actually do with their accounts—as they age and progress thru the lifecycle. One very large employer who managed a very large *super* in Brisbane, a lovely city in Queensland on the east coast of Australia, has carefully tracked the financial behaviour of tens of thousands of retirees during the last thirty years. Needless to say, many of the people who retired in their late 60s and early 70s are no longer alive three decades later, in which case the money in those accounts are transferred and bequeathed to surviving spouses, children and beneficiaries. But, after digging into all that spending and investing data over three decades of retirement, researchers in Brisbane noticed something very peculiar—and is an insight at the core of this book.

The Brisbane "discovery"—which is what I'll call this, with a shout out to Brnic Van Wyk was that on *average* the amount of money left in people's retirement account when they died was *equal* to the original sum they had started with when they retired decades before. Members ended their retirement journey with an average balance equivalent to when they started the journey. If they began with a million dollars at age 65, they ended with a million dollars. If they only had half a million in their pot when they exited the labour force, they left this world with half a million dollars, etc.

Now, to be very careful, the Brisbane "discovery" was a loosely defined average and there were many exceptions to this result, but the behavioural implications were even more interesting. Remember that the pension super pot wasn't sitting under their mattress or deposited in a bank account earning little interest income. The money was allocated to stocks (shares), bonds and many other investment asset classes over the 10, 20 or even 30 years of retirement. They had proper investment portfolios much like their brethren who were still accumulating. So, these account values fluctuated over time, bobbing up and down with markets and interest rates. They might have been tilted a bit more conservatively, but these pots earned dividends, interest and realized capital gains over time, which means that they increased and decreased from day to day and year to year. But again on average retirees adjusted and fine-tuned withdrawals and spending, so that the balance of the pension pot followed a rather flat trajectory over the long-run. How exactly? Well, if the pot grew in one year, the owner spent a bit more. If the pot shrunk, the owner would cut back and perhaps have one less "shrimp on the barbie", to overuse the Australian phrase. In economic terms, this might help smooth wealth but not consumption.

Australians had ***retired****,*
but they didn't ***decumulate.***

Decumulation is a relatively recent word, according to the Merriam Webster dictionary and is a noun defined as the *"disposal of something accumulated."* The

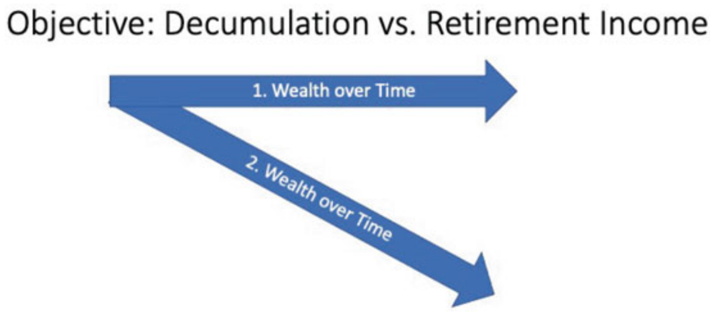


Fig. 1.1 What is your plan for the end of the life cycle?

word retirement is often used interchangeably with the word decumulation, but I hope you can see now that those are two very different terms (see Fig. 1.1). There are many experts in the field of economics, sociology, psychology, medicine and gerontology that study—and also give advice—on retirement. The former is about lifestyle choices, withdrawing from the labour force, spending quality time, etc. It's a massive field, well beyond the expertise of one person, one book or one department.

In contrast to that very large area, I am interested in decumulation, how people are doing it, how it should be done and whether there are things that governments and industry can do to make decumulation more efficient. And, to get to my main point, this book is about creating a new category of products—inspired by a very old product—that will help people do a much better and more efficient job of decumulation. That product is called a *Modern Tontine*. I claim that if modern tontines were widely available, it might be easier for Australians (and many others around the world) to both retire and decumulate. This book will explain how they work and how to build one. And, by the end of the book the hope and expectation is that you will see how this might help solve the Brisbane problem.

I should note that Australians aren't the only group who need help with decumulation, although they do provide an extreme and current example. In the USA, researchers have documented a similar albeit more nuanced phenomena, and one of the leading economists who has worked to uncover the drivers of decumulation is Professor James Poterba at MIT, with various colleagues. If I can quote from their article in the *Journal of Public Economics*, directly: "…The relatively modest age-related changes in wealth…suggest that the distribution of wealth near the end of life may be largely determined by wealth at age 65." Source: Poterba et al. [24]

Retirement and decumulation should be treated as distinct domains of activity and expertise. Technically speaking decumulation is an extraordinarily complex mathematical optimization problem for which insurance, risk management and stochastic control is the proper apparatus and modelling lens. Consumers who want (good, reliable) advice on this matter will have to pay more than they did for simple, easy, passive accumulation advice. To be clear, retirement is a slow ongoing progression in which people gradually withdraw from the paid labour force, often involuntarily and abruptly, with many non-financial externalities.

Intermediaries in the financial services industry must be careful not to veer from the technical domain of decumulation assistance—also encompassing the timing of retirement benefits such as Social Security—into the nebulous region of retirement planning. If I may digress just a little bit from the main objective of this chapter, I would suggest that eager 30-year-old advisors, brokers and insurance agents with little in the way of life experiences or financial assets (to be very blunt) should refrain from counselling financially successful middle-aged couples in their 50s, 60s and 70s, on exactly when they can afford to stop working, withdraw from the labour force and/or disengage from their commercial network. I suggest they leave that sort of advice to experienced gerontologists, social workers, psychologists and perhaps even family members who are in the best position to assess non-financial externalities. In sum, I would encourage my readership here who are interested in decumulation planning to learn more about the *modern tontine.*

1.2 Annuity Benefits: Credits vs. Insurance

I'm stumbling into a growing cloud of confusion creeping into the dialogue around guaranteed lifetime income. This fog is pervasive and thick within the neighbourhood of rationales and reasons offered to retirees as to why they might want to include annuities in their aging portfolios. Generally speaking these well-intentioned conversations centre around the longevity risk that is associated with living an unexpectedly long time, or the financial cost of becoming a centenarian and the benefit of pooling resources with a large group of similar retirees. Indeed, if only one or two people from a starting group of 20 reach the age of 100, then if everyone collectively combines a portion of their financial portfolio to support the few who are lucky enough to beat the odds and reach advanced ages, it will be cheaper for the entire group, etc. I have no quibbles with this line of reasoning and in fact have used most of these tropes myself. But to be clear and quickly get to the essence of this essay, there really are two different things going on within the life annuity story: credits and insurance. As an industry and as a community we must agree not to conflate them.

One aspect of the annuity story is the financial benefit of risk pooling, and the other is the insurance benefit and comfort from having a guaranteed income that you can't outlive. Again, those are two quite distinct features. And, right now I'm growing in favour of the former (credits) over the latter (insurance), which I tried to illustrate graphically in Fig. 1.2. This all might sound rather jumbled and theoretical, so allow me to elaborate with a statement that some readers might find bizarre. If you are 75 years old with $100,000 in your retirement account and would like to **guarantee** a protected annual income for the rest of your life, there is absolutely no need to purchase a life annuity. There are other options.

Longevity Credits
>=
Longevity Insurance

Fig. 1.2 Credits vs. insurance: can you tell the difference?

This might sound like something odd for an *annuity advocate* to say. But the fact is that I can assure you that if you politely ask a non-insurance company investment-bank or your favourite broker-dealer, for example, they can grab some inventory and design a lovely portfolio of zero-coupon strip bonds that will do the job. That collection of bonds will generate $4,000 per year for the rest of your life, even if you reach the grand old age of 115. Ok, BDs need to eat too, so they may not do it for $100,000, but I'm sure that a lump-sum of $1,000,000 will pique their interest and in exchange you will get $40,000 per year. Scale it up and they will come.

Moreover and with these strips, if you don't make it all the way to the astonishing age of 115, they will continue to send those $4,000 (or $40K) to your spouse, children or favourite charity until the date you would have reached 115, if you had been alive. This collection of strips would be completely liquid, tradeable and fully reversable, although subject to the vagaries of bond market rates. For those readers who dream of numbers, I have assumed a conservative, safe and constant 2.5% discount rate across the entire yield curve, which isn't entirely unreasonable in this environment. Think of the 30-year US treasury rate as a proxy, perhaps with a smidgen of corporate credit risk.

Stated technically, the present value of the $4,000 annual payments, for the 40 years between age your current 75 and your maximum age 115, is exactly equal to $100,000 when discounted at 2.5%. Yes, those numbers and ages were deliberately selected so my numerical example rhymes with the infamous 4% rule of retirement planning but has absolutely nothing to do with it. Now, if you are still with me—and perhaps have been trained by a good annuity wholesaler—I'm sure you must be thinking (or even yelling) "Moshe, but what if you live beyond age 115, eh? You will run out of money!"

Touché. Let's unpack that common knee-jerk reaction to a non-insurance solutions for a moment. To start with, the probability of becoming a supercentenarian—that is reaching age 110—is ridiculously and unquantifiably low. There are only about 30 of them (verified) in the USA, out of a population of 330,000,000. The chances of reaching age 115, remember that is when your strips run out, are even lower.

Up until the summer of 2021, the oldest living man in the world was Emilio Flores, who lived in Puerto Rico and died at the age of 112 years old. That's three years short of the terminal strip. And, if you do happen to be the one in a 100 million (or perhaps billion) that reaches age 115, I suspect you will have other things on your murky mind. Personally and post-covid, there is a very long list of hazards that worries me more than beating Emilio's record. Alas, some might argue that I'm neglecting medical breakthroughs and the risk we become a nation of Emilios. However, I'm more of a mortality compression-ist than extension-ist, which I'll explain in another essay. In English, you won't live to 115.

More importantly, nobody really "runs out of money" in retirement in the twenty-first century. That is plain utter fear mongering nonsense. With national social security programs in all developed countries, all Emilios will continue to receive some income for as long as they live even if they have completely emptied every piggy bank on their personal balance sheet. In fact, with tax-based means-testing you might get more benefits if you actually do empty your bank accounts.

Ok, back to my prior claim, if you want a guaranteed (liquid, reversable, bequeathable) income for the rest of your life, you can exchange your $100,000 for a bunch of strip bonds and voila, you have created a protected pension plan. My point here is that the primary objective isn't a guaranteed lifetime of income—which anyone can create with a simple discount brokerage account and a DIY instruction manual.

The goal is to get the HIGHEST possible income
and at the LOWEST possible cost.

Here we go. Transferring the above-noted $100,000 into an insurance company sold income annuity would result in a guaranteed income of $9,000 per year, which is $5,000 more per year, even if they cut-you-off (and forget to send you further payments) at age 115. That more-than-double number assumes the insurance company uses the exact same 2.5% interest rate to price their products, which they don't. In fact, if I do the same simple math with a valuation rate of 3.5% instead of 2.5%, the $100,000 would generate $10,000 per year at age 75, with the income annuity. Back to my favourite investment-bank or broker-dealer, they would need to price strips at north of 8% to get me that sort of income, which they obviously can't do unless the bonds are floated by some DDD country that will default well before I need dentures.

The reason for this rather magical jump to $9,000 from a mere $4,000 is that via the income annuity I have pooled my resources with many other similar 75-year-olds but have given up the assets in the event of early-death in exchange for a subsidy to those living longer, etc. If you are willing to forfeit the money when the longevity coin falls on tails, then you can benefit from heads, etc. If you're reading to this point, you know the drill.

To repeat, the motivation and rationale for the life annuity is *not* to necessarily generate a guaranteed lifetime income to some ridiculous age. Again, I can do that with simple discount bonds. Nor should the rationale be driven by the fear of running out of money in retirement. We don't do that to people. Rather, the legitimate concern retirees have is that their accustomed standard of living might be forcefully and involuntarily reduced if the markets don't cooperate, and/or they live longer than anticipated. That can be mitigated by pooling resources and benefiting from mortality & longevity credits that accrue to those who are willing to share with their neighbours. Moreover, those credits will be more valuable in states-of-nature in which markets are performing poorly and the rest of my investments have taken a tumble. The implicit 8% return from a fixed-income product is the magic of longevity credits, a term that sounds better than mortality credits.

This distinction between longevity credits versus longevity insurance opens the door for a universe of pooling products that don't necessarily guarantee (or even offer) to pay income for the rest of your life or guarantee anything for that matter. One can harvest mortality & longevity credits without requiring a rest-of-life horizon. For example, imagine a pooling arrangement that lasts for 25 or 30 years in which survivors inherit the investment assets of the deceased, thus acquiring mortality & longevity credits, but the entire fund is designed to be wound up when everyone reaches some predetermined age. Actuaries will recognize this as a temporary life annuity. That fund might not promise longevity insurance per se, or income for the rest of your life, or money that you can't outlive. But it would certainly include generous longevity credits. That is the *modern tontine.*

1.3 How Does This Book Differ from All the Other Books?

For starters, this isn't a book about seventeenth century financial or insurance history nor is it a collection of theorems and proofs about the mathematical properties of tontines. I have written both of those already, cited as Milevsky [20] and Milevsky and Salisbury [19], so that certainly isn't my intention. Indeed, I plan to stay very focused on the present day twenty-first century and will be using an absolutely minimal amount of mathematics. In that sense, I would describe what follows in the next 8–10 chapters as an instruction manual on how to build a modern tontine for decumulation. Those instructions or directions will be written in a language called **R**, which enables the users to copy-and-paste algorithms and generate their own results. Readers will be able to quickly and easily reproduce every result, number or figure using their own parameters and assumptions.

The growing media and commercial interest in *modern tontines* over the last decade has convinced me that this might be a good time to write a cook-book or instruction manual, accessible with a minimal level of technical background. If you can download R (which is free) and write some basic code (which everyone should), then you too can build your own *modern tontine*.

With the above in mind, the ideal audience for this book is practitioners or advanced students looking for a thesis project. The first group are financial engineers working for asset managers, financial services companies and even start-ups who are building *modern tontine* funds and would like to quickly and easily stress test various design and behavioural features. Perhaps the reader has been tasked with building or coding-up the expected payout from a *modern tontine.* What if more or less people die? What if returns are higher or lower? What if interest rates move up or down? What if investors surrender earlier or later? These and other questions will likely be on the mind of tontine sponsors, and the algorithms and R-scripts provided in this book will help shed light and perhaps even answer those questions. For those readers who are familiar with my prior book on retirement income recipes, cited as Milevsky [21], this work can be viewed as a sequel.

A secondary audience for this book is students—both graduate and undergraduates in finance, economics or even business—who are interested in quickly getting up to speed on how a *modern tontine* works in practice. Notice that I didn't mention insurance actuaries, who tend to use their own language, notation and framework for managing risk. In other words, if you know nothing about mortality and longevity tables but realize that some minimal knowledge of those topics is absolutely necessary for properly thinking about *modern tontines*, then this book is for you. This book is therefore not meant as a comprehensive review of the scholarly literature on tontines. My objective isn't for this book to be *cited* by other scholars, but rather to be used by practitioners. And, the algorithms included with this book—albeit not very sophisticated or unique—can be considered my modest contribution to a free and open-source movement for *modern tontines*.

1.4 Outline and Plan

1. **Introduction and Motivation:** This Chapter You are Reading.
2. **Actuarial Background:** A Crash Course in Mathematics.
3. **Core Simulation in R:** Introducing the Secret Sauce.
4. **Statistical Risk Management:** What Drives the Outcomes?
5. **Introducing Death Benefits:** Promises without Guarantees.
6. **Richer Returns & Models:** More Realistic Investments.
7. **Mortality of the Future:** Benjamin Gompertz is Old.
8. **Running a Business:** Theory vs. Practice.
9. **Solutions and Tips:** End-of-chapter Questions Solved (with Joe Bisk).
10. **Conclusion:** Final Thoughts.

1.5 Tontine Literature: What (and Who) Else to Read

The traditional approach to writing a review of the literature is to create a (very) long list of articles that are pertinent to the topic being discussed and to provide a brief overview and summary of those articles. The objective of that classical activity is to embed the author's own work into the *literature*, offer credit where it is due and then progress to carving-out the newer contribution by the author. But within the context of this particular book, I believe such a list of articles might be unnecessary and redundant. This book really isn't intended to be a contribution to the scholarly literature and it certainly isn't a tontine history book, both of which have been attempted elsewhere and by others.

Rather, I am assuming that the reader(s) of this book—hopefully more than singular—are interested in building a *modern tontine* and as part of that process would like to learn who *else* they should be learning from. So, what follows is a list of writers, researchers and scholars who (I know) have spent much time and energy *thinking about* the design of *modern tontine* products. They have worked-on and contributed to the **technical** actuarial development of instruments with similar aims and objectives, but perhaps with different names. I list them here in alphabetical order and highlight one or two of their research papers or monographs. My intent here is not to be exhaustive or even to highlight the research paper they would most likely recommend. Rather my point with this list is to encourage you to google, carefully read and dig deeper for their latest *tontine thinking*.

1.5.1 A. Chen

In a series of articles with a number of her students and co-authors, the prolific Ann Chen [7] introduces many innovations into the basic tontine design, including options on tontines, tontines that allow for bequest and legacy, tontines that are linked to health status and optimal allocations to a mixtures of tontines and annuities. Under her tutelage, *Ulm University* in Germany has become something of a modern factory for tontine research, and more generally strategies for managing personal longevity risk.

1.5.2 J.M. Bravo

In an article cited as Bravo [6], this Portuguese professor of economics examines something (he calls) *participating longevity-linked life annuities* (PLLA), in which benefits are updated periodically based on the observed survival experience of a given underlying population and the performance of an underlying investment portfolio. As you might sense, this is a type of *modern tontine*. Likewise, in a series of articles with well-known global pension expert R. Holzmann and various co-authors they address how to incorporate longevity heterogeneity into the design of modern retirement plans.

1.5.3 C. Donnelly

In a series of papers, including a recent one cited as Bernhardt and Donnelly [4], the Scottish actuary and professor of mathematics has conducted and published a number of research projects funded by the *UK Institute and Faculty of Actuaries* on the optimal way for individuals to manage personal longevity risk. In particular, she has helped further *tontine thinking* by emphasizing the costs embedded in traditional (capital intensive) annuities and the point at which consumers are willing to "risk" their retirement on *modern tontines* to avoid those fees.

1.5.4 R.K. Fullmer

An American practitioner who is actively engaged in designing and engineering the next generation of *modern tontines*, the monograph cited as Fullmer [12] is a *CFA Institute* publication that provides a detailed and practitioner-accessible explanation of how tontines actually work. See his work with M.J. Sabin on tontine bond ladders, and his work with law professor J.B. Forman on how *tontine thinking* can be legally embedded inside modern pension funds.

1.5.5 M. Guillen

A well-known and prolific researcher in statistics and actuarial science based in Barcelona, Spain, M. Guillen has recently focused her attention on tontines and pooled annuity funds, which recall is yet another one of the many names for modern tontines. In a paper with J.P. Nielsen, cited as Bräutigam et al. [5] she compares pooled annuity overlay funds based on actuarial fairness, to equitable retirement income tontines and notes that "the market would appear to be ready for such innovations" and I wholeheartedly agree.

1.5.6 S. Haberman

One of the original deans of actuarial science research based in London, and focused on longevity risk management, S. Haberman together with co-authors, cited as Denuit et al. [8] suggested a number of innovative ways in which *modern* annuities can be designed, by sharing and pooling longevity risk. He has supervised many graduate students at Bayes Business School (formerly Cass Business School) at City University of London, which is yet another power-house of research in this area.

1.5.7 J. Piggott

The director of the Australian Research Council Centre of Excellence in Population Ageing at the University of New South Wales, Professor Piggott was one of the first classically trained economists to direct his attention to the design of (better) longevity risk pooling arrangements. In particular, the article cited as Piggott et al. [25] is the earliest to suggest how to build a pooled annuity fund which is another type of *modern tontine*. They called this arrangement *group self-annuitization*, explained how to mix cohorts of different ages into one large pool and were able to prove the existence and uniqueness of their design. That article has formed the basis of many follow-up studies and has been widely cited in this literature.

1.5.8 E. Pittacco

In a series of articles with his co-author A. Olivieri, cited as Olivieri and Pittacco [23], this Italian demographer, actuary and expert of longevity dynamics examines the many different ways in which longevity risk can be shared. His extensive research work also discusses capital requirements in the presence of guarantees, as well as the (complicated) problem of forecasting future mortality.

1.5.9 R. Rogalla

A researcher who is interested in the design of annuities, as well as the management of longevity risk over the lifecycle. In a series of articles, including the one cited as Gemmo et al. [14], their results indicate that early on in retirement, a tontine is an attractive investment option if the tontine funds are invested in a risky asset, which is precisely the approach taken in the next few chapters. He is based at the Maurice R. Greenberg School of Risk Management, Insurance and Actuarial Science (SRM) at St. John's University in New York.

1.5.10 T.S. Salisbury

A noted Canadian probabilist and mathematician based at York University, T.S. Salisbury developed an interest in actuarial science, annuities and then tontines while supervising graduate students (including the author of this book). In a series of research articles beginning with Milevsky and Salisbury [19], he helped rekindle a discussion about retirement income product that has been long neglected, and then leveraged economic theory and tools from mathematical finance to design the next generation of tontine annuities.

1.5.11 M.J. Sabin

An American entrepreneur M.J. Sabin (2010) proposed what he called a *fair tontine annuity* which is an arrangement that provides a lifetime payment stream whose expected present value matches that of a fair annuity. The article itself was never peer-reviewed and published in an academic journal (unfortunately) but is available as an SSRN research paper and has been quite influential within the literature on modern tontines. Although his original work is a based on a one-period model and static (known) mortality rates, his idea can easily be extended to multiple time periods.

1.5.12 M. Sherris

A prolific Australian mathematician who has supervised many students and projects on longevity risk. In particular, the article, cited as Qiao and Sherris [26], addresses challenges for designing group pooled schemes that include decreasing average payments when mortality improves significantly, decreasing numbers in the pool at older ages, and the impact of dependence from systematic mortality improvements across different ages of members in the pool. This article uses a multiple-factor stochastic mortality model in a simulation study to show how pooling can be made more effective and to quantify the limitations of these pooling schemes arising from the impact of systematic longevity risk.

1.5.13 M.J. Stamos

In one of the earliest papers in this genre, cited as Stamos [28], and then in a series of papers under the supervision of R. Maurer at Goethe University in Germany, Stamos merged ideas from asset allocation and personal longevity risk management. He noted that pooled annuity funds insure very effectively against longevity risk even if their pool size is small. Furthermore, he showed (again, rather early on) that only very risk averse investors would prefer to pay a risk premium (i.e. higher costs) to access conventional life annuities that completely eliminate longevity risk.

1.5.14 J.H. Weinert

In research work that grew out of his Ph.D. dissertation in Germany under the supervision of his professor H. Grundel, and then published as Weinert and Grundel [29], he uses insights from behavioural economics and Prospect Theory to design better tontines. As part of his thesis work he conducted a comprehensive survey of *modern tontine* products that are available around the world in various guises and forms.

References

1. Ayuso, M., Bravo, J. M., Holzmann, R., & Palmer, E. (2021). Automatic indexation of the pension age to life expectancy: When policy design matters. *Risks, 9*(5), 1–28.
2. Boon, L. N., Brière, M., & Werker, B. J. (2020). Systematic longevity risk: To bear or to insure? *Journal of Pension Economics & Finance, 19*(3), 409–441.
3. Bernhardt, T., & Donnelly, C. (2019). Modern tontine with bequest: innovation in pooled annuity products, *Insurance: Mathematics and Economics, 86*, 168–188.
4. Bernhardt, T., & Donnelly, C. (2021). Quantifying the trade-off between income stability and the number of members in a pooled annuity fund. *ASTIN Bulletin: The Journal of the IAA, 51*(1), 101–130.
5. Bräutigam, M., Guillén, M., & Nielsen, J. P. (2017). Facing up to longevity with old actuarial methods: a comparison of pooled funds and income tontines, *The Geneva Papers on Risk and Insurance-Issues and Practice, 42*(3), 406–422.
6. Bravo, J. M. (2021). Pricing participating longevity-linked life annuities: A Bayesian model ensemble approach. *European Actuarial Journal*, 1–35.
7. Chen, A., Hieber, P., & Klein, J. K. (2019). Tonuity: A novel individual-oriented retirement plan, *ASTIN Bulletin: The Journal of the IAA, 49*(1), 5–30.
8. Denuit, M., Haberman, S., & Renshaw, A. (2011). Longevity-indexed life annuities, *North American Actuarial Journal, 15*(1), 97–111.
9. Donnelly, C. (2015). Actuarial fairness and solidarity in pooled annuity funds, *ASTIN Bulletin: The Journal of the IAA, 45*(01), 49–74.
10. Donnelly, C., Guillén, M., & Nielsen, J. P. (2014). Bringing cost transparency to the life annuity market. *Insurance: Mathematics and Economics, 56*, 14–27.

11. Forman, J., & Sabin, M. J. (2015). Tontine pensions, *University of Pennsylvania Law Review, 163*(3), 755–832.
12. Fullmer, R. K. (2019). *Tontines: A practitioner's guide to mortality-pooled investments*. CFA Institute Research Foundation.
13. Fullmer, R. K., & Sabin, M. J. (2019). Individual tontine accounts. *Journal of Accounting and Finance, 19*(8), 31–61.
14. Gemmo, I., Rogalla, R., & Weinert, J. H. (2020). Optimal portfolio choice with tontines under systematic longevity risk. *Annals of Actuarial Science, 14*(2), 302–315.
15. Goldsticker, R. (2007). A mutual fund to yield annuity-like benefits. *Financial Analysts Journal, 63*(1), 63–67.
16. Iwry, J. M., Haldeman, C., Gale, W. G., & John, D. C. (2020). *Retirement tontines: A new way to finance retirement income*. Washington: Brookings Policy Brief.
17. Li, Y., & Li, C. (2020), Selection and redistribution in the Irish tontines of 1773, 1775, and 1777. *Journal of Risk and Insurance, 87*(3), 719–750.
18. Maurer, R., Mitchell, O. S., Rogalla, R., & Kartashov, V. (2013). Lifecycle portfolio choice with systematic longevity risk and variable investment - Linked deferred annuities. *Journal of Risk and Insurance, 80*(3), 649–676.
19. Milevsky, M. A., & Salisbury, T. S. (2015). Optimal retirement income tontines. *Insurance: Mathematics and Economics, 64*, 91–105.
20. Milevsky, M. A. (2015). *King William's tontine: Why the retirement annuity of the future should resemble its past*. Cambridge: Cambridge University Press.
21. Milevsky, M. A. (2020). *Retirement income recipes in R: From ruin probabilities to intelligent drawdowns*. New York: Springer Nature.
22. Olivieri, A., & Pitacco, E. (2020). Linking annuity benefits to the longevity experience: Alternative solutions. *Annals of Actuarial Science, 14*(2), 316–337.
23. Olivieri, A., & Pitacco, E. (2020). Longevity-linked annuities: How to preserve value creation against longevity risk. In: M. Borda, S. Grima, & I. Kwiecień (Eds.), *Life insurance in Europe. Financial and monetary policy studies* (Vol. 50). New York: Springer.
24. Poterba, J., Venti, S., & Wise, D. A. (2018). Longitudinal determinants of end-of-life wealth inequality. *Journal of Public Economics, 162*, 78–88.
25. Piggott, J., Valdez, E. A., & Detzel, B. (2005). The simple analytics of a pooled annuity fund. *Journal of Risk and Insurance, 72*(3), 497–520.
26. Qiao, C., & Sherris, M. (2013). Managing systematic mortality risk with group self-pooling and annuitization schemes. *Journal of Risk and Insurance, 80*(4), 949–974.
27. Sabin, M. J. (2010). *Fair tontine annuity*. Available at SSRN # 1579932.
28. Stamos, M. Z. (2008). Optimal consumption and portfolio choice for pooled annuity funds. *Insurance: Mathematics and Economics, 43*(1), 56–68.
29. Weinert, J. H., & Gründl, H. (2020). The modern tontine. *European Actuarial Journal, 11*, 49–86.

Financial and Actuarial Background 2

This chapter summarizes the main technical background on financial and actuarial modelling required, with particular emphasis on present values, stochastic investment returns, the Gompertz law of mortality and the valuation of temporary life annuities which are very close cousins to *modern tontines*. To be clear, this chapter is a review of background and notation. It's definitely not the place to visit to learn anything new.

2.1 Setting the Stage

Figure 2.1 is a summary or schematic which lays out the plan for the next few chapters and the algorithmic objective of this book. The *modern tontine* simulation I am building combines two different types of stochastic models, one for investment returns and the evolution of a portfolio (in the top left corner) and the other for projecting the number of survivors and future mortality rates more generally (in the top right corner). Those two projections or random generators (i.e. the top corners of the triangle) are then combined deterministically with a very specific *sharing rule* that feeds into the bottom corner, the periodic dividends or payout to investors. The point is to get to the bottom of the triangle and give investors a sense or range of what they can expect if they elect to participate in a *modern tontine.*

Philosophically and for most of this book I will assume that investment returns are the so-called Lognormally distributed and that mortality obeys the Gompertz law. Both of which will be carefully explained if you have never heard of them before, and carefully defended if what you heard hasn't been kind. More importantly, the *sharing rule* is designed to amortize and distribute the market value of the underlying fund over a fixed and shrinking time horizon, for example, 30 years. Now, in Chap. 5 I will augment the algorithm and allow for death benefits and voluntary surrenders as yet another way for people to leave the fund with (some of) their money. That will require some tinkering and adjustments to the *sharing*

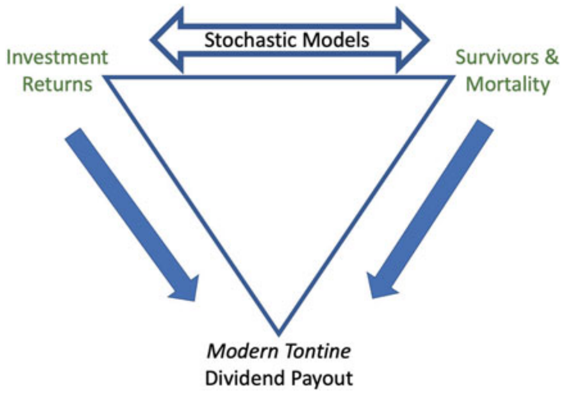


Fig. 2.1 Schematic diagram of a *modern tontine* algorithm

rule, since it can't be as generous. Then, Chap. 6 will focus on the upper left-hand corner and go beyond Lognormal returns, hoping to assuage the investment experts. Chapter 7 (focusing on the upper right-hand corner) will go beyond Gompertz, hoping to assuage the actuaries and demographers. But the over-arching objective of the **R**-script is to move from the dual base modelling assumptions to the bottom corner, projected dividends and payouts.

In other words, the algorithm or **R**-script is designed in a modular manner so that if you (e.g. the CFA) do not approve of my upper left-hand corner, then you can use your own economic forecasts and financial scenarios in that black box. Likewise, if you (e.g. the FSA) do not like my models for life and death, you can include your favourite actuarial basis or N-parameter mortality models. Just as importantly, both groups (CFAs and FSAs) can independently tinker with the tops and then merge their favourite models into the bottom corner of the triangle. I should also note that I will be assuming independence between markets (left) and mortality (right), despite the growing evidence (and pandemic) those two might have complex, non-linear and lagged dependencies. The advanced reader (e.g. the PhD) might want to introduce some (lagged) dependencies between the two main sources of uncertainty or might dispense with my *sharing rule* altogether, and devise their own. The only thing that shouldn't change is the upside down triangle, or the flowchart in Fig. 2.1.

What remains to be done here before the coding begins is to introduce some basic notation, terminology and *stochastic ideas* at the core of the top two corners of this triangle. The objective in the next few pages of this chapter then is to get all the mathematics out of the way so that the next few chapters can progress with as little formality, symbols or math as possible. Needless to say, please treat the next few pages as a review or a crash-course, as opposed to a proper stand-alone introduction to stochastic modelling for investments and mortality.

2.2 Investment Returns: Notation and Moments

Here I will focus on investment returns (only) and use the symbol $\tilde{R}[i, j]$ to denote the random *effective periodic* investment return during the j'th period, for the i'th simulation run. So, if we are generating 10,000 annual investment returns over a period of 30 years, the counter (row) value of i ranges from a minimal value $i = 1$ to maximal value of $i = 10{,}000$ and the counter (column) value of j ranges from $j = 1$ to $j = 30$. In this case it would create 300,000 random numbers in total.

For greater clarity, what I mean by *effective periodic* investment return is that if you invest \$100 at the beginning of the period, then it will be worth $\$100 \times (1 + \tilde{R})$ at the end of the period, where $\tilde{R}$ is shorthand notation for the full $\tilde{R}[i, j]$. In the tontine simulation, I will for the most part assume that $(1 + \tilde{R}[i, j])$ is *lognormally* distributed, which is synonymous with: $\ln[1 + \tilde{R}[i, j]]$ being *normally* distributed. Indeed, another name for the quantity: $\ln[1 + R[\tilde{i}, j]]$ is the *continuously compounded* (CC) investment return (IR), denoted by its own: $\tilde{r}[i, j]$, where $\tilde{R}[i, j] = e^{\tilde{r}[i,j]} - 1$.

The mean or expected value of the (theoretical) CC-IR is equal to and denoted by ν, and the standard deviation of the (theoretical) CC-IR is σ. Remember that the underlying normality assumption imposes very tight restrictions on the 3rd and 4th moment, namely that the skewness is zero and the kurtosis is 3, both of which I will now describe and derive via their moments.

Computationally, the first *central moment* of the CC-IR in the j'th period is denoted by and computed as follows:

$$M1[j] := \frac{1}{N}\sum_{i=1}^{N} \tilde{r}[i, j] \ \sim \nu \tag{2.1}$$

This is (a fancy name for) the average or mean, which when computed over a large sample size N should be very close to the (theoretical) parameter ν. To further clarify notation, when $j = 1$, which is the first period (month, quarter, year), the first central moment of the CC-IR is: $M1[1] = \frac{1}{N}\sum_{i=1}^{N} \tilde{r}[i, 1]$, and in the second period it's: $M1[2] = \frac{1}{N}\sum_{i=1}^{N} \tilde{r}[i, 2]$, etc. Therefore, if the simulation model operates over T distinct periods, then a simulation run should generate T values of $M1$, all of which should be (very) close to ν and converge to ν when $N \to \infty$.

Now, in theory we could simulate a very complicated process for $\tilde{R}[i, j]$ over the time period j (e.g. GARCH, Jump Diffusion, Stochastic Volatility), under which the $\tilde{r}[i, j]$ values are **not** independent and identically distributed (i.i.d.) normal variables. Even then, we would use the same notation and would compute logarithms of the one-plus investment return: $\ln[1 + \tilde{R}]$, before any sample statistics were computed. The convention is to work with the CC-IR, even if it isn't normal.

Moving on, the second central moment of the CC-IR during the j'th period is defined as follows:

$$M2[j] := \frac{1}{N} \sum_{i=1}^{N} (\tilde{r}[i, j] - M1[j])^2 \tag{2.2}$$

This is the variance, which implies that the standard deviation (a.k.a. volatility) would be: $\sqrt{M2[j]} \sim \sigma$. Note that in this formulation we divide by N, not $(N - 1)$, which is yet another convention. I'll emphasize (again) that in the core simulation model the return-generating process is: (i) stationary, and (ii) normal, so the volatility $\sqrt{M2[j]}$ is unchanged over time or periods.

Moving on to the higher moments for the purpose of defining skewness and kurtosis, the third central moment is computed as follows:

$$M3[j] := \frac{1}{N} \sum_{i=1}^{N} (\tilde{r}[i, j] - M1[j])^3 \tag{2.3}$$

The rhythm or pattern should by now be familiar. We subtract off the first central moment and then raise to the power of whatever central moment we are interested in computing. This leads to the fourth (and final, for our purposes) central moment of the CC-IR, which is defined and computed as:

$$M4[j] := \frac{1}{N} \sum_{i=1}^{N} (\tilde{r}[i, j] - M1[j])^4 \tag{2.4}$$

Every period (month, quarter, year) in the simulation will be associated with these four quantities, $M1$, $M2$, $M3$ and $M4$, which are then combined and scaled for the skewness and kurtosis. Investment strategies that have a large option component, that is puts and calls, will be associated with very different higher moments and will obviously impact tontine dividends. Formally, the Fisher coefficient of *skewness* is:

$$\texttt{skewness}[j] = \frac{M3[j]}{M2[j]^{3/2}} \tag{2.5}$$

It's the third central moment scaled by the standard deviation to the power of three. Finally, the coefficient of *kurtosis* is computed as follows:

$$\texttt{kurtosis}[j] = \frac{M4[j]}{M2[j]^2} \tag{2.6}$$

I sum up with the following three reminders. First, the (theoretical) skewness of $\tilde{r}$ is zero, and the (theoretical) kurtosis is 3, when the CC-IR $\tilde{r}$ is normally distributed. But in a finite sample (especially with small values of N) the estimates might deviate from (0) to (3). Second, and just as importantly, the *effective periodic* investment return which I first denoted by $\tilde{R}$ will **not** generate a skewness value of zero, or a kurtosis value of 3. In all likelihood the skewness of $\tilde{R}$ itself will be positive, due to the exponentiation of $\tilde{r}$. Third and in conclusion, skewness and kurtosis are defined across the *columns* and not the *rows* of the vector $\tilde{R}$, but after the logs have been taken.

2.3 But Why Logarithms?

There are a number of reasons why all investment variables are computed—and investment statistics are compiled—based on the logarithm of one-plus return, and why I prefer to use $r := \ln[1 + R]$, instead of R. Some of those reasons are based on historical conventions and others are driven by convenience.

Historically, the most famous formula in finance, namely the Black-Scholes-Merton (1973) equation for the value of an option is expressed in terms of the annual risk-free interest rate ν and the volatility σ, both of which are the *continuously compounded* rates. In other words, they assume that the **logarithm** of the underlying security process obeys a Brownian motion with standard deviation σ. Hence, to use the BSM formula you have to estimate and then use the standard deviation of the logarithm of the (stock, commodity, currency) price, plus one. Sure, F. Black, M. Scholes and R. Merton could have re-written their formula in terms of the standard deviation of the actual security price—not the one-plus logarithm—but it would have been messier and unnecessary.

Ergo, if the second moment is computed *after* logarithms, then for consistency it makes sense to do the same for skewness and kurtosis. Of course, when your (volatility) numbers and (investment) returns are small, these two methods (with and without logs) will lead to statistical estimates that are quite close and perhaps indistinguishable from each other, but when rates get higher it matters.

Beyond historic tradition and convention, there are other reasons to (only) work with the natural (not base ten!) logarithm of one-plus effective investment returns. That has to do with lower and upper bounds. The effective investment return can (at worst) be minus 100%, or $R = -1$; the entire capital is lost. But in theory its upper bound could be infinity. Mathematically, the random variable $R \in [-1, +\infty)$. This awkward truncation at minus one makes it difficult to fit a continuous univariate distribution to the underlying investment return process. Taking logarithms solves this modelling problem, since $\ln[1 + R]$ as $R \to -1$ is $-\infty$, and $\ln[1 + R]$ as $R \to \infty$ is $+\infty$. The logarithm of one-plus effective investment return lives on $r := \ln[1 + R] \in (-\infty, +\infty)$ which enables a wider class of probability distributions, with the most common being normal.

Finally, when working with log returns, the growth is additive. This means that if the *continuously compounded* investment return (CC-IR) during one period is r_1 and the CC-IR during a subsequent period is r_2, we can add them up without having to worry about compounding periods. It's just easier to work with. The same thing goes for variance σ^2. The annual variance is the sum of the monthly variances. The monthly volatility is $\sigma/\sqrt{12}$, etc. That only works with logarithms. In fact, get used to working with logarithms because in the next section, when I discuss models of mortality, they will also be expressed in terms of logs.

2.4 Gompertz Survival Probabilities

A key ingredient of the *modern tontine* simulation presented and developed in this book is the survival probability curve, that is the probability an individual investor who is alive at age x will survive for the next t years and be entitled to a dividend payout at age $(x + t)$. This is also denoted by $\Pr[T_x \geq t]$, where T_x represents the *remaining lifetime random variable*. In most of what follows in the next few chapters that critical probability will be constructed based on the so-called Gompertz law of mortality which I will briefly explain.

The main insight of this law of mortality—cited as Gompertz [4]—and for which he is (still) famous is that the natural logarithm of adult mortality hazard rates are linear. If one denotes the instantaneous hazard rate—that is the rate at which people die—by the symbol λ_x, where x is age, then $\lambda_x = h_0 e^{gx}$. Using this formulation h_0 and g are constants, which equivalently means that $\ln[\lambda_x] = \ln[h_0] + gx$, where h_0 is an (initial) time-zero mortality hazard rate, and g is the mortality growth rate.

Using the above parameterization, which is quite common in the demographic literature, it's easy to see that mortality rates grow exponentially with (adult) age. For the purpose of what follows (and most of this book) I'll re-parametrize using a slightly different formulation. In particular, I'll assume the following functional form for the mortality hazard rate function, a.k.a. the mortality rate:

$$\begin{aligned}\lambda_x &= \frac{1}{b}\mathrm{e}^{(x-m)/b} \\ \ln[\lambda_x] &= \overbrace{-\ln[b] - m/b}^{\ln[h_0]} + \overbrace{(1/b)}^{g}\,x \end{aligned} \tag{2.7}$$

At first this might seem overly complicated, relative to a simpler "hazard rates are exponential" via $\lambda_x = h_0 e^{gx}$ specification. But there are good reasons for writing the mortality hazard rates in this manner. The parameters (m, b) have a real tangible interpretation associated with ones expectations of life. Nevertheless and regardless of the exact parameter specification, the Gompertz model survival probability can be expressed as follows:

$$\Pr[T_x \geq t] = \exp\{\mathrm{e}^{(x-m)/b}(1 - \mathrm{e}^{t/b})\} \tag{2.8}$$

Using this formulation, the parameter m represents the modal value (in years) of the distribution and the b parameter represents the dispersion (in years) coefficient. For example, I might say that your random lifetime has a modal value of $m =$ 90 years and a dispersion coefficient of $b = 10$ years. Note that the modal value isn't the mean (average) value, and that the dispersion coefficient isn't quite the standard deviation either, although both are close. This particular specification of the mortality rate is known as the pure Gompertz assumption, which will play a very central role in the *modern tontine* simulations. I will now create our first function in **R**-script, which will be used over (and over) again in what follows. It is called the `TPXG` function, based on Eq. (2.8).

```
TPXG<-function(x,t,m,b){exp(exp((x-m)/b)*(1-exp(t/b)))}
```

Notice that the function has four arguments (x, t, m, b). The first argument is the current (conditioning) age x, the second argument is time t, and the third and fourth arguments are the Gompertz (m, b) parameters. Here are some numerical examples.

```
round(TPXG(65,35,90,10),digits=3)
> 0.072
round(TPXG(85,15,90,10),digits=3)
> 0.121
```

The interpretation is as follows. If you are (a.k.a. conditional on) age $x = 65$, the probability of surviving to age $y = 100$, which is $t = 35$ years, is 7.2%, under a Gompertz model with parameters $m = 90$ and $b = 10$. But, if you are already (conditional on) age $x = 85$, the probability of surviving to age $y = 100$, which is $t = 15$ years, is 12.1%. The older you are, the greater the probability of reaching any given age.

Now, for the sake of completeness (and an independent verification) I will compute the 15-year survival probability by asking **R** to integrate the mortality hazard rate curve numerically. To do this properly (and without error messages) I must set all the Gompertz parameter values (first) and then define the mortality rate function in terms of time (only). Then, I can perform the numerical integration, between $t = 0$ to $t = 15$. The result (to 3 digits) is as follows:

```
m<-90; b<-10; x<-85
lambda<-function(t){-(1/b)*exp((x+t-m)/b)}
round(exp(integrate(lambda,0,15)$value), digits=3)
> 0.121
```

The analytic result from Eq. (2.8) is consistent with the brute-force numerical integration displayed above. Note that the lower the Gompertz modal value m, the earlier the person is expected to die and the survival probability to any future age should be lower. To be clear, the modal value of life m is the age at which this person is most likely to die, which is different (and actually higher) than the mean lifetime, denoted by $E[T_x]$, in a Gompertz model. I can use numerical routines in **R** to obtain these values. In particular, I can leverage the extremely useful and fundamental construction of the expectation as the integral of the survival probability function.

$$E[T_x] = \int_0^\infty \Pr[T_x \geq t]dt. \tag{2.9}$$

Here is a short script in **R** that computes $E[T_x]$, using the numerical integration routine I noted earlier, in conjunction with the `TPXG(.)` function I just defined and created.

```
m<-90; b<-10; x<-65
theta<-function(t){TPXG(x,t,m,b)}
round((integrate(theta,0,45)$value), digits=2)
[1] 21.75
```

The lower bound of integration is zero (years) and the upper bound is 45 (years), after which the survival probability is assumed to be negligible. So, at $x = 65$, your expected remaining lifetime is 21.75 years, and your expected age at death is 86.75, whereas your modal age at death is $m = 90$. Note the difference between mean, mode as well as median age at death, which is the $(x + t)$, at which `TPXG(x,t,m,b)=0.5`.

Finally, the expectation of remaining life at birth can be expressed in terms of Euler's constant $\gamma_e := 0.5772$, and the standard deviation can be approximated in terms of: $\pi = 3.1415$, both of which are given here without any proof or justification.

$$\begin{aligned} E[T_0] &= m - b\gamma_e \approx m - (0.5772)b, \\ SD[T_0] &\approx b\frac{\pi}{\sqrt{6}} \approx (1.28255)b, \end{aligned} \tag{2.10}$$

which implies that (at age $x = 0$) the standard deviation of life is greater than the dispersion coefficient b. Finally, for those readers interested in a deeper understanding of mortality models in general and the Gompertz model in particular, I would suggest you consult the companion (2020) book called: *Retirement Income Recipes in R*, from which most of this section has been sourced. I will return to mortality models in Chap. 7, where I will compare Gompertz to some discrete mortality tables that are more common and familiar to practicing actuaries.

2.5 Need-to-Know: Annuity Values

2.5.1 Life Only Immediate Annuity

Following standard methodology, which one can find in the referenced books by Promislow [8], Kaas et al. [5], Dickson et al. [3], Bowers et al. [2], or Booth et al. [1], I will use the following to denote the discounted actuarial present value of a life annuity.

$$a(x,r) = \int_0^\infty e^{-rt}\,\Pr[T_x \geq t]dt = \int_0^\infty e^{-rt}\,({}_tp_x)dt \tag{2.11}$$

Table 2.1 Pure life annuity factor values. Gompertz ($m = 90, b = 10$)

Age (x)	$r = 2\%$	$r = 4\%$
55	22.12615	16.82003
65	17.04378	13.73359
75	11.91615	10.17229

Under Gompertz mortality, this can be solved analytically, see, for example, Milevsky [6], and leads to the expression:

$$a(x, r) = \frac{b\Gamma(-rb, e^{(x-m)/b})}{\exp\{(m-x)r - e^{(x-m)/b}\}}, \tag{2.12}$$

where $\Gamma(A, B)$ is the incomplete Gamma function. For calibration and comparison purposes I now display values for life annuity prices assuming two different interest (valuation) rates: $r = 2\%$ and $r = 4\%$. Table 2.1 provides a range of numerical values of actuarial present values: $a = a(x, r)$, for $x = 55, x = 65, x = 75$ under valuation rates $r = 2\%$ and $r = 4\%$.

Here is how *actuarial present values* are converted into actual market or insurance company *prices*. For example, under an $r = 4\%$ interest rate and an insurance loading of 20%, a $P = \$100{,}000$ premium at the age of $x = 65$, would result in a lifetime cash-flow of $C = \frac{100000}{1.2(13.73359)} = \$5676$ per year, or (closer to continuous time) \$109.15 per week, which ceases at death. This is the last I'll say of *insurance loadings*, other than to remind readers that most of the algorithms and scripts in this book are presented within a fee-free zone. In practice, loadings, commissions and transaction costs must be incorporated and at the very least deducted from anticipated payouts. Likewise, the transition from actuarial factor $a(x, r)$ to payout rate $\kappa(x, r) := 1/a(x, r)$ might involve yet another layer of fees and charges that are for the most part ignored.

2.5.2 Refund at Death IA

Most consumers and annuity buyers are unwilling to "invest" in a life annuity in which all-is-lost at death, and many buyers are asking for some sort of death benefit or at least a refund of the un-earned premium. In the more advanced version of the *modern tontine* I will allow for and include such a feature, so in this section I will examine the mathematical properties of the corresponding annuity, the cash-refund immediate (or life) annuity. When you think about how the value of the basic annuity factor is defined in Eq. (2.11), the first thing to note is that its price must be defined recursively. Why? Because the periodic cash-flow itself depends on the original price. In the absence of any insurance loading, this can be expressed mathematically as follows:

$$a^{\star}(x,r) = \int_0^{\infty} e^{-rs}\,({}_s p_x) ds + \int_0^{a^{\star}(x,r)} \big(a^{\star}(x,r) \;-\; s\big)\, e^{-rs}\,({}_s p_x)\,\lambda_{(x+s)}\,ds,$$

where the right-hand side represents the actuarial present value of payments. The annuitant receives \$1 for life, but if they die early, that is before the entire $a^{\star}(x,r)$ has been returned, the beneficiary receives the difference between the price paid $a^{\star}(x,r)$ and payments received prior to death. In other words, death prior to age $y = x + a^{\star}(x,r)$ triggers a (declining) death benefit.

Notice how the first integral above is the basic income annuity factor $a(x,r)$, and the second integral represents the non-negative life insurance component. The cash-refund value minus the life only value can also be expressed as:

$$a^{\star}(x,r) - a(x,r) \;=\; \int_0^{a^{\star}(x,r)} \big(a^{\star}(x,r) \;-\; s\big)\, e^{-rs}\,({}_s p_x)\,\lambda_{(x+s)}\,ds \quad \geq 0. \tag{2.13}$$

Solving for the (implicit) variable that determines the value of the cash-refund annuity is best done via a bisection algorithm, or using the built-in solver in **R**. For more on this, please read the paper referenced as Milevsky and Salisbury [7]. For example, under a Gompertz law of mortality with $m = 90$, $b = 10$ and $r = 3\%$, the standard annuity factor at age $x = 65$ is $a(65, 0.03) = 15.25$ while the factor with a death benefit feature is $a^{\star}(65, 0.03) = 16.91$, both per dollar of lifetime income. See Table 2.2 for more.

Note, once again, how in contrast to Eq. (2.12), the $a^{\star}$ appears on both sides of Eq. (2.13) which leads to an implicit recursivity. The paper cited as Milevsky and Salisbury [7] offers a proof of existence and uniqueness and shows that $a^{\star}(x,r)$ actually declines in both x and r. In sum, this chapter contains a crash-course assortment of formulas, results and algorithms that will be used in later chapters to simulate scenarios and payouts for the *modern tontine*. For anything more I refer the reader to the references.

Table 2.2 A life annuity with a cash refund at death. Gompertz ($m = 90$, $b = 10$)

Age (x)	$r = 2\%$	$r = 4\%$
55	23.79569	17.47113
65	19.54472	14.90225
75	15.19471	11.97156

References

1. Booth, P., Chadburn, R., Haberman, S., James, D., Khorasanee, Z., Plumb, R. H., & Rickayzen, B. (2020). *Modern actuarial theory and practice*. Chapman & Hall, CRC Press.
2. Bowers, N. L., Gerber, H. U., Hickman, J. C., Jones, D. A., & Nesbitt, C. J. (1997). *Actuarial mathematics*. Schaumburg, IL: Society of Actuaries.
3. Dickson, D. C. M., Hardy, M. R., & Waters, H. R. (2009). *Actuarial mathematics for life contingent risks*. Cambridge: Cambridge University Press.
4. Gompertz, B. (1825). On the nature of the function expressive of the law of human mortality and on a new mode of determining the value of life contingencies. *Philosophical Transactions of the Royal Society of London, 115*, 513–583.
5. Kaas, R., Goovaerts, M., Dhaene, J., & Denuit, M. (2008). *Modern actuarial risk theory: Using R* (Vol. 128). Springer Science & Business Media.
6. Milevsky, M. A. (2020). *Retirement income recipes in R: From ruin probabilities to intelligent drawdown*. New York: Springer-Nature.
7. Milevsky, M. A., & Salisbury, T. S. (2022). Refundable income annuities: Feasibility of money-back guarantees. *Insurance: Mathematics and Economics, 105*, 175–193, ISSN 0167-6687. https://doi.org/10.1016/j.insmatheco.2022.03.004
8. Promislow, S. D. (2006). *Fundamentals of actuarial mathematics*. Toronto: John Wiley & Sons.

Building a Tontine Simulation in R

3

In this chapter I explain the core of the (basic, version 1.0) modern tontine simulation algorithm and provide R-scripts that can be used to generate forecasted values for what I have called the Modern Tontine (MoTo) Fund. This work will proceed in three separate sections. Section 3.2 will focus exclusively on simulating (projecting, forecasting) the life and death of participants in the modern tontine. Section 3.3 will move on to modelling portfolio investment returns, using a very simple LogNormal model. Then, Sect. 3.4 will combine mortality and returns from the two prior section to simulate (project, forecast) tontine dividends as well as modern tontine fund values. The basic models presented and assumptions made in this chapter will form the basis of more advanced models that will be developed and presented in further chapters. At times, I will distinguish between the *fund manager* who selects investments and oversees asset allocation for the underlying fund, and the *tontine sponsor* who is responsible for the dividend and payout policy. They involve different trade-offs, so it helps to keep them separate, even if managed by the same organization or individual.

3.1 On Rates, Yields and Returns

There are 3 key variables (numbers, percentages, rates) that might sound the same—and might be confused at first—but must be kept distinct in our MoTo simulation model. The **first** variable is the continuously compounded long-term *assumed* rate of return (ARR) earned by the MoTo, which I'll label r. This could be (as low as) $r = 2\%$ or (as high as) $r = 6\%$ or whatever and obviously depends on how the manager plans to invest the underlying assets of the fund. The ARR is critical to setting the *initial* tontine dividend payout and also for *forecasting* the tontine dividend payout made to surviving investors. The ARR value is the single most important system-wide parameter, other than perhaps the mortality assumptions. For

M. A. Milevsky, *How to Build a Modern Tontine*, Future of Business and Finance,
https://doi.org/10.1007/978-3-031-00928-0_3

most of the numerical examples reported, I will assume its $r = 4\%$. This r number is net, which means after all fees, investment charges and management expenses.

The **second** variable—or more precisely, vector—is the *realized* investment return (RIR) over time which I'll denote by $r[, j]$, where the j is an index for the year (or period) after the fund was launched. So, for example, the fund manager might have *expected* to earn the above-noted $r = 4\%$ in the first year of the MoTo, but the *realized* investment return could have been $\tilde{r}[, 1] = 2\%$, and in the second year $\tilde{r}[, 2] = 6\%$, and in the third year $\tilde{r}[, j] = 4\%$, etc. So, although the arithmetic (or path integral) average indeed was 4%, the path itself was rocky. I will be simulating many different (vector) paths for $\tilde{r}_i$, which I'll describe in Sect. 3.3. Future chapters will describe more advanced simulations.

The **third** and final rate variable relates to the annual cash-flow distribution to shareholders, which I'll call the tontine dividend rate (TDR) and denote it by κ_i. It denotes the percentage of the fund value that is distributed to all survivors at the end of period i. So, for example, if the MoTo is worth F_7 at the end of year 7, the tontine dividend shared by all survivors would be denoted by $\kappa_7 F_7$. And, the tontine dividend payout per survivor would be: $\frac{\kappa_7 F_7}{GL_7}$, where GL_7 over here represents the number of survivors alive at the very beginning of year 7. The TDR function κ could in theory be changed, updated and revised on an ongoing basis depending on realized investment returns, realized mortality and other (unplanned) business factors. Nevertheless, at time zero, tontine sponsor should have a good idea or sense of what κ_i will look like over time.

In fact, if the tontine sponsor wants to maintain and generate stable tontine dividend for survivors, then they must solve for the κ_i function endogenously—and not assume it exogenously. In terms of notation, the sequence $\kappa_1 = 7\%, \kappa_2 = 8\%, \kappa_3 = 9\%$, means they plan in advance to distribute 7% of the mark-to-market value of the fund at the end of the first year, 8% at the end of the second year, 9% in the third year, etc. To be clear in terms of timelines, though, dividends are declared at the start of the year and paid out at the end of the year. For example, the tontine dividend for the i'th year, $i \geq 1$ is paid out just before year $(i + 1)$. Of course, this entire model and set-up can be changed to quarters, months or even weeks.

Regardless of timing, I can't emphasize enough how important it is to understand that κ_i must be: (1.) solved for within the model, and (2.) be an increasing function of time, and (3.) must be greater than the assumed rate of return, that is $\kappa_i > r$. What I'm trying to say is that to maintain and manage the fund equitably for all initial investors, the sponsor can **not** simply distribute *realized* deaths plus *realized* returns and only give that (rather stingy amount) to surviving shareholders. That rule will create a last-man or (most likely) woman standing tontine, instead of the natural tontine with natural payouts in which cash-flows to unit holders are relatively constant over time. In other words, the modern tontine fund is designed to shrink over time by spending longevity (a.k.a. mortality) credits before they are actually earned.

3.2 Life & Death: Known Parameters

The Gompertz Law of mortality, with modal parameter m and dispersion parameter b is at the core of the tontine simulation model, as noted in Chap. 2. The simplest Gompertz-based model of life & death is the following script, which generates a matrix of `N=10,000` simulation paths, assuming `GL0=1000` initial lives, over a time horizon of `TH=30` years.

```
# Binomial Simulation of Gompertz
# Parameters are set.
x<-65; m<-90; b<-10; GL0<-1000; TH<-30; N<-10000
# Placeholders are created.
GLIVE<-matrix(nrow=N,ncol=TH)
GDEAD<-matrix(nrow=N,ncol=TH)
# Loop through N simulations.
for (i in 1:N){
  # Simulate deaths in year 1.
  GDEAD[i,1]<-rbinom(1,GL0,1-TPXG(x,1,m,b))
  # Subtract deaths from GL0 to get survivors .
  GLIVE[i,1]<-GL0-GDEAD[i,1]
  # Loop through remaining years.
  for (j in 2:TH){
    # Simulate deaths.
    GDEAD[i,j]<-rbinom(1,GLIVE[i,j-1],1-TPXG(x+j-1,1,m,b))
    # Count survivors.
    GLIVE[i,j]<-GLIVE[i,j-1]-GDEAD[i,j]
  }
}
```

Running the above-noted script creates two (rather large) matrices called `GDEAD` and `GLIVE`, both of which will be used (later on) to determine natural tontine dividends. Notice how the baseline age is set at $x = 65$, and the Gompertz parameters are assumed to be `m=90` and `b=10`, the initial size of the tontine pool is `GL0`, and the time horizon is `TH`. All of these parameters can (and should) be modified to help develop intuition for their effects. You may have noticed that I simulated and calculated the year one numbers separately. The reason I did this is because year one uses the number `GL0` as a parameter which does not appear in the `GLIVE` matrix, since the first column of `GLIVE` shows the number of survivors at the **end** of year one.

Note that this was no more than a trivial accounting assumption, to have a matrix begin with the end of the first period, but obviously has implications for the computation of tontine dividends and fund values. The first column you will see in any of the summary matrices should not be interpreted as the initiating value.

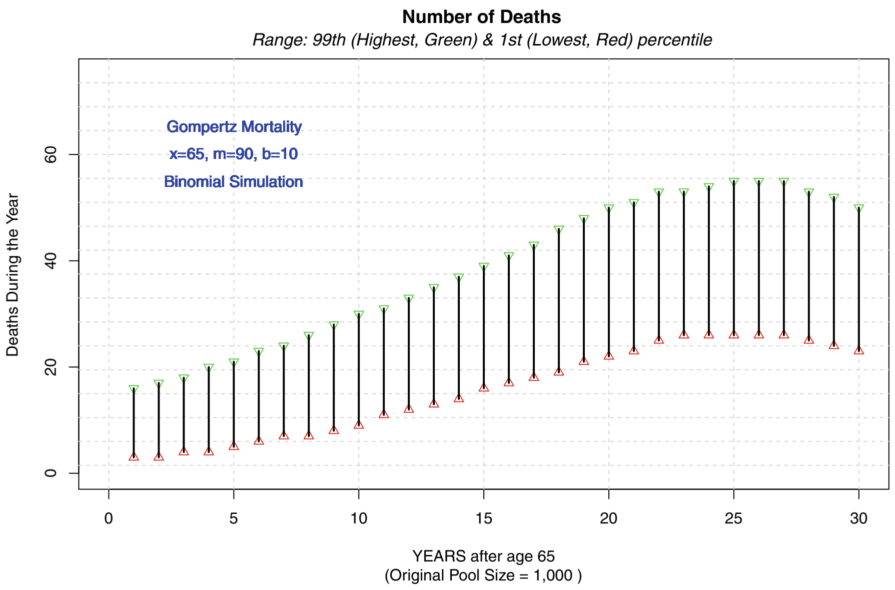
```
plot(c(0,30),c(0,1000),type="n",
     xlab="YEARS after age 65",
     ylab="Alive at Year-end")
title(main="Number of Survivors",
      sub="(Original Pool Size = 1,000 )")
```

```
mtext(side=3, line=0.3,
"Range: 99th (Highest, Green)  & 1st (Lowest, Red)percentile"
      ,cex=1.1,font=3)
grid(ny=18,lty=20)
for (i in 1:30){
  pct99<-as.numeric(quantile(GLIVE[,i],0.99))
  pct01<-as.numeric(quantile(GLIVE[,i],0.01))
  points(i,pct99,col="green",pch=6)
  points(i,pct01,col="red",pch=2)
  segments(i,pct01,i,pct99,lwd=2)
  text(5,400,"Gompertz Mortality",col="blue")
  text(5,350,"x=65, m=90, b=10",col="blue")
  text(5,300,"Binomial Simulation",col="blue")
}
```

Figure 3.1 displays a 98% confidence interval for the number of deaths within any given year and the number of survivors at the end of the year. The R-script which generated the survivor plot is listed here for convenience. I will not (needlessly) copy-and-paste this basic R-script over and over again every time I display a figure, but the syntax used to generate the equivalent figure for the number of deaths is similar with the `GLIVE[i,j]` replaced with `GDEAD[i,j]`, and the y-axis rescaled. Notice the key piece of R-code is the `quantile(.)` function which computes the 99th and 1st percentile and should also be changed for different confidence intervals.

The figure in the bottom panel echoes the `TPXG(x,t,m,b)` function, where the value of time is increased from zero to 30. If you look closely there is a noticeable spread between the 99th and 1st percentile, which captures the randomness in the number of survivors. Recall that the `TPXG(x,t,m,b)` function captures the *expected* number of survivors, when applied to the initial group size `GL0`. In contrast, the top panel of Fig. 3.1 offers a clearer view of the uncertainty in the number of survivors, by showing the uncertainty in the number of deaths each year.

The phrase *Binomial Simulation*, which appears in both the script and the figure, is meant to remind users and readers that deaths were generated by drawing from a Binomial random variable for the number of deaths, in which the probability of death is Gompertzian `(1-TPXG(x,1,m,b))`. So, while the *expected* number of deaths in any given year is the *realized* number of survivors at the end of the prior year, times the Gompertz mortality rate, the actual number will be simulated using `rbinom(.)`. Stated differently, and to explain this from another direction, if the initial size of the tontine pool was very (very) large, the bottom panel in Fig. 3.1 would perfectly match the Gompertz survival curve (all percentiles would be the same), and the top panel would resemble the probability density function (pdf) of the Gompertz distribution. Once again, there would be no upper and lower bound.



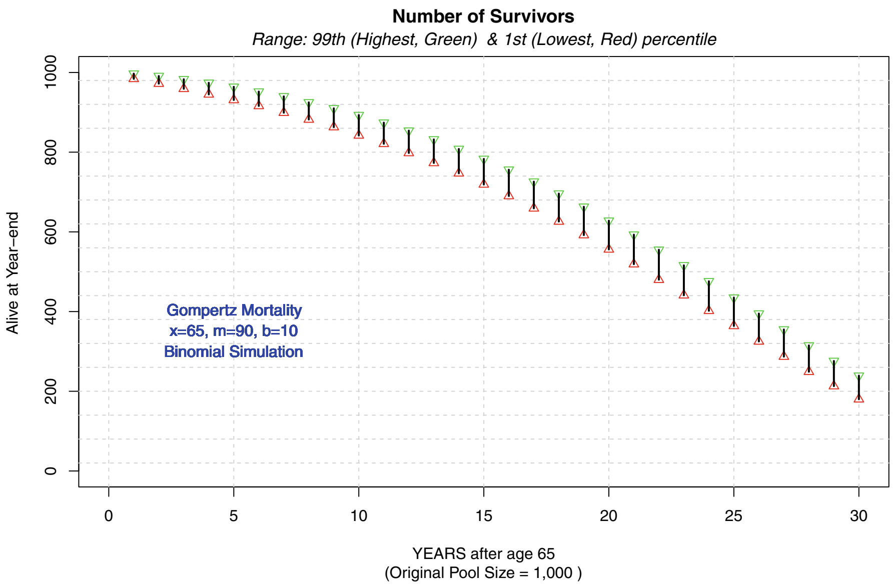


Fig. 3.1 Life & Death for fixed m, b, (with 98% C.I.)

3.2.1 Doubly Stochastic Death

In some situations the sponsor of the *modern tontine* fund might be unsure about the exact Gompertz parameters (m, b) to use for forecasts. This chapter and most of what follows takes the Gompertzian view calibrated to an actuarial mortality table. But, sponsors might want to account for health & gender of the initial investors. This then raises the technical problem of which precise values to use for b, and especially for m. While conservative assumptions as well as scenario analysis are the obvious solutions to this dilemma, I would recommend randomizing the values of the two parameters to get a crude sense of the range of outcomes. In this subsection I will demonstrate how to simulate (forecast, project) future lives & deaths while also randomizing the one or both the values of (m, b). I call this a *doubly stochastic* simulation because the parameters themselves are randomized as well as the number of deaths. Although this shouldn't be done haphazardly—mostly because of mortality compensation relationships—a simple script that would perform that double-simulation could look something like this:

```
# Doubly Stochastic Gompertz
# Set parameters and define placeholders.
x<-65; Em<-90; b<-10; GL0<-1000; TH<-30; N<-10000
GLIVE<-matrix(nrow=N,ncol=TH)
GDEAD<-matrix(nrow=N,ncol=TH)
# Loop through N simulations.
for (i in 1:N){
  # Generate a random m value.
  m<-runif(1,Em-5,Em+5)
  # Use that m value to simulate year 1 deaths.
  GDEAD[i,1]<-rbinom(1,GL0,1-TPXG(x,1,m,b))
  # Count year 1 survivors.
  GLIVE[i,1]<-GL0-GDEAD[i,1]
  # Loop through remaining years.
  for (j in 2:TH){
    # Generate a random m value.
    m<-runif(1,Em-5,Em+5)
    # Use that m value to simulate deaths in that year.
    GDEAD[i,j]<-rbinom(1,GLIVE[i,j-1],1-TPXG(x+j-1,1,m,b))
    # Count the survivors.
    GLIVE[i,j]<-GLIVE[i,j-1]-GDEAD[i,j]
  }
}
```

If you look closely at the script and compare with the earlier one labelled the *Binomial* simulation, the important difference is the extra line `m<-runif(1,Em-5,Em+5)`. Notice the fixed expected modal value, denoted by `Em=90`, but the simulated actual (to be used) value of m using a uniform distribution in a ten year range. In other words, the m that I used to generate the *random* number of death is itself simulated with a *random* value of m that is uniformly distributed between $(85, 95)$. This is doubly stochastic, and perhaps even excessively random. Why excessive? Well, it's unlikely that over a 30 year horizon the underlying value of m will fluctuate from year-to-year. If you believe that the Gompertz assumption itself is problematic, it won't be fixed by randomizing m or b. This might sound cryptic, and I will return to more sophisticated models of mortality and longevity in Chap. 7.

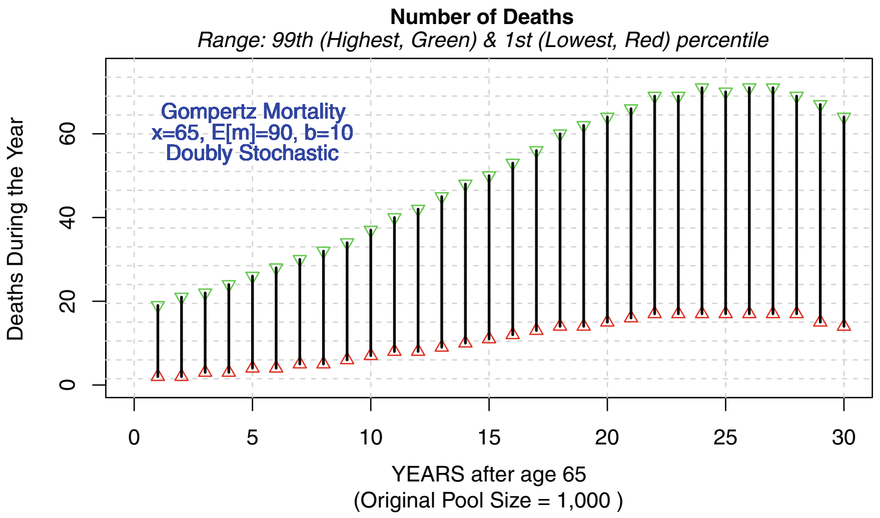
Figure 3.2 should be compared with Fig. 3.1. They both display the projected number of people that will die (top panel), from an initial group of `GL0=1000`, as well as the projected number of survivors (bottom panel), assuming a Gompertz law of mortality. But Fig. 3.2 contains that extra noted amount of randomness.

The key qualitative takeaway is rather obvious, namely that it's harder to predict—and there are wider bands for—the number of deaths in any given year, although the midpoint (or average) is the same in both Figs. 3.2 and 3.1. I should emphasize (again) the rather simplistic nature of my *doubly stochastic* simulation, in that I have assumed independence of (random) m values from year to year, and no uncertainty in b, as well as no noticeable or meaningful trend in either m or b over time. All of this will (eventually) be re-examined and questioned to see if-and-how a richer model of life & death might change economic results. Remember, the objective of this simulation exercise isn't to generate demographic or population forecasts. On those two dimensions, Gompertz would fail. Rather, the point is to see if the tontine dividend payouts and the risk to the tontine sponsor changes in any significant way if, for example, mortality is lower/higher in the first few year, or if mortality plateaus at some advanced age, etc.

3.3 Investment Returns

As I noted in the introductory remarks, there are three critical numbers that all sound like rates (yields, returns) that are *baked* into a modern tontine fund, and one must be very careful not to confuse them with each other. In this subsection I will explain how to simulate (simple) random values for the realized investment return, which I labelled r_i.

Intuitively, the assumed rate of return used to construct and engineer the modern tontine, r, is the average of the expected realized returns, denoted by r_i. So, for starters, I will assume that returns are *LogNormally* distributed, that is their logarithm is *Normally* distributed with a mean value `EXR` and standard deviation `SDR`. So, while I will keep r distinct from `EXR` for expositional purposes, they really should be the same. In the following script, the mean is assumed to be 4% and the standard deviation is 3%. Future chapters, and in particular #6 will revisit this rather stale assumption and take our investments into the (non-parametric) twenty-first century.



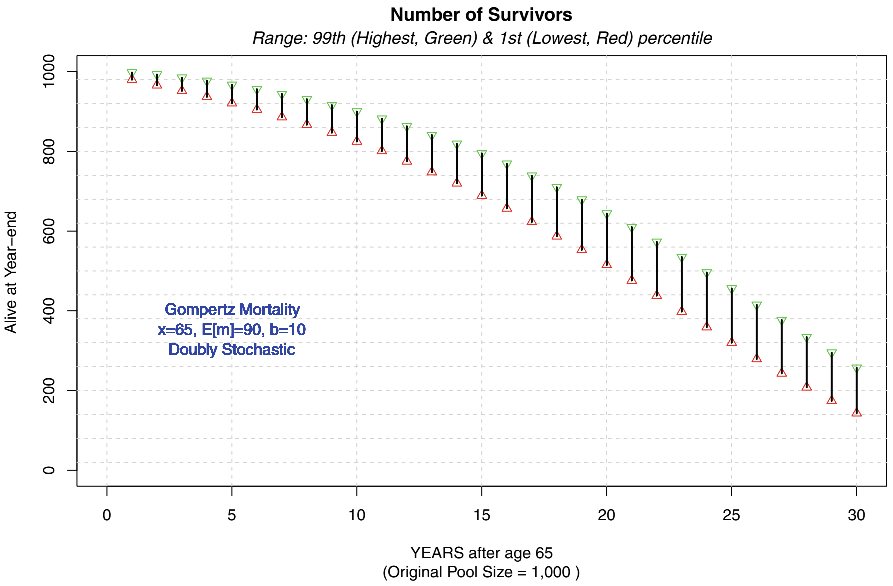


Fig. 3.2 Life & Death for variable m, b, (with 98% C.I.)

```
# Simulated LogNormal Returns
# Set base parameters.
EXR<-0.04; SDR<-0.03; TH<-30; N<-10000
# Define placeholders.
PORET<-matrix(nrow=N,ncol=TH)
STPRV<-matrix(nrow=N,ncol=TH)
# Simulate N paths of TH returns.
for (i in 1:N){
  PORET[i,]<-exp(rnorm(TH,EXR,SDR))-1
  }
# Calculate stochastic present value.
for (i in 1:N){
  for (j in 1:TH){
    STPRV[i,j]<-prod(PORET[i,1:j]+1)^(-1)
  }
}
```

Notice how the script generates a new matrix `PORET` as well as a sister matrix `STPRV`, which are abbreviations for portfolio returns and stochastic present values. Without any loss of generality, I could have assumed `SDR=0` and the above script would represent a (very cumbersome and long) way of setting every entry in the `PORET` matrix equal to exactly $e^{0.04} - 1$, which is an effective annual rate of: 4.0811%. The `STPRV` value would be $(1.04811)^{-j}$, which is obviously the present value of a dollar to be received at the end of year j. Once again, more sophisticated and advanced models for `PORET` will be described in Chap. 6.

I remind readers that it's best to think of and express the `EXR` number as the expected continuously compounded investment return, which was very carefully explained in Chap. 2, and is often described as the *geometric mean* investment return, which is smaller than the *arithmetic mean* investment return. Indeed, when `SDR=0`, both means exactly the same. But, as the value of `SDR` increases—a number that is often denoted by the Greek letter σ—the *geometric* and *arithmetic* mean will diverge from each other. You can confirm this to yourself by comparing `mean(PORET)+1`, which is closer to the *arithmetic*, versus `exp(mean(log(PORET+1)))` which is *geometric*.

Figure 3.3 displays the output of one particular simulation run in which (as mentioned) the expected CC-IR is: `EXR=0.04`, the standard deviation (or σ) is `SDR=0.03`, the number of periods is $TH = 30$ years, and the total number of scenarios is $N = 10{,}000$. The top panel of Fig. 3.3 is rather boring and uninformative. Every year the 98% confidence interval ranges from close to negative 3% to over 11%. Losses are indeed possible and the fund value could decline (absent payouts), but it's unlikely. After all, the number zero is over 1.3 standard deviations away from 4%, and in a normal distribution 90% of outcomes would fall above zero. And, although Fig. 3.3 is LogNormal, the differences are minor. Finally, the bottom panel of Fig. 3.3 should be interpreted as the present value curve with the dispersion representing the variation of year-by-year returns. The critical role of `PORET` in driving the fund values over time should be obvious.

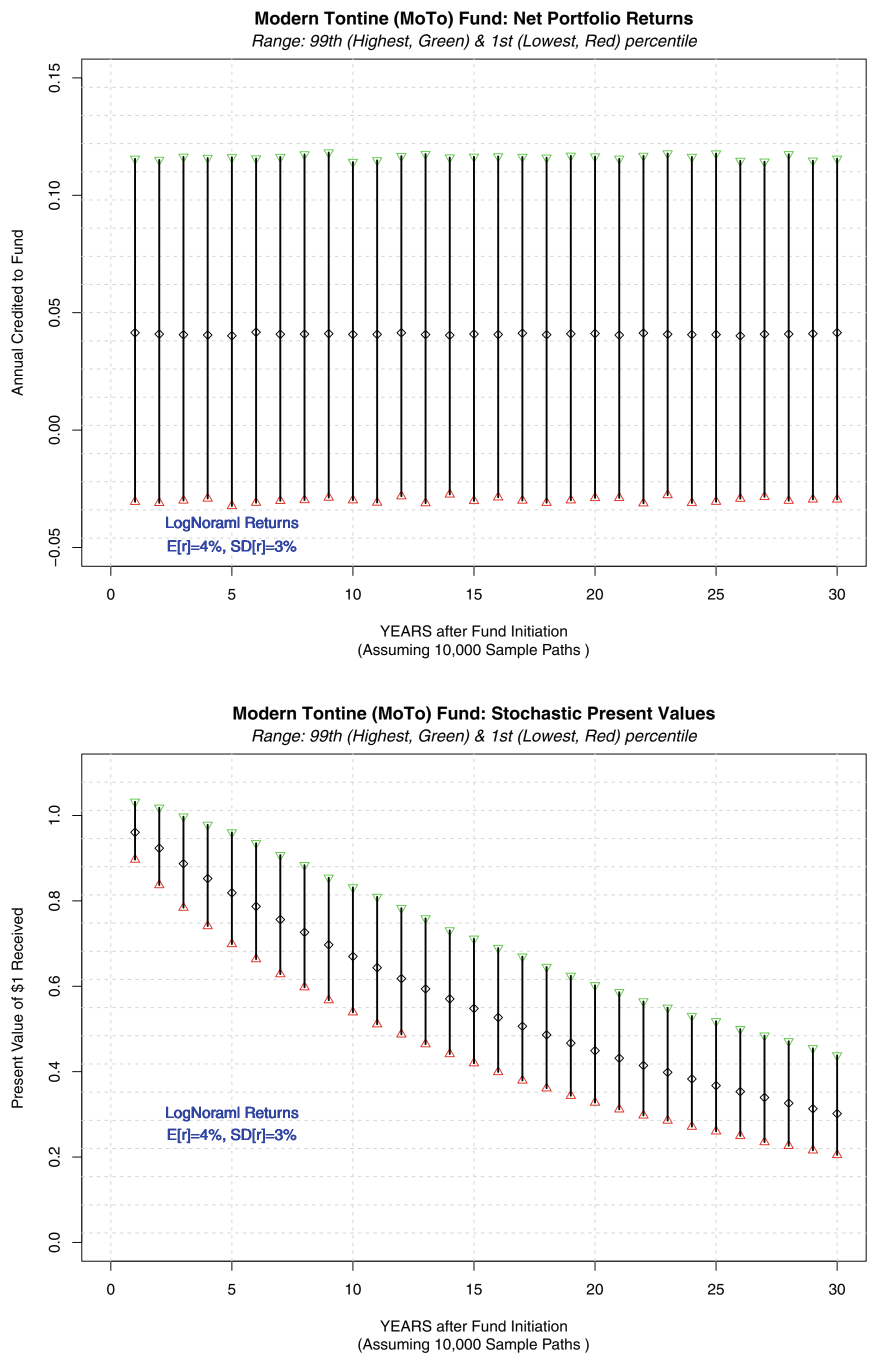


Fig. 3.3 Investment returns and stochastic present value (98% C.I.)

3.4 Dividend & Fund Values

3.4.1 Temporary Life Income Annuities

Before I explain how to combine the GLIVE investor survivor matrix and the PORET investment return matrix to compute natural tontine dividends and fund values, I will (re)define the value of a *temporary* life income annuity in discrete time as follows:

```
# Temporary Life Income Annuity
TLIA<-function(x,y,r,m,b){
  APV<-function(t){exp(-r*t)*TPXG(x,t,m,b)}
  sum(APV(1:(y-x)))
}
```

Note how I use the sum function to add-up the actuarial present value APV, which is the product of the Gompertz survival rates and constant interest discount rates. Both of those (i.e. mortality and interest) will be modified in later chapters. Moreover, the summation begins at one year (or period)—because payments begin at the end of the period/year—and conclude at time (y-x), which is the end of the temporary period. Needless to say, that can be at the very end of the mortality table and lifespan. So, the TLIA function is just a generalized and standard expression for valuing any-and-all income annuities.

```
TLIA(65,105,0.04,90,10)
> 13.23439
TLIA(65,100,0.04,90,10)
> 13.20022
TLIA(65,95,0.04,90,10)
> 13.03634
```

The value (or zero commission cost) of \$1 income annuity purchased at age $x = 65$ would range from \$13.04 to \$13.23, depending on whether payments terminate at age $y = 95$, $y = 100$ or $y = 105$. The TLIA function and its reciprocal yield value will be used to determine the natural dividend payout rate, which early on here I called κ_i. In particular, the (reciprocal) yield will be used and referenced in the R-script.

```
1/TLIA(65,105,0.04,90,10)
> 0.07556072
1/TLIA(65,100,0.04,90,10)
> 0.07575632
1/TLIA(65,95,0.04,90,10)
> 0.07670865
```

So, for example, a 65-year-old who buys a temporary income annuity to age 95, would be entitled to a yield of 7.67%. An initial premium or investment of \$100,000 would entitle the annuitant to \$7,671 yearly, until age $y = 95$, at which point all payments would terminate. This number TLIA and rate κ will make many appearances in the next few pages. Notice that extending the temporary age to $y = 105$, or even $y = 110$ won't make much of a difference to (i.e. and won't reduce) the yield, which is why the simulations (in the basic core model) will terminate after TH=30 years.

3.4.2 A Perfect Fund Value Over Time

We now arrive at the core section (of this core chapter) which explains how to simulate modern tontine fund values, using the random returns and random lifetimes generated in the prior sections. The process is recursive in the sense that the algorithm begins with an initial modern tontine fund value DETFV at time zero, equal to the initial investment f0 (reported in thousands of dollars) multiplied by the initial number of live investors GL0. The time-zero fund value: f0*GL0 is increased by investment returns during the year, and then reduced by tontine dividend payouts TONDV to survivors at the end of the year. That then becomes the new tontine fund value, and the process continues again for the second year, etc.

Given the centrality of this particular script to the core simulation of the tontine, in the next few pages I will provide three distinct (but similar) versions of the R-script for generating the DETFV as well as the TONDV matrix. The first script doesn't really represent a true tontine fund but is better described as an *annuity* fund in which dividend payouts are rigidly fixed at some pre-determined value kappa*f0. The payout is maintained at that fixed dollar level for all TH=30 years, but the annual investment returns credit to the fund are fixed at r and the deaths are deterministic according to the Gompertz survival curve. In other words, there is absolutely no uncertainty in this R-script. Nothing is random. All is deterministic.

I urge readers and users to review this R-script line-by-line very carefully to ensure a proper and careful understanding of how the updating and revision mechanism is reflected in DETFV[i,j] as well TONDV[i,j]. Critically, notice how the value of κ is fixed once at the very beginning of the simulations, which means that the TONDV matrix is identical at all times and in all scenarios. I am resetting its value (thousands of times, again and again) to the same value. The reason for that (waste in computational time) is that it allows me to make only minor changes to the script later on, when I get to the proper Modern Tontine (MoTo) simulation.

```
# Perfect Annuity Fund: Fixed Returns & Deterministic Deaths
# Parameters are set.
x<-65; r<-0.04; m<-90; b<-10; TH<-30; N<-10000;
GL0<-1000; f0<-100;
# Placeholders are defined.
DETFV<-matrix(nrow=N,ncol=TH)
TONDV<-matrix(nrow=N,ncol=TH)
# Define single kappa value.
kappa<-1/TLIA(x,x+TH,r,m,b)
for (i in 1:N){
  # Dividend and fund value at end of year 1.
  TONDV[i,1]<-kappa*f0
  DETFV[i,1]<-f0*GL0*exp(r)-TONDV[i,1]*GL0*TPXG(x,1,m,b)
  for (j in 2:TH){
    # Dividend and fund value at end of year j.
    TONDV[i,j]<-kappa*f0
    # The next two lines should be combined as one.
    DETFV[i,j]<-DETFV[i,j-1]*exp(r)
    -TONDV[i,j]*GL0*TPXG(x,j,m,b)
  }
}
```

I have denoted this artificial situation the perfect annuity fund case, because it leads to constant fixed (annuity) dividends of `kappa*f0` every single year, and the fund value will terminate or mature after `TH=30` years at exactly a value of zero. Basically, I have replicated a temporary life income annuity which can be confirmed with the following numerical values of the fund (or reserves supporting the annuity) at the end of various years.

```
> round(summary(DETFV[,30])/1000,3)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
      0       0       0       0       0       0
> round(summary(DETFV[,29])/1000,3)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  1.538   1.538   1.538   1.538   1.538   1.538
> round(summary(DETFV[,28])/1000,3)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  3.278   3.278   3.278   3.278   3.278   3.278
> round(summary(DETFV[,27])/1000,3)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  5.224   5.224   5.224   5.224   5.224   5.224
```

As one moves backward in time, from the terminal `TH=30` to the prior year 29, 28, etc, the fund value will continue to reverse-shrink (grow) and eventually lead to the original \$100 million fund value at time zero, which is the \$100,000 investment times the 1000 investors. Note also that the simulation is conducted in units of thousands, and I have divided the value of `DETFV[i,j]` in the above script by a further 1000 so that the reported numbers are in millions. I have also rounded the numbers to include only three significant digits. And, since this is most definitely not a simulation, you (the user) should get the exact same results I do!

The perfect annuity fund will be worth zero million dollars in the final year, 1.54 million dollars in the 29th year, 3.28 in the 28th year, etc. Figure 3.4 displays those values graphically, which should further confirm the intuition of the perfect annuity fund. For the record, this isn't a tontine. This is an annuity. In the next subsection I will randomize deaths and randomize returns, in other words I'll import the `PORET` and `GLIVE` matrix, and this perfect picture will disappear.

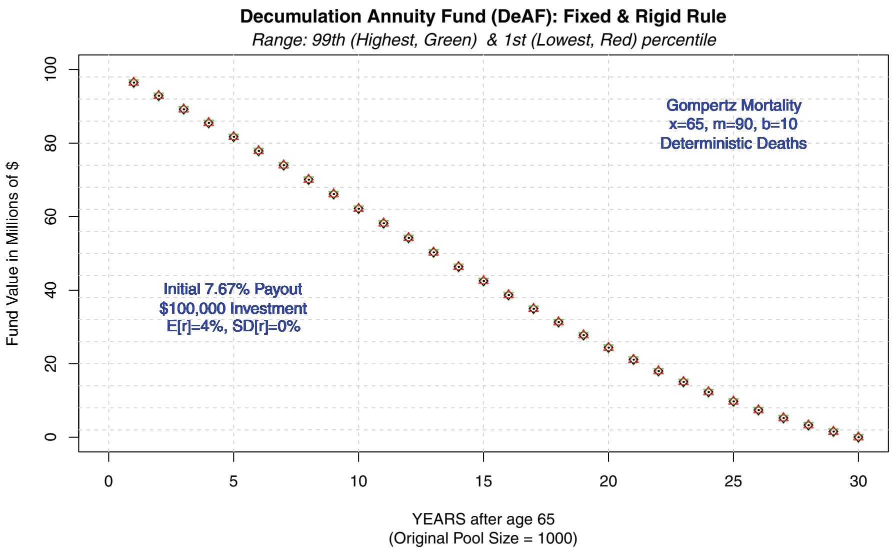


Fig. 3.4 Perfect replication of a temporary life income annuity

3.4.3 Fixed Rules in a Variable World

In the next bit of R-script, which is the second of three simulation scripts for the fund itself, I have made two small changes to the algorithm but which have a significant impact on results. Please review the script carefully—compare with the prior perfect annuity fund case—and you will see that the (random) investment return `(1+PORET)` matrix has now replaced the constant `exp(r)` value, and the actual (random) number of survivors `GLIVE` has replaced the estimate based on `TPXG`.

This introduces two sources of randomness—and we finally have a proper simulation—but notice that dividend payout rule reflected in `TONDV` is still very much fixed and rigid. This still isn't a modern or natural tontine. Rather, it's a very reckless way to manage the liabilities generated by selling temporary income annuities. You will understand why it's reckless once you see the results.

Before running this script make sure to run the *Binomial Simulation of Gompertz* script (included below) so that you are using the regular `GLIVE` matrix and not the Doubly Stochastic one. You should also already have the `PORET` matrix loaded from running the *Simulated LogNormal Returns* script.

```
# Redefine GLIVE.
x<-65; m<-90; b<-10; GL0<-1000; TH<-30; N<-10000;
GLIVE<-matrix(nrow=N,ncol=TH);GDEAD<-matrix(nrow=N,ncol=TH)
for (i in 1:N){
  GDEAD[i,1]<-rbinom(1,GL0,1-TPXG(x,1,m,b))
  GLIVE[i,1]<-GL0-GDEAD[i,1]
  for (j in 2:TH){
    GDEAD[i,j]<-rbinom(1,GLIVE[i,j-1],1-TPXG(x+j-1,1, m,b))
    GLIVE[i,j]<-GLIVE[i,j-1]-GDEAD[i,j]}}
```

```
# Annuity Replication: Stochastic Returns and Deaths
# Parameters are set.
x<-65; r<-0.04; m<-90; b<-10; TH<-30; N<-10000;
GL0<-1000; f0<-100;
# Placeholders are defined.
DETFV<-matrix(nrow=N,ncol=TH)
TONDV<-matrix(nrow=N,ncol=TH)
# Define single kappa value.
kappa<-1/TLIA(x,x+TH,r,m,b)
for (i in 1:N){
  # Dividend and fund value at end of year 1.
  TONDV[i,1]<-kappa*f0
  DETFV[i,1]<-f0*GL0*(1+PORET[i,1])-TONDV[i,1] *GLIVE[i,1]
  for (j in 2:TH){
    # Dividend and fund value at end of year j.
    TONDV[i,j]<-kappa*f0
    # The next two lines should be combined.
    DETFV[i,j]<-DETFV[i,j-1]*(1+PORET[i,j])
    -TONDV[i,j]*GLIVE[i,j]
  }
}
```

Every year the sponsor will extract or withdraw or redeem exactly `kappa*f0` (in thousands of dollars) times the number of survivors, regardless of the actual investment performance or the actual number of survivors. The fund value could (theoretically) collapse and everyone could still be alive, and the payout rule is blindly followed. That is a recipe for disaster and the reported numbers reflect the problems with such a strategy. Notice the large probability this fund will hit zero (get ruined) and fall into a deficit situation. Reviewing the numbers in the following script the median fund value is negative in the 30th year, and there is at least a 25% chance the fund has a deficit of \$16.8 million. In year 25 the 1st quartile is negative, and in year 15 the worst case scenario has a deficit of \$0.36 million.

```
> summary(DETFV[,30])/1000
    Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
-82.2616 -16.8075  -0.8042   0.8374  16.4262 124.6543
> summary(DETFV[,25])/1000
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
-60.123  -4.029   9.064  10.430  23.182 117.246
> summary(DETFV[,20])/1000
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 -27.79   13.25   23.86   24.90   35.39  105.06
> summary(DETFV[,15])/1000
    Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
 -0.3617  33.4796  42.0511  42.9000  51.4952 104.1820
```

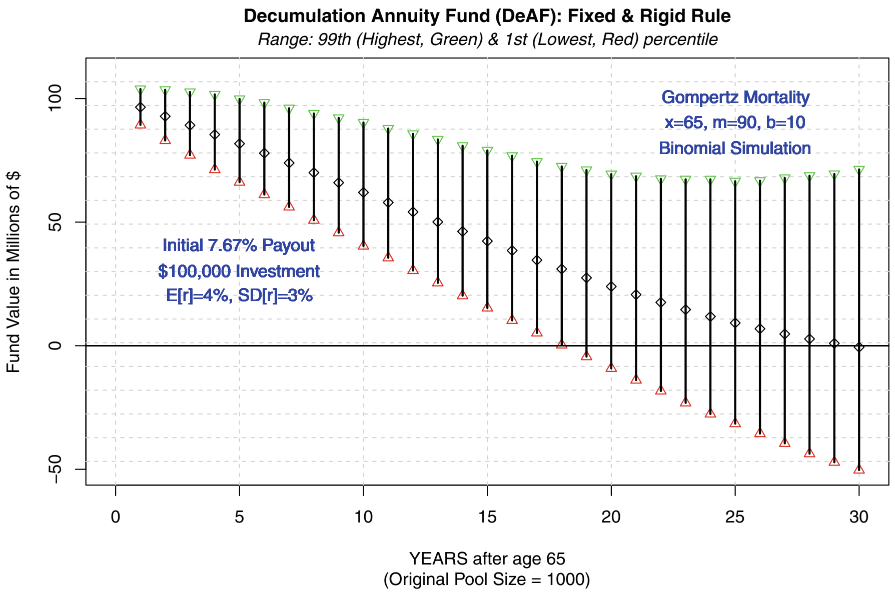


Fig. 3.5 Fund values for a rigid and poorly designed Tontine

Figure 3.5 tells the exact same (scary) story, and by year 19 there is a non-trivial (>1%) chance the fund gets ruined. The reason for all this negativity lies in the rule driving the `TONDV` calculation. Again, the `TONDV` is rigid, fixed and doesn't adjust to the fluctuating nature of investment returns and realized mortality. That's not a tontine. Think back to the bottom corner of the triangle at the beginning of Chap. 2.

In some sense, this is the essence of *tontine thinking*. An optimized fund adjusts the yearly payout so that in times when the fund is showing a loss—relative to the above mentioned perfect value—the sponsor reduces tontine dividend. In years when the fund is doing better than expected, the dividends are increased. On average, the tontine dividends will be similar to the fixed and rigid payout rule, but the ability to modulate payouts (like a thermostat) will keep the fund solvent. A *natural* tontine dividend rule goes beyond making `TONDV` flexible or adapted to returns; it makes them equitable for all participants as well as safe for the sponsor.

3.4.4 The Natural Tontine Rule

This brings me to the third and final (for this chapter) fund simulation script, which adjusts the tontine dividend `TONDV` by the realized investment returns reflected within the `PORET` matrix as well as the realized survivors as reflected in the `GLIVE` matrix. At its essence, the algorithm adjusts the next period's annuity income the survivors would be able to afford given the current level of assets, by dividing into the relevant annuity factor `TLIA` at that age.

This 15-line script is the essence (or secret sauce) of the Modern Tontine (MoTo) fund, and the dividends are thus *natural*. It tries to maintain a constant tontine dividend profile, but adjusts up and down depending on circumstances. In the language of Actuarial Science 101, annuity payouts are adjusted to fit the updated annuity reserves. That's an explanation of the *natural tontine*, while standing on one-foot and the rest is implementation. Of course, there is a chance this fund hits zero (at least theoretically) in which case the dividends will be terminated. In practice though, the structure of the *thermostat* is such that the tontine dividends would have been reduced long before the fund itself reaches such distressful levels. If you want, you can actually count the number of values at or below zero in either DETFV or TONDV matrix, but you likely won't find any.

```
# The Natural Tontine: Stochastic Returns and Deaths
# Parameters are set.
x<-65; r<-0.04; m<-90; b<-10; TH<-30; N<-10000;
GL0<-1000; f0<-100
# Placeholders are defined.
DETFV<-matrix(nrow=N,ncol=TH)
TONDV<-matrix(nrow=N,ncol=TH)
# Vector of kappa values.
kappa<-c()
for (i in 1:TH){kappa[i]<-1/TLIA(x+i-1,x+TH,r,m,b)}
for (i in 1:N){
  # Dividend and fund value at end of year 1.
  TONDV[i,1]<-kappa[1]*f0
  DETFV[i,1]<-f0*GL0*(1+PORET[i,1])-TONDV[i,1] *GLIVE[i,1]
  for (j in 2:TH){
    # Dividend and fund value at end of year j.
    TONDV[i,j]<-kappa[j]*DETFV[i,j-1]/GLIVE[i,j-1]
    # The next two lines should be combined as one.
    DETFV[i,j]<-max(DETFV[i,j-1]*(1+PORET[i,j])
    -TONDV[i,j]*GLIVE[i,j],0)
    }
  }
```

Although the above script is based on the N=10000 scenarios I would urge (new) readers to gain further intuition and start by setting N=1 and manually trace thru the DETFV and TONDV vector year by year, so you understand how the fund value changes in response to investment returns PORAT[1,] and deaths GDEAD[1,]. Using the full N=10000 results generated, let us examine the tontine fund values towards the end of the tontine's life. Once again, the results are reported in thousands, so I have divided the summary(.) by 1000 and the numbers are in $ millions.

```
> summary(DETFV[,30])/1000
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
0.00000 0.00000 0.00000 0.02609 0.04192 0.29884
> summary(DETFV[,25])/1000
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  4.682   8.640   9.703   9.830  10.866  18.949
```

```
> summary(DETFV[,20])/1000
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  13.53   21.96   24.32   24.53   26.81   43.10
```

Notice how the final DETFV[,30] value of the fund is nearly a perfect zero, and as one goes backwards in time (from age 95 to age 65), the fund value grows. In the 25th year of the fund, its value is between \$4.68 million and \$18.9 million. And, in its 20th year (when survivors are 85 years old), the tontine fund value is worth between \$13.5 million and \$43.1 million. These numbers and the general pattern are reflected in Fig. 3.6. Without exaggerating, these last 2-3 pages are the core of all future simulations, models and extensions.

Make sure you understand how the kappa[] vector is used to determine dividends, and how it might result in higher/lower values compared to kappa[1]. The tontine dividends themselves (in thousands of dollars) are not fixed and can fluctuate from year to year. Rather, what is determined in advance is the κ_i rate used to compute those tontine dividends. It's the reciprocal of the income annuity factor applicable at the relevant age $(x + i)$. So, if the fund *randomly happens* to be worth \$50 million in year $i = 13$, and there *randomly happens* to be 780 survivors at age 78, then in that scenario next year's natural tontine dividend would be set at $(50{,}000{,}000/780)/8.161 = \$7{,}855$. The number 8.161 is the value of TLIA(78,95,0.04,90,10). It represents the annuity 780 survivors could afford at age 78, if they annuitize their fund value.

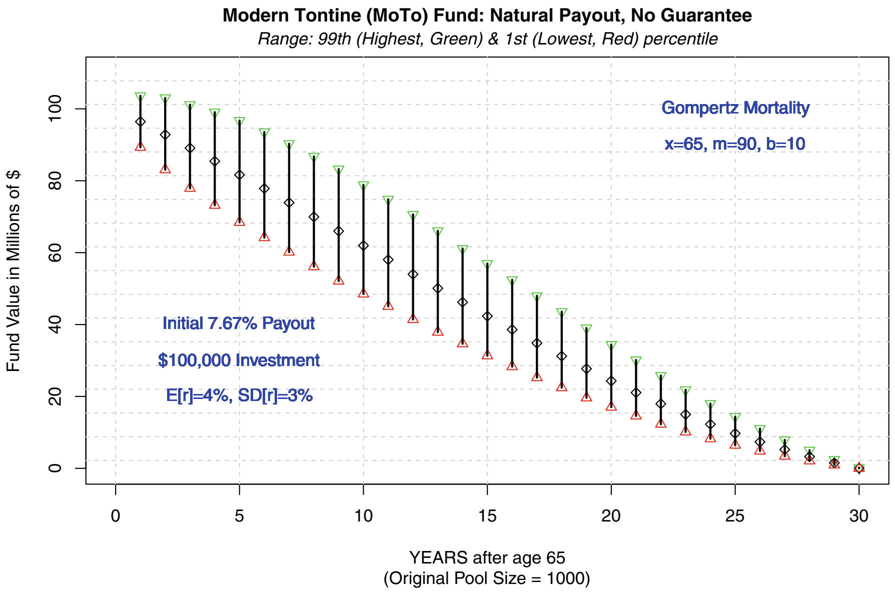


Fig. 3.6 Modern Tontine (MoTo) fund values, with confidence intervals

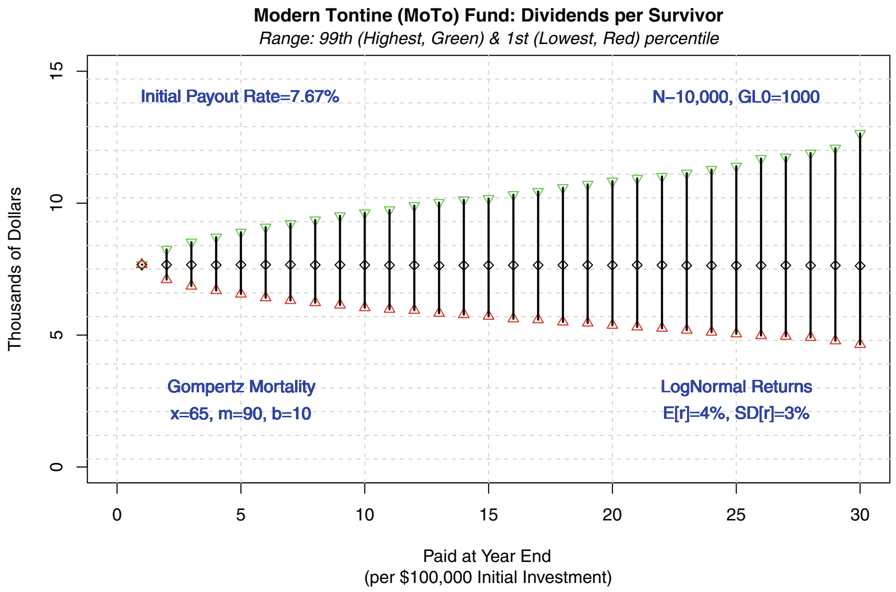


Fig. 3.7 Confidence intervals for the Modern Tontine (MoTo) dividends

Let's focus on those dividends carefully and in more detail. Figure 3.7 displays the usual 98% confidence interval, with the 1% "worst" case and the 99% "best" case, assuming all the parameter values and structure noted earlier. Notice how the cone of uncertainty in the tontine dividend increases over time, although the expected midpoint is stable around the initially determined $7, 671 per year. At the very end of the `TH=30` time horizon the range is quite wide but is comfortably north of $4, 500 even in the worst case scenario.

The following set of values are the standard simulation summaries for the tontine dividends in the 1st, 10th, 20th, 25th and 30th year. Notice how the mean value remains in the range of $7, 600 – $7, 700 until year 20 (or age 85), after which it increases just slightly. This trend reflects a built-in skew in the underlying return mechanism, which tilts towards higher tontine dividends over time. In fact, one could legitimately claim that the 3rd quartile payments (read: the good scenarios) trend upwards over time. Although one must also be careful to note the extreme volatility in payment that might be observed once the number of survivors (from the initial 1000) reaches the teens and single digits.

```
> summary(TONDV[,1])
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  7.671   7.671   7.671   7.671   7.671   7.671
> summary(TONDV[,10])
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  5.213   7.159   7.656   7.686   8.168  11.051
> summary(TONDV[,20])
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  4.216   6.918   7.647   7.723   8.440  14.022
```

```
> summary(TONDV[,25])
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
  3.656   6.803   7.639   7.745   8.578  15.473 
> summary(TONDV[,30])
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
  3.190   6.578   7.625   7.791   8.784  17.654 
```

3.4.5 The Cumulative Payout

One final dataset I would like to define is the cumulative tontine dividends received by survivors: TCPAY. It's reported in thousands of dollars and grows (i.e. never declines) over time, as new tontine dividends are paid out. This matrix helps us answer the question: *Do investors get their money back?* The construction is rather trivial, using cumsum() and the next paragraph offers a simple numerical example.

```
TCPAY<-matrix(nrow=N,ncol=TH)
for (i in 1:N){
TCPAY[i,]<-cumsum(TONDV[i,])}
>   summary(TCPAY[,30])
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
  153.8   214.6   230.0   231.4   246.6   353.5 
```

If you happen to survive for the entire 30 years (from age 65 to 95) the median cumulative payout in this particular simulation run would have been \$230 thousand dollars. Obviously, if-and-when you generate your own simulation you will get (slightly) different values, but they shouldn't differ too much from mine. Here's how to interpret. If you survived for the entire horizon, your original investment of \$100,000 would have generated a total (median) cash-flow of \$230,000, with 3rd quartile of \$246, 600. This sum is almost two and a half times what you invested at age 65, although it shouldn't be interpreted as a rate of return since it's not adjusted for the time value of money, but it could be using the STPRV matrix generated earlier in the chapter.

Finally, Fig. 3.8 displays the TCPAY data graphically, and in particular the point in time at which the original \$100,000 is returned to the tontine investors. This takes place sometime between year 12 and year 15. At this point the TCPAY number is just an interesting curiosity or side-product of the calculation. In a later chapter I will return to this number (and concept) when I discuss death benefits in Chap. 5.

3.5 Conclusion and What's Next

At this point in the (basic) tontine simulation there should be a total of seven distinct data matrices in your R-studio environment. They are: GLIVE and GDEAD, which are the number of surviving investors at the *end* of the year and deaths *during* the year, assuming the Gompertz law with parameters (m, b), both of which could also

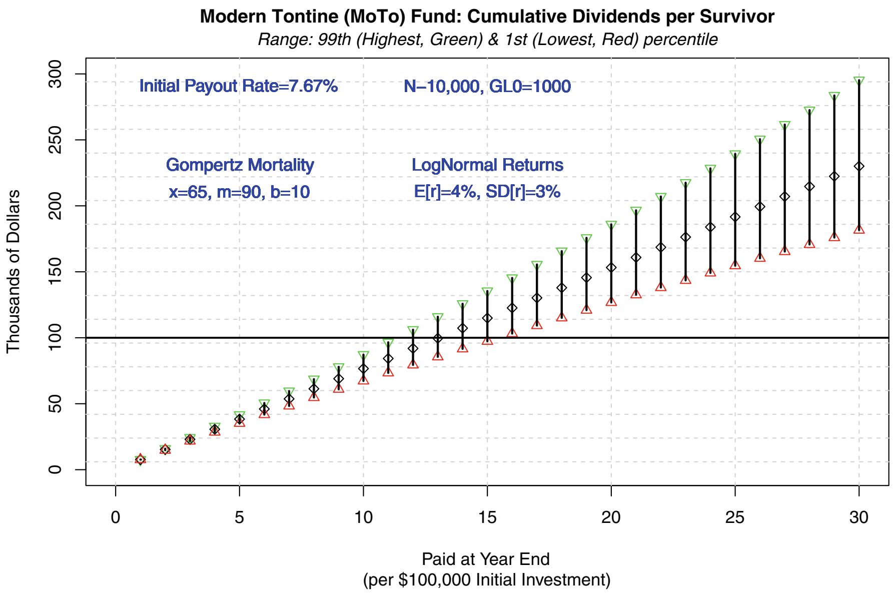


Fig. 3.8 Cumulative payments made to survivors

be randomized. The matrices PORET and STPRV capture the randomized annual investment returns, and the present value of a \$1 payment assuming the individual path of returns. Finally, the matrices TONDV, TCPAY and DETFV combine the random investment returns and the random deaths to compute the tontine dividends, total cumulative payments and the actual tontine fund value, at the end of each of the *TH* years, all for each of the N simulated scenarios. If you are keeping track of space and memory, with $N = 10,000$ scenario paths, and $TH = 30$ years, there are a total of 300,000 data points per matrix.

3.6 Test Yourself

1. Investigate the impact of changing the Gompertz (m, b) assumptions, in the baseline 4% case. In other words, assume the $m = 95$ (in all scenarios) or that $m = 85$ (in all scenarios) and discuss the qualitative difference in the tontine dividends TONDV and the modern tontine fund DETFV values.
2. Instead of the 4% assumption for both r and EXR, please generate simulation results assuming a more conservative 3% return, with a 1% standard deviation, and a more aggressive 5% return, with a 4% standard deviation. In particular, focus on the TONDV matrix and create a summary table of the range of tontine dividend payout in years 10, 20 and 30, with a 50% confidence interval. In other words, compute the first and third quartile at those dates. Explain and comment on the results, and remember that EXR=r.

3. Going back to the $r = 4\%$ case, focus on the `TCPAY` matrix and plot the distribution of the time at which investors get their money back. Remember, there are 10,000 scenarios embedded within `TCPAY`, and the objective of this question is to get a sense of the times (and ages) at which investors are "made whole" assuming they are still alive.
4. Focus on the `GLIVE` matrix and the `TCPAY` matrix and please use those two datasets to compute the number of original investors `GL0`, who when they died, did not get their full money back. In other words, the sum of the tontine dividends they received until their death didn't exceed their original investment. Remember that people who die in year i aren't entitled to any tontine dividends in that year. Once you have figured this out, compute the number of investors who got less than 80% back, less than 60% back and less than 40%, by the time they died.
5. Modify the basic simulation so dividends are paid quarterly, and generate results assuming the same Gompertz: $m = 90$, $b = 10$, and $r = 4\%$ case. Be very careful when you modify the code (and increase the size of all the matrices) that your returns and payouts are properly adjusted. For example, the 3-month survival probability and investment return is obviously quite different from the 12-month values. Once you are done, confirm the `TCPAY` values after 10, 20 and 30 years are virtually the same.

4 Statistical Risk Management

In this chapter I dig deeper into the results from the (very basic) simulation presented in the prior chapter so users can develop a more sophisticated sense of the drivers of the outcomes. Along the way I will introduce summary metrics that can quickly and easily represent the thousands of possible data points using key *dashboard metrics* for the dividends and fund values. I will also examine the mortality assumptions and dig deeper into some *what if* questions and broader risk management issues.

4.1 Stable Tontine Dividends: Defined

To get us started, I have re-generated the `GLIVE` and `GDEAD` matrix, but this time starting with `GL0=5000` investors (instead of the 1000 used in the prior chapter). For now, I have maintained the same age $x = 65$, and Gompertz parameters ($m = 90$, $b = 10$), although I will modify those as we progress thru the chapter. For now, I have also assumed the same $r = 4\%$, and $\sigma = 3\%$, which means that as far as the R-script is concerned `EXR=0.04` and `SDR=0.03`. As noted in Chap. 3, this leaves a 10,000 by 30 matrix `TONDV` and `DETFV`, representing the dividends and fund value.

My first objective in this chapter is to extract some summary metrics from these large and extensive simulation results (besides pretty pictures), which can give users a quick sense of the *stability* of the projected dividends and fund, but in a manner that is completely objective and statistically sound. Every time the user modifies any of the investment, mortality or tontine dividend payout rule parameters, the entire `TONDV` matrix will change, and I would like to create some summary metrics. First up is the mean value, standard deviation and relative variability of the tontine dividends:

M. A. Milevsky, *How to Build a Modern Tontine*, Future of Business and Finance,
https://doi.org/10.1007/978-3-031-00928-0_4

```
mean(TONDV)
> 7.710858
sd(TONDV)
> 1.002761
sd(TONDV)/mean(TONDV)
> 0.1300454
```

The average dividend is \$7.71 thousand dollars per year, per initial \$100,000 investment, which is very close to the $\kappa_1 = 7.67\%$. The standard deviation divided by the mean, or the tontines dividend variability is approximately 13%, which is across 10,000 scenarios and 30 years. While this number doesn't account for differences in time or the time value of money, it is a baseline of sorts, one that I will return to later.

The next issue I would like to examine is the existence of any possible trend of the tontine dividends over time: which is something we don't want, and shouldn't happen if the payout rules are being followed. I define a new vector `mtd` as the median tontine dividend every year, from year $i = 1$ to the final $i = 30$. The value `mtd[j]` is assigned the 50th percentile value of the `TONDV` matrix, to remind users that the median is the 50% mark. I could also use the built-in command `median` as well, and will do so interchangeably. While on the topic of choices, I could have also used the *mean* value or perhaps even the *modal* value of the $N = 10{,}000$ scenario dividends in each of the 30 years as the metric on which I focus on for stability; they are all debatable.

```
mtd<-c(); t<-1:TH
for (j in 1:TH){
  mtd[j]<-as.numeric(quantile(TONDV[,j],0.50))
}
fit<-lm(mtd~t)
```

The last line in the above script generates a simple linear regression using the `lm()` command, in which time is the *independent* variable and the median tontine dividend is the *dependent* variable. This is obviously not the only way to examine the pattern of (median) dividends over time, and an alternative is to fit some sort of time series model, test for unit roots, etc. The key though is to test whether or not there is any trend in the median dividends, which is reflected in the coefficient term.

The results are generated and displayed using the following command in **R**, which is quite good as far as I'm concerned. Normally, when generating regression results, one wants to see statistical significance of results—but in this case I don't. In fact, to declare that tontine dividends are stable over time, one should see a regression slope that is statistically indistinguishable from zero. And, that is precisely the result in this case. Of course, these are my (averaged, crude) simulation results and your coefficients will vary from mine, but they should be very close to zero. Indeed, the tontine payout formula was designed this way. It's one of the many summary statistics and dashboard metrics that will alert us to potential problems and serve as an early warning system for possible modelling errors.

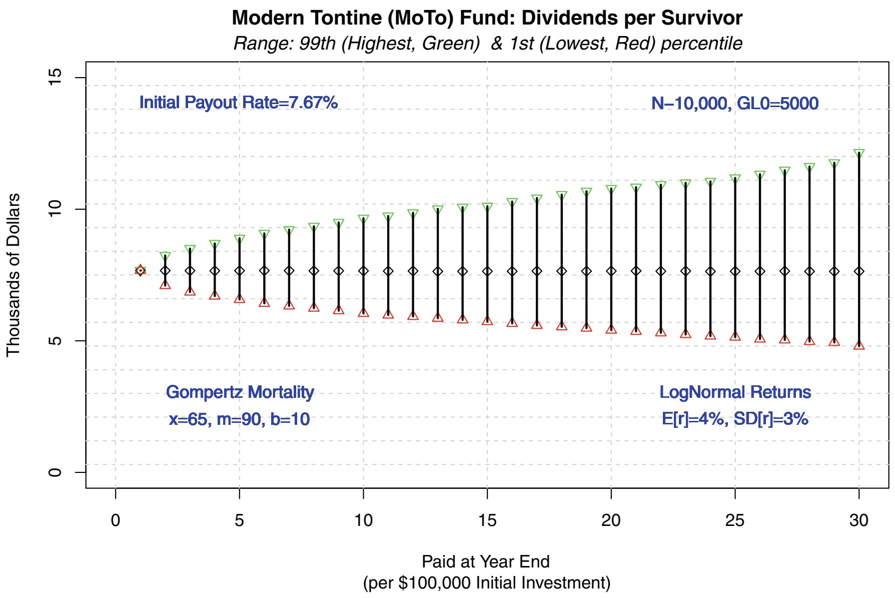


Fig. 4.1 Tontine dividends with low risk portfolio (98% C.I.)

```
summary(fit)

Call:
lm(formula = mtd ~ t)

Residuals:
       Min          1Q      Median          3Q         Max
-0.0167151 -0.0046365  0.0000555  0.0036623  0.0166466

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) 7.669e+00  2.953e-03 2596.97   <2e-16 ***
t           8.656e-05  1.663e-04    0.52    0.607
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.007885 on 28 degrees of freedom
Multiple R-squared:  0.009578, Adjusted R-squared:  -0.02579
F-statistic: 0.2708 on 1 and 28 DF,  p-value: 0.6069
```

So, although the variability in the tontine dividends do increase and grow over time as evidenced by Fig. 4.1, the median tontine payout is rather stable. I will return to this basic *tontine dividend stability* test and use it to detect (unwanted) trends in payouts over time as well as to examine and summarize worst-case scenarios.

For example, if I generate similar regression results but with a dependent variable (i.e. on the left) that is defined as the natural logarithm of the lowest percentile dividend, `log(quantile(TONDV[,j],0.01))` for example, the resulting coefficient on the regressor is approximately negative 1% and is statistically significant at the highest levels of confidence. In English: The worst case scenario gets worse over time. But you already knew that from Fig. 4.1. Later on in this chapter I will examine a few of the (10,000) individual sample paths of the tontine dividends, versus the confidence intervals, to develop yet another angle and perspective on their behaviour over time.

4.2 Riskier Portfolios and Wider Bands

Along the path to a deeper understanding of the simulation, R-script and results, the next parameter I would like to focus on is the standard deviation of returns, occasionally denoted by σ, or `SDR` in the R-script. How does it affect the `TONDV` matrix? Figures 4.2 and 4.3 offer a perspective on the answer.

Note the only difference is the variability in the portfolio investment return driving `PORET`. The `GLIVE` and `GDEAD` matrix are exactly the same, and I simply generated `TONDV` values using the different portfolio returns. Figure 4.2 has $\sigma = 0.1\%$ and Fig. 4.3 assumed $\sigma = 4\%$. Although the median path for both figures are

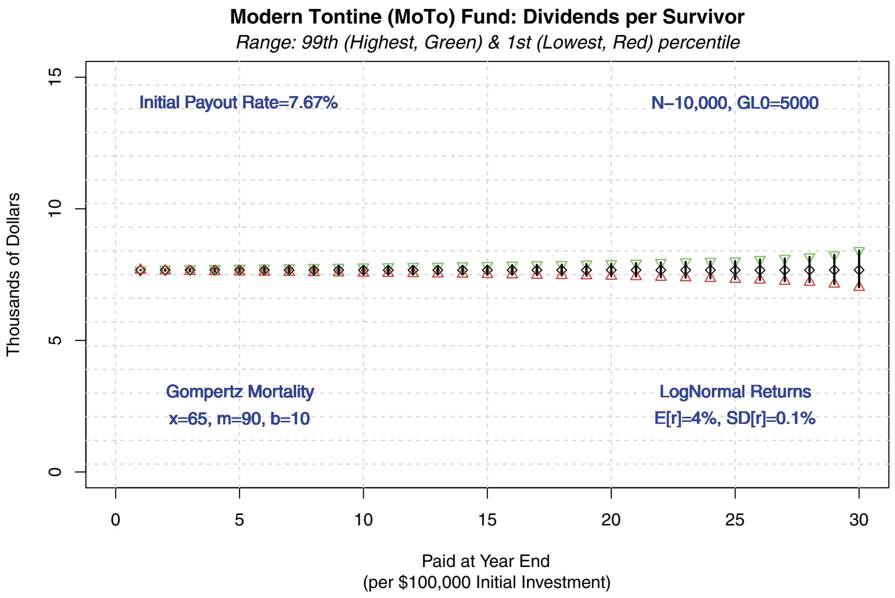


Fig. 4.2 Tontine dividends with close-to-zero risk portfolio (98% C.I.)

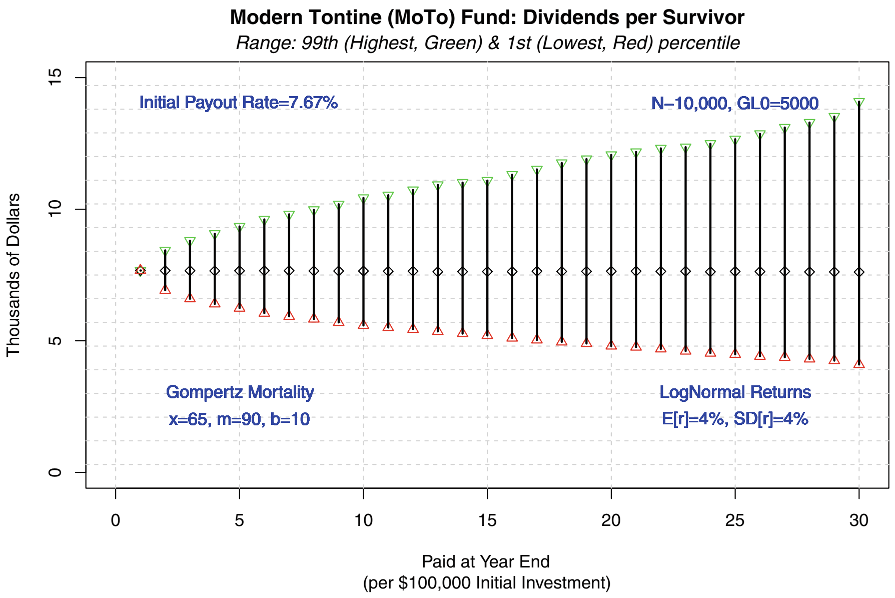


Fig. 4.3 Tontine dividends with medium risk portfolio (98% C.I.)

constant, and both pass the *stability* regression coefficient test I explained earlier, the 10,000 scenarios reflected in Fig. 4.3 are much riskier than those underlying Fig. 4.2. Readers and users can confirm that the ratio: `sd(TONDV)/mean(TONDV)` is a mere 1.5% when $\sigma = 0.1\%$, but jumps to 17% when $\sigma = 4\%$. This tontine dividend dispersion ratio is computed numerically, but in fact can also be derived via analytic methods in continuous time assuming Gompertz mortality and LogNormal returns, but taking us beyond the mandate of this book.

Nevertheless, the wide bands for the 98% confidence interval are not driven by the shrinking and small pool of survivors, although there is a trace of that in Fig. 4.2. Rather, the over $10,000 range in annual tontine dividends (from $4 to $14 thousand) for investors who survive to their mid-90s is entirely driven by investment returns. This means two things. First, we might want to think more carefully about how to reduce and control σ, to reduce the variability of payouts. Second, and just as importantly, the tontine sponsor must ensure they model future investment return volatility properly, or the tontine dividends themselves might fluctuate more than anticipated. I'll return to both of these matters later on in Chap. 6 when I discuss alternative investment model-assumptions and derivative-based strategies for controlling volatility. For now, let's examine a related question.

4.3 What's Worse: Mortality or Markets?

For the sake of completeness, I will stitch the various pieces of R-script I displayed and used in the prior chapter and place them in the next page as one (long) sequence of commands. The code begins with the Gompertz survival curve and the temporary life annuity function, then proceeds to define the system parameters and the various matrices I have already discussed at length, it then populates those matrices. The very last line performs my quick-and-dirty regression test for the stability of the tontine dividends over time. Users can simply cut-and-paste the code (which I am calling version 1.0), run it in R and generate a full set of results. In later chapters, when I augment the basic code, I will use different version numbers and will focus on the parts of the R-script that have to be changed.

I now return to the basic core simulation results in which the initial size of the pool is `GL0=1000` investors, the assumed return is $r = 4\%$, and the simulated investment returns are: `EXR=0.04` with `SDR=0.03`, with the usual age and Gompertz parameters `x=65` with `m=90` and `b=10`. I generated `N=10000` scenarios in which an initial investment `f0=100` thousand dollars is tracked over a time horizon of `TH=30` years. And, in this particular "run" the expected tontine dividend over the 30 years and 10,000 scenarios is: `mean(TONDV)=7.73` thousand dollars per year, with a variability of `sd(TONDV)/mean(TONDV)=0.134`, or approximately 13%. The average tontine fund value (at year end) over those 30 years and 10,000 scenarios is: `mean(DETFV)/1000=43.47` million dollars.

```
# Modern Tontine (MoTo) Fund Version 1.0
# No Death Benefits Paid or Reserved
TPXG<-function(x,t,m,b){exp(exp((x-m)/b)*(1-exp(t/b)))}
TLIA<-function(x,y,r,m,b){
  APV<-function(t){exp(-r*t)*exp(exp((x-m)/b)*(1-exp(t/b)))}
  sum(APV(1:(y-x)))}
x<-65; m<-90; b<-10; GL0<-1000; TH<-30; N<-10000
EXR<-0.04; SDR<-0.03; r<-0.04; f0<-100
kappa<-c()
GLIVE<-matrix(nrow=N,ncol=TH)
GDEAD<-matrix(nrow=N,ncol=TH)
TCPAY<-matrix(nrow=N,ncol=TH)
PORET<-matrix(nrow=N,ncol=TH)
DETFV<-matrix(nrow=N,ncol=TH)
TONDV<-matrix(nrow=N,ncol=TH)
STPRV<-matrix(nrow=N,ncol=TH)
for (i in 1:N){
  GDEAD[i,1]<-rbinom(1,GL0,1-TPXG(x,1,m,b))
  GLIVE[i,1]<-GL0-GDEAD[i,1]
  for (j in 2:TH){
    GDEAD[i,j]<-rbinom(1,GLIVE[i,j-1],1-TPXG(x+j-1,1,m,b))
    GLIVE[i,j]<-GLIVE[i,j-1]-GDEAD[i,j]
  }}
for (i in 1:N){
  PORET[i,]<-exp(rnorm(TH,EXR,SDR))-1}
for (j in 1:TH){
   for (i in 1:N){
     STPRV[i,j]<-prod(PORET[i,1:j]+1)^(-1)
   }}
for (i in 1:TH){kappa[i]<-1/TLIA(x+i-1,x+TH,r,m,b)}
for (i in 1:N){
   TONDV[i,1]<-kappa[1]*f0
   DETFV[i,1]<-f0*GL0*(1+PORET[i,1])-TONDV[i,1]*GLIVE[i,1]
    for (j in 2:TH){
     TONDV[i,j]<-kappa[j]*DETFV[i,j-1]/GLIVE[i,j-1]
DETFV[i,j]<-DETFV[i,j-1]*(1+PORET[i,j])-TONDV[i,j]*GLIVE[i,j]}}
for (i in 1:N){TCPAY[i,]<-cumsum(TONDV[i,])}
mtd<-c(); t<-1:TH
for (j in 1:TH){mtd[j]<-median(TONDV[,j])}
fit<-lm(mtd~t)
summary(fit)
```

What I would like to do next is (cherry) pick a few choice years and carefully examine the distribution of tontine dividends in those years, and the various factors that are correlated (or not) with those dividends. In particular, I will define three new vectors, `td5<-TONDV[,5]`, as well as `td15<-TONDV[,15]`, and `td25<-TONDV[,25]`, which is obviously the entire array (and range) of payouts to survivors in the 5th, the 15th and the 25th year of the *modern tontine* fund. I get my simulation results, their mean values are as follows (and your numbers should be very close to):

```
mean(td5)
> 7.686171
mean(td15)
> 7.722712
mean(td25)
> 7.771013
```

First things first, let's examine (and interpret) the correlation of these three vectors (of three numbers). I will create a matrix via `cbind()` and then use the `cor()` command to generate a 3x3 matrix of correlation values. I could obviously have done this for all 30 years and urge users to do that at their leisure, but the story this matrix tells is the same.

```
dv<-cbind(td5,td15,td25)
> cor(dv)
           td5      td15      td25
td5  1.0000000 0.5210515 0.3752384
td15 0.5210515 1.0000000 0.7206721
td25 0.3752384 0.7206721 1.0000000
```

Notice how the numbers are all positive and decay over time but are rather high. Here is how to interpret them. The tontine dividend survivors receive at the age of 90 (which is 25 years after they invested at age 65) assuming they are alive, exhibits a correlation of 38% with the tontine dividend received two decades prior at the age of 70. If the dividend was above average in year 5, it will be above average in year 25. There is a clear persistence in the dividend. The converse is disappointing. If the year-5 tontine dividend didn't reach expectations then there is a good chance that 20 years later the tontine dividends will be under-average as well. If the first few years or periods are disappointing the rest are likely to disappoint as well.

Some readers will recognize this as the *sequence of returns* effect which is a well-used phrase in the retirement income arena. It should be intuitive. The reason tontine dividends are likely to be under-average in the first few years is because investment returns were under-average in those years. Yes, it could be that "too many" investors survived and that's why the tontine dividends were low, but in all likelihood the investment returns dominated. This is yet another reason to implement *volatility controls* in the first few years of the fund, perhaps using put and call options. I'll get back to this later.

4.4 Excess Life

Now, in the spirit of stress-testing and enterprise risk management let's *assume* a hypothetical under which deaths were *expected* to occur based on Gompertz mortality with $(m = 90, b = 10)$, but the *reality* was different. Using those Gompertz parameters the initial tontine dividend was set at $\kappa_1 = 7.67\%$ at age $x = 65$, using a valuation rate of $r = 4\%$. However, assume that in fact *nobody actually died* during the first 10 years of the fund. All 1000 investors survived to age 75. What happens to the fund and tontine dynamics in that case? Intuitively, the lack of deaths implies that individual tontine dividends will be forced downwards during the first decade (and perhaps beyond) because of the larger-than-expected number of survivors. Eventually though, as investors do begin to die at a normal rate (if such a thing exists) the tontine dividends will stabilize with the thermostat effect, but that will take some time. To get a more precise sense of how this will all affect the *modern tontine* fund, I will now generate a revised `GLIVE` and `GDEAD` matrix, one in which the first 10 columns (and years) are rather boring. Namely, the `GLIVE` values are all set equal to `GL0` and the `GDEAD` values are set to zero. This can be easily and quickly done using the following modified script. Nothing else is changed *anywhere* in the master script.

```
# Simulating Modified Gompertz Values
# Realizing No Deaths in First 10 Years
x<-65; m<-90; b<-10; GL0<-1000
for (i in 1:N){
  GDEAD[i,1:10]<-0
  GLIVE[i,1:10]<-GL0
  for (j in 11:TH){
    GDEAD[i,j]<-rbinom(1,GLIVE[i,j-1],1-TPXG(x+j-1,1,m,b))
    GLIVE[i,j]<-GLIVE[i,j-1]-GDEAD[i,j]
  }}
```

Once you have generated both matrices, be sure to use `summary(GLIVE[,1:10])` to confirm that during the first decade under all $N = 10{,}000$ scenarios there are no deaths. Then, in year #11, dying begins based on (Binomially simulated) Gompertz mortality, a.k.a. q_{75}, for a 75-year-old. I should note that this (simple and unrealistic) modelling assumption also means that nature is **not** catching-up and killing more investors after age 75, to account for the missing deaths in the first decade. Rather, the investors who were supposed to die in the first decade *escape* death altogether, as if the tontine fund started at age 75 with 1000 investors. In reality, if indeed none of the `GL0=1000` investors died during the first decade, I would anticipate seeing a slightly elevated mortality rate (relative to the Gompertzian value) at age 75 and beyond. But I leave this rather advanced discussion of more realistic models of life and death to a later stage, and at this point focus exclusively on the behaviour of tontine dividends during and after that peculiar decade.

Figure 4.4 displays the usual 98% confidence intervals, with the obvious flat line for the first decade and then a sudden (again, unnatural) jump in deaths starting in year #11. Notice how after the first decade the distribution of deaths resembles the earlier figures, which is exactly what one would expect under Gompertz mortality.

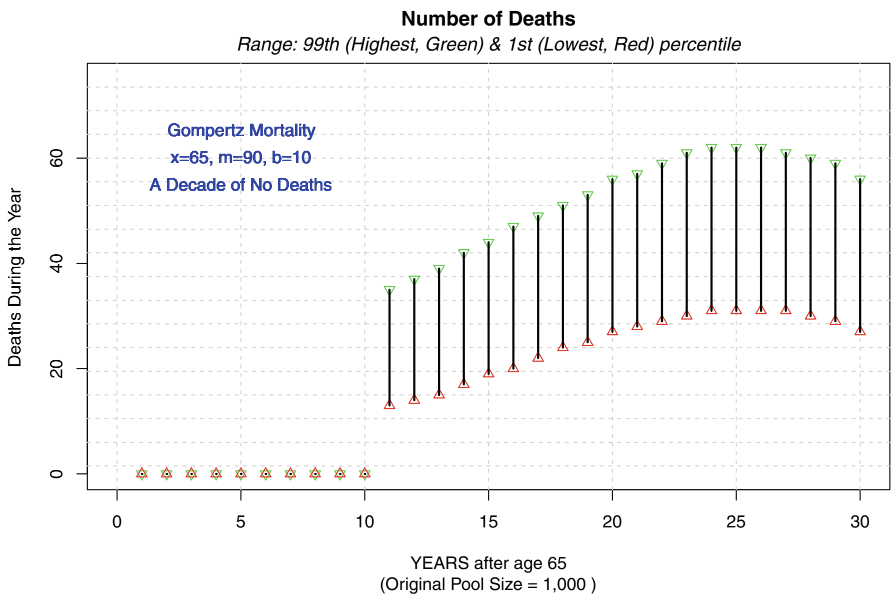


Fig. 4.4 Deaths delayed for decade and then Gompertz resumes

More importantly, moving on to the actual tontine dividends we can now use the latter part of the master script after the PORET segment and matrix should remain unchanged. Why? Recall that the stochastic model generating market portfolio returns is *independent* of the stochastic model generating life and death. Bottom line, after running the entire script the following script provides a statistical summary of the tontine dividends—using the revised GLIVE and GDEAD matrix—in the first few years of the *modern tontine* fund's life. Can you spot the pattern?

```
> summary(TONDV[,1])
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  7.671   7.671   7.671   7.671   7.671   7.671
> summary(TONDV[,3])
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  6.383   7.290   7.517   7.527   7.753   8.937
> summary(TONDV[,5])
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  5.752   7.032   7.342   7.358   7.659   9.491
> summary(TONDV[,7])
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  5.291   6.755   7.125   7.148   7.518   9.870
> summary(TONDV[,9])
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  4.822   6.447   6.868   6.900   7.311   9.694
```

Indeed, let's go thru the first 10 years of the life of the *modern tontine* fund in this (rather bizarre) universe during which nobody dies. At the end of the first year the tontine dividend is \$7.67 thousand per survivor, a number that was set and fixed at time zero. But, with all those extra survivors (that weren't expected) the tontine fund value at the end of year #1 will be lower than originally anticipated, leading to a pre-programmed and necessary reduction in the tontine dividend for year #2. In fact, this process continues during the entire decade and the tontine dividends must constantly be revised downwards albeit based on a well-defined and transparent algorithm. Notice how the median tontine dividend in year #3 is reduced to \$7.52 thousand, and then \$7.34 thousand and then \$7.12 thousand and then \$6.87 thousand. The 1st and 3rd quartile of the entire distribution of tontine dividends exhibits the same shrinking pattern, which is the thermostat effect. But then, starting in year #10 the tontine dividends stabilize as people (finally) start dying at the (originally) anticipated Gompertzian rate. Figure 4.5 displays that stabilization process.

What are the qualitative implications? Well, clearly the originally promoted—first year advertised dividend—payout rate of 7.67% cannot be maintained over time. It's obviously not sustainable if the promoters believed in (and priced based on) full Gompertz, but in fact there were no deaths during the first decade. At the end of every year the thermostat mechanism forces the sponsor to reduce the payouts as realized mortality credits full under expected mortality credits. And yet, despite the "error" or mistake in pricing assumptions the tontine dividends themselves will stabilize at approximately \$6.55 thousand dollars per year over time. In other words, once mortality begins to behave "normally" the *modern tontine* will settle-down, although to levels that were lower than originally anticipated under full Gompertz mortality.

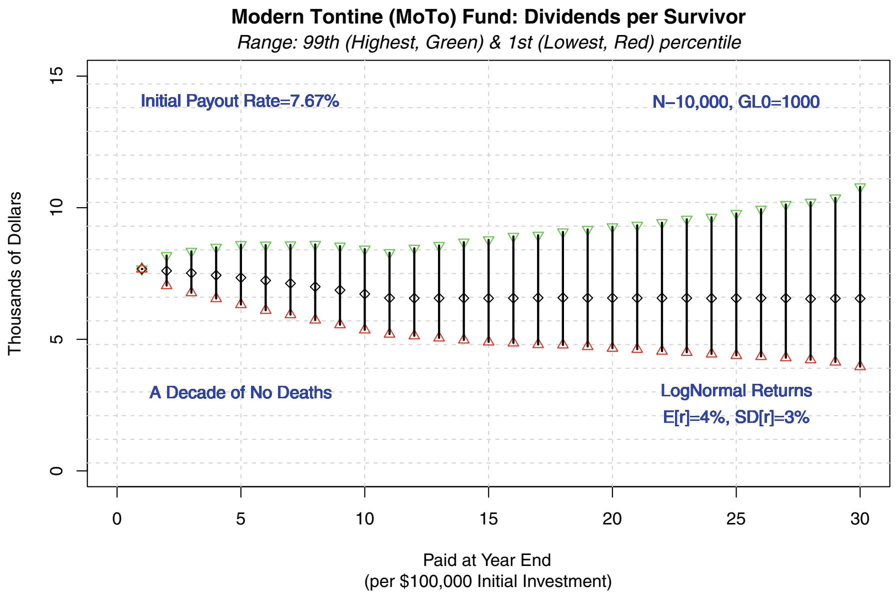


Fig. 4.5 Dividends: decade no death, then Gompertz resumes

To wrap up this particular discussion one could summarize the last few pages by loosely declaring that the absence of any deaths during the first decade will reduce the tontine payout rate by about 110 or so basis points, from the initially promised 7.67 to the sustainable 6.55, albeit slowly and over the course of a decade. Needless to stay, this doesn't account for the standard deviation or dispersion in payouts, but it does enable users and readers to intuit the impact of misestimating mortality.

4.5 Conclusion and What's Next

In some sense, my job is done. With this chapter I have delivered the core of the simulation and discussed a number of risk management aspects. In theory, you can use Chap. 3 and this chapter to design and manage a very simple and crude *modern tontine*. The next few chapters are really about additions, augmentations and alternative formulations.

4.6 Test Yourself

1. Assuming that in fact nobody dies in the first decade of the tontine fund, and that mortality only kicks-in after the age of 75, please compute the number of *extra survivors* that this creates at the end of the `TH=30` year horizon. Does this have a material impact on the number of people who survive from age $x = 65$ to age $x = 95$? Explain this intuitively.

2. Along the same lines, please compute the long-run tontine dividend value (i.e. the intercept in the regression) if the modern tontine fund was set-up assuming the modal value of the Gompertz parameter was $m = 90$, but in fact realized mortality was (much lower, and) consistent with $m = 93$. What is the cost in basis points (i.e. initial yield versus eventual yield)? How many more survivors will this (90 vs. 93) lead to age 95? Explain intuitively.
3. Going back to the canonical (standard) simulation results, with $(x = 65, m = 90, b = 10)$, carefully examine the `TONDV` matrix and compute the number of scenarios in which the tontine dividend payout falls below 80% of the **original** payout κ_1, at some point over the 30 year horizon. In other words, what is the probability of a 20% (or more) reduction in the cash-flow provided by the annuity, over the retirement horizon? What is the probability of (only a) a 10% or more reduction?
4. Similar to the prior question, but subtly different, what is the probability that at any point during the life of the fund the tontine dividend is reduced by 20%? Notice that this "event" is a larger subset of cases, because it also includes the situation in which tontine dividends are increased (in year 5, for example) and then reduced (in year 10, for example) so that from peak to trough the reduction was 20% or more.
5. Imagine that every single year the sponsor or manager extracts or removes \$100,000 from the fund (per \$100 million of initial fund value) to cover operating expenses. Clearly, the theoretical κ_1 payout rate is no longer sustainable and the tontine dividends will experience a negative drift over time, as evidenced by the regression slope coefficient. Using a *numerical* process of trial and error, and again assuming the canonical parameter values, please locate the revised value of $\hat{\kappa}_1$ that will support a stable and non-declining tontine dividend over time. How many basis points of initial yield $\kappa_1 - \hat{\kappa}_1$ does this annual \$100,000 fixed withdrawal cost the shareholders in the fund? Are there any other issues or problems that are encountered when \$100,000 is extracted every year?

5 Death Benefits, Refunds and Covenants

This chapter implements a design feature in which investors in the modern tontine are assured they will be *made whole* if they happen to die prior to receiving their original investment back in periodic payments. This is the symmetric equivalent of a cash-refund income annuity, which incidentally is the most popular rider in the US market. Now obviously, being that we are operating in a universe of modern tontines it's impossible to *guarantee* or to *promise* anything. Nevertheless, this chapter begins by presenting what I am officially labelling `v2.0` of the tontine R-script. In contrast to `v1.0`, this simulation algorithm adds an *expected* cash refund payment to those investors who aren't fortunate enough to be blessed with longevity. In other words, both people who die (early) and people who live (late) will get something from the modern tontine. This feature must come at the expense of reduced tontine dividend payouts. How much this actually *costs* investors in basis points and whether or not the fund is able to deliver on this *expected money back* feature are questions that will be addressed under a variety of numerical scenarios. Finally, the chapter concludes with another feature, one in which investors are allowed to change their mind and *lapse or surrender* the tontine. This is `v3.0` and the final version for this book.

5.1 Set Your Seed

I have noted that your simulation results will differ from what you see in the chapter due to the nature of randomness. The time has come to dispense with that rather inconvenient discrepancy with a simple fix: `set.seed()`. Do that before you generate random numbers with the same *seed* as mine and your results will be identical. Let's give it a try. Start with 1000 investors who are $x = 65$ years old and simulate deaths before the age of $y = 80$. Recall that in a Gompertz model in which $(x = 65, m = 90, b = 10)$, the 15-year survival probability is: `TPXG(65,15,90,10)` is 75.14%, and probability of dying during those 15 years is: 24.86%. We anticipate *losing* one quarter within the first 15 years, but the realized number will be random. It can be generated with the following script:

```
set.seed(1693)
rbinom(5,1000,0.2486)
> 254 242 264 242 279
```

The `rbinom()` command is quite familiar by now, but the new `set.seed(1693)` command assures the user they will generate and see the same 5 random numbers noted above. Give it a try and see how it works. Run the above command `rbinom(5,1000,0.2486)` without setting the seed first and your five numbers will obviously differ from mine. But by forcing the R-script to initialize the random numbers at a starting *seed* value of 1693—which is completely arbitrary, by the way—your simulation results will match mine. The results and set of five random numbers represent possible realizations of total deaths over the first 15 years of the tontine pool. As a refresher, in one scenario 254 people die and in another 242 die, etc. The important thing is that you have now learned how to generate simulation results that precisely match the ones in the text. If you want to see different results—-or you don't like mine—then pick any other initial number for the seed; other than `1693` of course. If you are curious about what exactly the `set.seed()` does and how it relates to *linear congruential generators*, then please google exactly that phrase. And, if you are wondering why I personally picked those four numbers for my seed, when in fact you can select any number of digits, the answer is rather trite: It's the year in which England launched its first government-sponsored tontine, a.k.a. *King William's Tontine.*

5.2 Can You Get Your Money Back?

Although it's possible for traditional insurance companies to sell *conventional life annuities* with a guaranteed refund-at-death feature, an investment company attempting to do so with a *modern tontine* is likely to run into some problems. There are three subtle reasons for this. First, conventional life annuity offerings are predicated on the *law of large numbers*, which assumes a very (very) large pool of annuitants. This (theoretically) guarantees that realized mortality will converge to the assumed mortality and that what happens is exactly as expected. In other words, by virtue of the large size of the annuitant pool an insurance company can predict with much greater certainty when they will have to pay death benefits, as well as the magnitude of those payments. In contrast, an investment fund company sponsoring a modern tontine with a mere few thousand investors can't afford to rely on the LLN. Recall that our simulations assumed a small finite group, a far cry from infinity, especially later when survivors decline. Second, insurance companies issue life annuities to different ages and issue life insurance policies across the entire lifecycle. Indeed, the nature of their business centres around continuously *increasing* the size of the pool. In our basic `v1.0` design, the pool is closed and doesn't admit newcomers after the initial period.

The third and final reason conventional insurance companies can offer and guarantee life annuities with a refund-at-death with confidence is that they invest the underlying assets in ultra-safe bonds and fixed-income products, which yield positive interest rates under most normal economic circumstances. Yes, interest rates might be at historical lows (today) and bonds themselves can lose value (tomorrow) if-and-when rates increase, but the investment portfolio of life insurers (or at least their general account) is extremely safe and highly regulated. What this effectively means is that insurance companies are unlikely to find themselves in a situation or simulated scenario in which they don't have enough money to pay promised death benefits. Contrast those safe and staid companies with an upstart investment or fund company that allocates the assets underlying the modern tontine to fluctuating stocks and bonds. Even if the standard deviation of returns σ is quite low, there is still a non-zero probability returns will be negative during the first few years (a.k.a. sequence of returns) **and** a large number of investors die early, leading to the rapid decline and perhaps even bankruptcy of the modern tontine.

At the risk of yelling—and although the next few pages are entirely focused on allowing for a refund-at-death feature—I should warn readers and users that unlike the basic (v1.0) modern tontine with a self-correcting payout thermostat, death benefits introduce a completely new risk into the picture: mission failure, financial ruin and bankruptcy. So, while a few hundred or even thousand simulations might not uncover or locate those rare scenarios, it's a theoretical possibility that is ever present. Moreover, at the end of this chapter when I introduce additional features that allow for ongoing liquidity this concern will be even more acute. My yelling will get louder. This is exactly why sponsors of modern tontines should be (and are) careful to attach the appellation *expected* to the phrase *refund-at-death*, or anything else for that matter. I will call it a death covenant and thus avoid the charged term guarantee.

To set the stage for why the R-script must be modified in a non-trivial manner, Fig. 5.1 graphically displays what would happen if a tontine sponsor decided to manage a fund in which the dividend is (imprudently) set at the original κ_1 value of 7.67%, per the results in the prior chapter but also provides a death benefit to beneficiaries. If at the time of (early) death the annuitant had *not* received a sum total of $f_0 = \$100{,}000$ back in dividends, the beneficiaries would receive the unpaid capital. That is akin to a *cash-refund* immediate annuity, when sold and issued by an insurance company. Go back to the end of Chap. 2 for a refresher.

If you look at the figure from a distance, it echoes the footprint of a normal (stochastic) tontine dividend whose cone of uncertainty increases over time. But upon careful inspection and regression, you will notice that the average tontine dividend drifts downward over time, and in a statistically significant manner. The original 7.67% tontine dividend is not sustainable over time because the fund assets are *leaking* death benefits, and rather large amount in the early life of the fund when `TCPAY` is rather small. Yes, eventually this fund stabilizes when everyone has received more than their original $f_0 = \$100{,}000$ back in dividends, but the

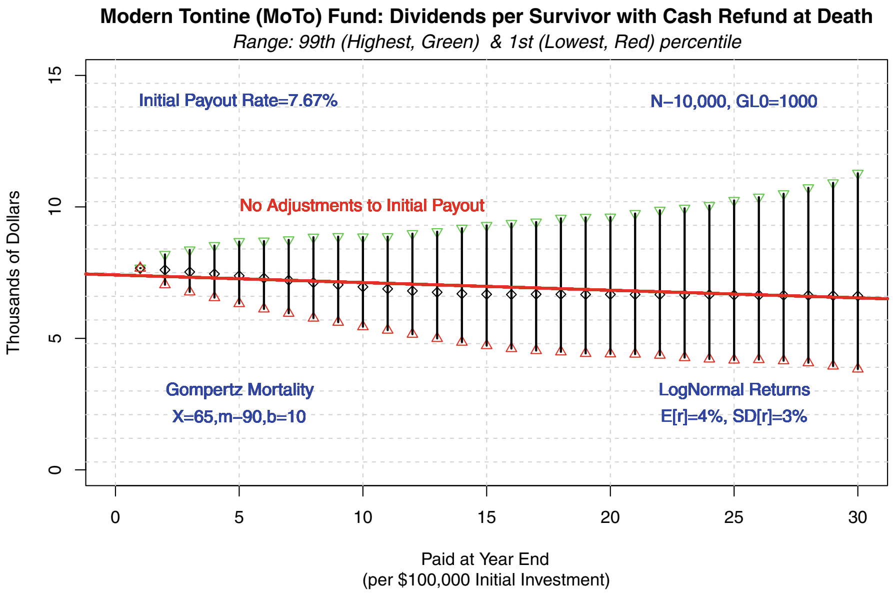


Fig. 5.1 When promises are greater than reserves (98% C.I.)

new plateau is at a much lower (average) level than $7,671. What does all of this mean? Well, if the tontine sponsor wants a stable tontine dividend stream (a.k.a. regression coefficient of zero), they must reduce the original payout. The usual κ_1 is too generous.

5.3 Modern Tontine v2.0

In this section I present the basic R-script for a modern tontine that *anticipates and reserves* for a refund-at-death. To be clear and perhaps at the risk of over-explaining, this entails promising investors *some* tontine dividend, **plus** if they happen to die before receiving their entire (for example) $100,000 investment back, their beneficiary will receive a one-time death benefit of $100,000 minus (what I previously labelled) `TCPAY()`, if positive. The relevant question then becomes how much does this design feature cost? More importantly, how much can the fund sponsors afford to pay out in *stable* tontine dividends each year, and yet still have enough reserves to pay those anticipated death benefits? The following script does all that (and more). I will break down the script into three distinct segments that should be fused and run together, but will explain the main features of each individually. Many of the commands, loops and algorithms will look familiar and echo `v1.0`.

```
# Modern Tontine (MoTo) Fund Version 2.0
# Cash Refund at Death with Proper Reserve
set.seed(1693)
# Calculate Gompertz survival probability.
TPXG<-function(x,t,m,b){exp(exp((x-m)/b)*(1-exp(t/b)))}
# Calculate probability of death between times t1 and t2.
TQXG<-function(x,t1,t2,m,b){TPXG(x,t1,m,b)-TPXG(x,t2,m,b)}
# Value temporary life annuity from age x to age y.
TLIA<-function(x,y,r,m,b){
  APV<-function(t){exp(-r*t)*TPXG(x,t,m,b)}
  sum(APV(1:(y-x)))}
# Value temporary life annuity with $1 reducing death benefit.
TLIA_RDB<-function(x,y,r,m,b,RDB){
  periods<-(y-x)
  value<-0
  for(i in 1:periods){
    PV<-exp(-r*i)
    # The next two lines should be combined.
    value<-value+TPXG(x,i,m,b)*PV+max(RDB-(i-1),0)
    *TQXG(x,(i-1),i,m,b)*PV}
  value}
# Function searches for RDB value that equals TLIA_RDB.
RTLIA<-function(x,y,r,m,b){
  RTLIA1<-function(a){abs(TLIA_RDB(x,y,r,m,b,a)-a)}
  optimize(RTLIA1,interval =c(0,1/(r+0.0001)))$minimum}
```

After the `set.seed()` command that ensures readers and users can replicate my numbers, the R-script refreshes `TPXG()` and introduces the `TQXG` command, which is the probability of dying in between two given times `t1<t2`, assuming the annuitant is alive at age `x`. After those relatively basic Gompertz-based functions, the R-script moves-on to define the previously used temporary life annuity factor `TLIA`, and then the temporary life annuity with a *reducing death benefit*, denoted by `TLIA_RDB`. As explained in the technical background of Chap. 2, such an annuity offers annual payments of $1 plus an additional death benefit of `RDB`, but reduced by $1 for every year the person has lived. So, if they die after 5 years, the death benefit is `RDB` minus 5, etc.

Remember that if-and-when the entire `RDB` has been returned to the investor (while they are alive), the death benefit obligation disappears ceases to exist. For example, if the `RDB` was set to $10 and the individual happens to die after year 10, their death benefit will be zero in that scenario. Note that in this function the `RDB` is a completely free parameter, entirely arbitrary and up to the user. It could be $100, which will take 100 years to reduce to zero, or a mere $5, which will terminate after 5 years. Of course, the higher the `RDB` specified by the user, the higher the numerical value of `TLIA_RDB`. It costs more.

The final `RTLIA` function in the above-segment uses the built-in *R* function `optimize` to locate the numerical value of the `TLIA_RDB` function that is equal to `RDB` itself. This is a *fixed point* argument of iterative technique required to obtain the present value of the annuity. For example

`RTLIA(65,100,0.04,90,10)=14.335`, which means that `TLIA_RDB` with a reducing death benefit of \$14.335, is exactly equal to \$14.335. Play around with this function before you move-on to the next segment, so you get a sense of how `TLIA` differs from RTLIA, and why the latter is higher (and more expensive) than the former.

The next (second) segment within the `v2.0` code is almost identical to its sibling in `v1.0`. It starts out by initializing the parameters used in the simulation, and no new parameters are added or used at this stage. The only change from the initial version of the algorithm—and really this is the only difference—is that I have added a new matrix `AGDEB` that is intended to keep track of the total amount that is paid to the people who die in each one of the `N` simulation paths, during each of the `TH` years. The simulation of life and death, as well as the investment returns is identical to `v1.0`. One can easily randomize the modal value of the Gompertz parameter m, or the dispersion parameter b, using the ideas discussed in Chap. 3.

```
set.seed(1693)
# All base parameters are set here.
x<-65; m<-90; b<-10; GL0<-1000; TH<-30; N<-10000
EXR<-0.04; SDR<-0.03; r<-0.04; f0<-100
# Placeholder for our data.
kappa<-c()
GLIVE<-matrix(nrow=N,ncol=TH)
GDEAD<-matrix(nrow=N,ncol=TH)
TCPAY<-matrix(nrow=N,ncol=TH)
AGDEB<-matrix(nrow=N,ncol=TH)
PORET<-matrix(nrow=N,ncol=TH)
DETFV<-matrix(nrow=N,ncol=TH)
TONDV<-matrix(nrow=N,ncol=TH)
# Compute Deaths and Survivors Here (V2.0, no change).
for (i in 1:N){
  GDEAD[i,1]<-rbinom(1,GL0,1-TPXG(x,1,m,b))
  GLIVE[i,1]<-GL0-GDEAD[i,1]
  for (j in 2:TH){
    GDEAD[i,j]<-rbinom(1,GLIVE[i,j-1],1-TPXG(x+j-1,1,m,b))
    GLIVE[i,j]<-GLIVE[i,j-1]-GDEAD[i,j]
  }}
# Compute Investment Returns Here (V2.0, no change)
for (i in 1:N){
  PORET[i,]<-exp(rnorm(TH,EXR,SDR))-1}
```

The third and final segment is the primary source of the difference between the *“no death benefit”* `v1.0` and the *“reimbursement covenant”* `v2.0`. More specifically, the new matrix `AGDEB`, which was defined in the second segment and which keeps track of the new death benefits, now gets populated. This then affects the evolution of the fund itself `DETFV` as well as the tontine dividends `TONDV`. I’ll delve into specific aspects of the code a little bit later. For now, the R-script must start by adjusting the payout rate vector κ, a process that isn’t as simple or easy as the prior temporary life annuity factor.

```
# Revised Initial Payout Rate to Provide Death Benefit
kappa[1]<-1/RTLIA(x,x+TH,r,m,b)
# The next two lines should be combined.
for (j in 2:TH){kappa[j]<-1/
  TLIA_RDB(x+j-1,x+TH,r,m,b,max(1/kappa[1]-j+1,0))}
# Compute fund values and dividends.
for (i in 1:N){
  # First we address the first period.
  # Define first dividend (per individual).
  TONDV[i,1]<-kappa[1]*f0
  # Define cumulative dividends paid (per individual).
  TCPAY[i,1]<-TONDV[i,1]
  # Define aggregate death benefits paid (for fund).
  AGDEB[i,1]<-f0*GDEAD[i,1]
  # Define total paid out by fund this period.
  outflow<-TONDV[i,1]*GLIVE[i,1]+AGDEB[i,1]
  # Define fund value.
  DETFV[i,1]<-f0*GL0*(1+PORET[i,1])-outflow
  # Loop through remaining periods.
  for (j in 2:TH){
    if (GLIVE[i,j]>0){
      # Define dividend per individual in year j.
      TONDV[i,j]<-kappa[j]*DETFV[i,j-1]/GLIVE[i,j-1]}
    else {TONDV[i,j]<-0}
    # Define cumulative dividends paid to date (per person).
    TCPAY[i,j]<-TCPAY[i,j-1]+TONDV[i,j]
    # Define aggregate death benefits paid (for fund).
    AGDEB[i,j]<-max(f0-TCPAY[i,j-1],0)*GDEAD[i,j]
    # Define total paid out by fund this period.
    outflow<-TONDV[i,j]*GLIVE[i,j]+AGDEB[i,j]
    # Define current fund value.
    DETFV[i,j]<-DETFV[i,j-1]*(1+PORET[i,j])-outflow}}
```

For now, I suggest readers **run** the above-noted code as-is and ensure that it compiles correctly on their platform or system. To verify that your code is indeed working properly, please confirm the following three numbers. First, the `mean(TONDV)` should be exactly `7.123656`, which is 12 cents over 7 dollars in tontine dividends per $100 investment. Second, the mean value of the underlying fund value in year 30 should be `mean(DETFV[,30])=0.6662558` thousand dollars. Third and finally the slope coefficient of the above regression should be `-0.0019254` and statistically indistinguishable from zero. Remember that that lack of trend (or slope) is good news and a sign of tontine stability, which is something I explained previously in Chap. 4.

Although all these numbers were generated by simulation, the `set.seed (1693)` ensures that your results are identical to mine. Once you confirm `v2.0` is indeed working correctly, I would urge users to plot the tontine dividends and plot the fund values with the usual scripts, to visually confirm everything is in good order.

In fact, perhaps change the (x, m, b) parameters up or down, to confirm the tontine payouts move in the proper direction. Change $(r = \nu, \sigma)$ and confirm that your results are reasonable, and perhaps even change the `set.seed()` a few times, to see how the simulation results change with different initial seed values.

Congratulations! You have learned to ride a simple bike without falling off, even though you aren't a mechanical engineer. Now it's time to dig deeper into the theory and understand exactly what this algorithm (or mechanism) is doing, and specifically how to use and interpret the new `AGDEB` numbers.

5.4 The Intuition of Refunds

Start by examining Fig. 5.2, which is a picture that simply doesn't exist in `v1.0` of the tontine design. It displays the range of payouts that the tontine sponsor must make to all the investors who die in the first few years, and whose beneficiaries demand a return of the unearned investment under the reimbursement covenant. Notice the interesting parabolic shape of this figure. Initially, around age $x = 65$, few investors are "expected" to die, so the payout is relatively small. Then, as age increases and mortality rates accelerate the number of investors dying continue to increase—think of the PDF of the remaining lifetime random variable—but the deceased have died sufficiently late enough so they have recovered a large part of their original investment. Finally, somewhere around year 15 or so, although deaths continue to accelerate (per the Gompertz law) most of the deceased have already received their original investment back in tontine dividends and the beneficiaries have no claim on the tontine fund. The obligation is very quickly and rapidly defused somewhere between the 17th and 18th year. After that date, the sponsors can concern

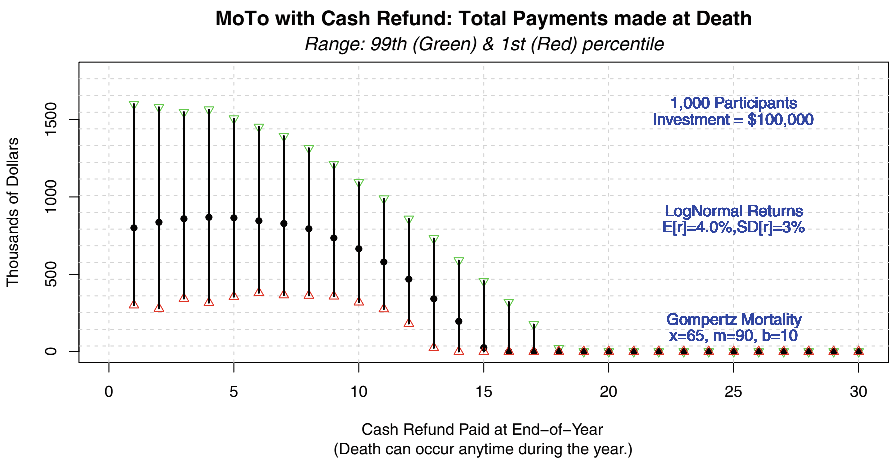


Fig. 5.2 Total death benefits paid to investors (98% C.I.)

themselves exclusively with the living, since the deaths will generate no obligations. The next and final question with this revised mandate is *but are the dividends stable?*

5.5 The Tontine Dashboard

In this section I would like to introduce and present a standardized way of reporting simulation results and in particular displaying the distribution of tontine dividends, fund values and other related financial information. I call this the *dashboard*. Here is how to interpret the results in Table 5.1.

We start with `GL0=1000` investors, each of whom contribute \$100,000 to the tontine pool, for a total initial fund of \$100 Million, which is invested in a portfolio of assets that are expected to grow at a (continuously compounded) rate of: $\nu = 4\%$ per year, with a standard deviation of: $\sigma = 3\%$. The valuation (a.k.a. crediting) interest rate denoted by r is also set to be 4%, which doesn't necessarily have to be the case. Finally, the demographic or biometric parameters are Gompertz, with $(x = 65, m = 90, b = 10)$, and a time horizon of `TH=30` years, with a fund liquidation age of $y = 95$, for those who survive. Now, being that we are simulating a `v2.0` modern tontine with a refundable investment death benefit, the initial tontine dividend must be set using the `RTLIA` function, and not the `TLIA` function from Chap. 3.

```
1/RTLIA(65,95,0.04,90,10)
> 0.07073756
1/TLIA(65,95,0.04,90,10)
> 0.07670865
```

This means that the first year's tontine dividend—paid at the very end of year one, when investors are all about to turn 66—is 7.073756% of \$100,000, or \$7,074 per survivor. That is the first data column in Table 5.1 and doesn't vary by scenario.

Table 5.1 Tontine dashboard: dividend distribution; replicate with `set.seed(1693)`

Modern Tontine (MoTo) Fund: Simulation Dashboard Lifetime Income with Refundable Death Benefit					
Statistical	**DIVIDENDS: End of Year Number...**				
Outcome	$T = 1$	$T = 5$	$T = 10$	$T = 20$	$T = 30$
1 pct. (worst)	\$7,074	\$6,069	\$5,538	\$4,664	\$4,060
25 pct.	\$7,074	\$6,769	\$6,581	\$6,305	\$5,979
Median	\$7,074	\$7,074	\$7,065	\$7,059	\$7,000
75 pct.	\$7,074	\$7,383	\$7,589	\$7,886	\$8,203
99 pct. (best)	\$7,074	\$8,315	\$8,979	\$10,162	\$11,929
St. Dev.	\$0	\$470	\$743	\$1,189	\$1,683
Assumptions:	Financial: $r = 4.0\%$, $\nu = 4.0\%$, $\sigma = 3.0\%$				
$N = 10000$	Gompertz: $x = 65$, $m = 90$ and $b = 10$.				
`TH=30`	Investors: $GL_0 = 1000$ with $f_0 = \$100,000$.				

That dividend is set in stone at time $T = 0$ and will have to be paid out to all survivors regardless of the performance of markets as reflected by `PORET` or the actual number of survivors as reflected by `GLIVE`.

It's important to remember this small modicum of risk that affects the way I have constructed the *modern tontine*. Namely, the payout rates are stated and declared at the start of the year (or period), but only paid to survivors at the end. This is like an interest rate on a bank deposit, or a dividend that is declared today but payable to the holders of record at some future time. Note that investors who die in that first year, before reaching the end of their 65th year of life, will be entitled to receive their entire $f_0 = \$100{,}000$ investment as part of the death benefit reimbursement covenant. All of this discrepancy in timing adds some additional commitments and risks to the sponsor of the *modern tontine*. Moreover, they certainly can't allocate the \$100 Million fund to bitcoin or something equally volatile, unless the payments themselves are made in those units. The reimbursement covenant is the precise reason the payout to survivors is pegged at the `RTLIA` value of approximately 707 basis points, versus the `TLIA` value of 767 basis points. The survivors sacrifice 60 basis points of dividends (every year, for 30 years) to ensure enough funds are set aside for those who die.

Moving on to the other columns, which by now should be familiar and well understood, the table displays the range of values—from the 1st percentile to the 99th percentile—in years 5, 10, 20 and 30. To be precise, they are values at the *end* of those years but are set and known at the beginning of the year. Think of an *ordinary* annuity versus an annuity *due* in the basic time value of money. The median tontine dividend is stable—remember the regression coefficient is statistically indistinguishable from zero—and fluctuates between \$7000 and \$7100 per year. But, what's clear from the columns is that as time marches on the pool shrinks and the number of survivors diminish, the range (and standard deviation) grows.

5.6 Lapses, Surrenders and Other Regrets

In this final section of the chapter I would like to address a more complex question. Namely, whether or not it's possible to design a *modern tontine* in which individual investors are allowed to voluntarily liquidate, sell or surrender (also known as lapse) and receive the market value in cash instead of units. Let's think about this process carefully before we rush to the R-scripts and the simulation laboratory. At first glance—especially obvious to anyone within the actuarial and insurance world—providing this sort of liquidity to investors would lead to a problem economists call *anti-selection*. Namely, if the investor happened to (sadly, unfortunately) get diagnosed with a life threatening disease, meaning they won't be around for much longer and won't benefit from the longevity insurance, they would immediately liquidate and ask for their money back. So what, you ask? Well, the problem with that behaviour is that it would leave only the healthy people in the *modern tontine* pool, which would then erode if not eliminate the mortality credits for everyone else. Of course, selling you shares and liquidating your investments might not be the first (or even second or third) thing on your mind should you happen to receive such horrible news, but it would definitely be something on the *To Do* list, as the end came near. Namely, sell, liquidate, lapse and get some money from this before it dies with you.

In fact, my first reaction when I was asked about adding a liquidity provision to a *modern tontine* was a straight out no. It would *spring a leak in the pool* and wouldn't allow me to assume (nice, smooth) Gompertz mortality and longevity *terminations*.

But on further and deeper inspection it really depends on how high (or low) the initial payout rate was set, and what was assumed at the time of pricing. After all, if the payout rate κ is set low-enough (think zero) you could allow any behaviour. Moreover, it should be possible to offer some amount of liquidity (although certainly not 100%), when the *modern tontine* includes a covenant that rebates unreturned capital (UC) at death, which was associated with a lower κ value. Here's the logic. If investors are allowed to leave the fund early and voluntarily but are penalized for this premature departure by forfeiting some of their equity in the fund to the benefit of others, perhaps such behaviour might actually benefit everyone else. After all, had this person who was just diagnosed with cancer *died* with unreturned capital, they would have received 100% of the UC. Hence, if the *modern tontine* sponsor allows them to leave early out of compassion, and they forfeit some of the UC, then the rest of the investors in the fund might not *lose* from this leniency and could actually benefit. Remember that the initial κ payout is lower when the fund includes a death benefit covenant.

Another point to keep in mind is that the more of the UC investors forfeit if they choose to lapse, the less likely it is that investors would actually choose to lapse and take this lower payout. So the lapsation rate and the lapsation penalty are not independent in the real world; however, for simplicity here we will treat them as two independent parameters.

In sum, despite concerns over *anti-selection*, it's possible to offer liquidity provided that *nothing* is guaranteed and that this feature—like the death benefit—is positioned as a covenant, one that can be bypassed or even ignored in extreme (negative) market conditions.

With this out of the way, we can get back to the simulation laboratory and modify the script so that it accounts for this feature. Namely, we will now simulate tontine payouts when a *random* number of investors decide to liquidate their investment every year—while there is still some UC left—and then pay a penalty. Now, one could model the rate at which investors lapse or surrender the *modern tontine* in many different ways, but in what follows I will assume that lapsing is just another form of dying. Namely, there is a *force of lapsation* similar to a *force of mortality* and it removes people from the `GLIVE` matrix, just like dying does.

I won't assume Gompertz lapsation—which has no empirical basis and is unnecessarily complex—and instead will assume a constant force of *lapsation*, which means that lapse time is exponentially distributed. Yes, this assumption is completely ad hoc, but I think of this as (for example) decrementing or losing 2% of people every single year for the first few years, but to a **third state** that is neither live nor dead. Those who move into that third state are entitled to receive 80% or 70% or perhaps as low as 50% of their unreturned capital. Had they (patiently) waited to die their heirs and estate would have received 100% of the UC, but perhaps they need the money now, or they don't want to wait, etc. Remember though, once the UC has been reduced to zero and all the original money has been returned as tontine dividends, those who lapse or die receive nothing back.

In the next few pages I will describe how to simulate the tontine dividends when random (non-strategic) lapsation is allowed, which will require two more new parameters, or perhaps even vectors that are time dependent. The first is the (deterministic, known, fixed) annual lapse rate, which I'll label η_j (Greek eta), think of 2% a year for the first 15 years for example. And the second is the penalty or surrender charge that is paid upon live exit—money that is shared between the remaining investors over time—when the shares are lapsed. I'll use the symbol and variable `dsc` to remind us this is a type of deferred surrender charge. I repeat that the η_j parameter represents a deterministic force of lapsation that is uncorrelated with either investment returns `PORET` or the number of deaths `GDEAD` or anything else that is moving in the model. It's superimposed exogenously without assuming any dynamic behaviour or optimizing behaviour. This is the first (baby) step. That said, to keep everyone safe, I will **not** change or increase the initial payout κ to account for this lapsation in advance. In other words, using an actuarial term, the tontine will not be *lapse supported.* If-and-when these investors lapse, the extra money will be added to the tontine fund as if it were another type of investment return and then shared over time.

With all of that preamble and introduction out of the way, here are the changes that must be made to `v2.0` so that users can simulate tontine dividends, fund values and survivors when we add a (deterministic) lapsation rate. For the sake of preserving space, I will display the changes that have to be made to the R-script, as opposed to printing yet-again the entire script, which I will present at the very end of this chapter. Here is the first addition to the script, to create version `v3.0`.

```
# Parameters that Govern Lapsation and Redemption
eta<-c(rep(0.02,15),rep(0,TH-15)); dsc<-0.25
LAPSE<-matrix(nrow=N, ncol=TH)
AGLAP<-matrix(nrow=N,ncol=TH)
```

Note that these few lines initialize the lapsation rate vector η at 2% per year for the first 15 years, and then zero lapsation after year 15. Why? Because for the most part there is nothing left to lapse (or die) for, since the UC value is zero and the entire original principal has been returned. Remember that if you are paying out (on average) $\kappa_1 = 7.07\%$, then after $1/\kappa_1 = 14.14$ years the original principal has been paid out (again, on average). By year 15, those who lapse or die would get nothing, even in the absence of a `dsc`. The above segment also includes the initialization of the two new matrices that will keep track of the number of people who lapse, as well as the amount of money they will extract from the fund.

The next step is to actually generate and fill-in the `LAPSE` matrix, which requires the following bit of R-script. Technically this requires just two extra lines of code, and modifications to two other ones, but I have reproduced the entire section here so that readers can see the parallels and symmetry between GDEAD and LAPSE.

```
set.seed(1693)
for (i in 1:N){
  LAPSE[i,1]<-rbinom(1,GL0,eta[1])
  GDEAD[i,1]<-rbinom(1,GL0-LAPSE[i,1],1-TPXG(x,1,m,b))
  GLIVE[i,1]<-GL0-GDEAD[i,1]-LAPSE[i,1]
  for (j in 2:TH){
    LAPSE[i,j]<-rbinom(1,GLIVE[i,j-1],eta[j])
    # The following two lines should be combined.
    GDEAD[i,j]<-rbinom(1,GLIVE[i,j-1]-LAPSE[i,j]
                       ,1-TPXG(x+j-1,1,m,b))
    GLIVE[i,j]<-GLIVE[i,j-1]-GDEAD[i,j]-LAPSE[i,j]
  }
}
```

Note that the first thing that happens each year is people decide whether or not to lapse. If people choose to lapse they are out of the model and can no longer be added to the GDEAD count, even if they die that year. Here is the logical flow. We simulate people who lapse during the year by using the familiar rbinom random number generator based on the number of investors alive at the beginning of the year, and our lapsation parameter η. Then we use the same process, but with the Gompertz probability, for the number of people who die, out of the people who did not lapse. In this particular run we assume the annual lapsation rate η (eta) is 2% per year for the first 15 years, and then zero afterwards. To delve deeper into the code, once the η vector is zero the LAPSE matrix will consist of a bunch of zeros (which might be a bit wasteful in terms of storage space). All those zeros are generated from a binomial distribution that is degenerate (which is also a bit wasteful). The point here is to visualize and intuit the three states: Alive, Dead and Lapsed. Finally, we are ready to compute tontine payouts and fund values. That actually requires a wholesale rewriting of the final section in the code.

```
# Compute fund values and dividends.
for (i in 1:N){
  # First we address the first period.
  # First dividend (per individual).
  TONDV[i,1]<-kappa[1]*f0
  # Cumulative dividends paid (per individual).
  TCPAY[i,1]<-TONDV[i,1]
  # Aggregate death benefits paid (for fund).
  AGDEB[i,1]<-f0*GDEAD[i,1]
  # Aggregate lapsation payout (for fund).
  AGLAP[i,1]<-f0*LAPSE[i,1]*(1-dsc)
  # Total paid out by fund this period.
  outflow<-TONDV[i,1]*GLIVE[i,1]+AGDEB[i,1]+AGLAP[i,1]
  # Fund value.
  DETFV[i,1]<-f0*GL0*(1+PORET[i,1])-outflow
  # Loop through remaining periods.
  for (j in 2:TH){
    if (GLIVE[i,j]>0){
      # Dividend in year j (per individual).
      TONDV[i,j]<-kappa[j]*DETFV[i,j-1]/GLIVE[i,j-1]}
    else {TONDV[i,j]<-0}
    # Cumulative dividends paid to date (per individual).
    TCPAY[i,j]<-TCPAY[i,j-1]+TONDV[i,j]
    # Aggregate death benefits paid (for fund).
    AGDEB[i,j]<-max(f0-TCPAY[i,j-1],0)*GDEAD[i,j]
    # Aggregate lapsation payout (for fund).
    AGLAP[i,j]<-max(f0-TCPAY[i,j-1],0)*LAPSE[i,j]*(1-dsc)
    # Total paid out by fund this period.
    outflow<-TONDV[i,j]*GLIVE[i,j]+AGDEB[i,j]+AGLAP[i,j]
    # Current fund value.
    DETFV[i,j]<-DETFV[i,j-1]*(1+PORET[i,j])-outflow
  }
}
```

The above segment replaces the relevant section in `v2.0`. Note that the `PORET` matrix doesn't change, since lapsation doesn't really affect investment returns, although it does behave like a subsidy at time. In fact, the tontine dividend in the first year doesn't change from the basic simulation. The new matrix `AGLAP` computes the total amount that is paid out to all the *lapsers*. Notice how I have (once again) broken the computation down into two parts: The first year (loop) and then the remaining `(TH-1)` years. In either loop, the amount of money the *lapsers* are entitled to is their unreturned capital (UC), which is the original investment `f0` minus the `TCPAY` matrix, multiplied by one minus the surrender charge `dsc`. Note that I have also added the variable `outflow` just to keep things clear and neat. Other than this extra little leakage from decumulation tontine fund, nothing much has changed. We are ready for some numerical results and figures.

Running `v3.0` in which the Gompertz and return parameters are the usual suspects, but the new $\eta = 2\%$ for 15 years and the surrender charge is 25%, and seed of `1693` leads to a median tontine dividend (for survivors) of \$7,964 per year. Make sure you replicate this (with my seed). The total standard deviation across the $N = 1000$ scenarios is 18.5%. The slope of the canonical diagnostic regression is positive \$71 per year, and the intercept is `7.122` units and obviously quite significant statistically. What does this all mean? Well, survivors can expect higher, larger and better tontine dividends over time. Recall from earlier in the chapter that when the lapse rate was set to zero (`v2.0`) the median tontine dividend was \$7,074, which was almost \$900 per year lower. Why better now? Thank the lapsers who leave behind 25% of their UC.

Now, what happens if we reduce the `dsc` value from a 25% penalty to no penalty at all, but still assume that 2% of live investors lapse and surrender their investment every year? You can die or you can lapse. In both cases you get the UC. Well, if you run the R-script, the median tontine dividend is now \$7584, which is lower than the earlier-noted \$7964, applicable when the `dsc` is set to 25%. Intuition? Less money is forfeited to the fund so the average tontine dividend over time is lower. But even so, the \$7584 is still higher than the baseline (non-lapsing) \$7074 tontine dividend.

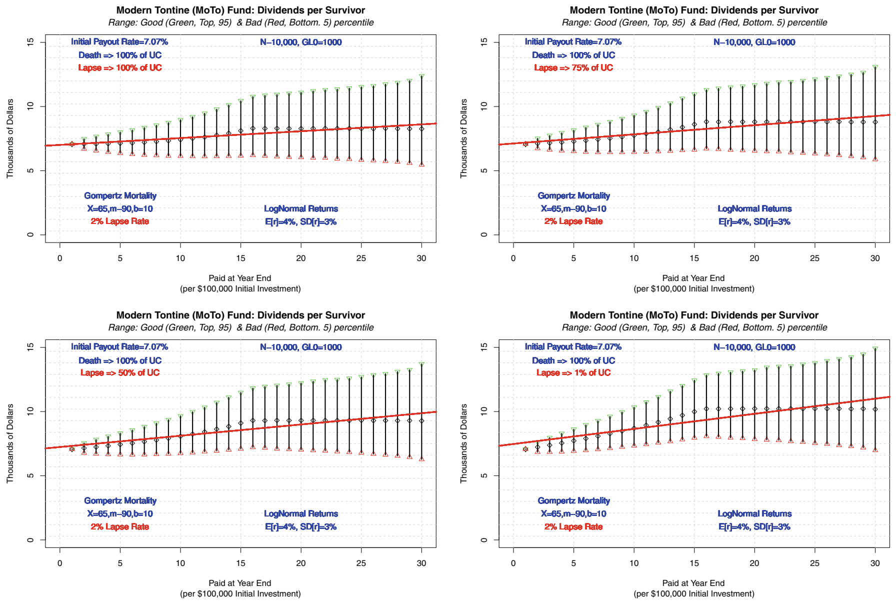


Fig. 5.3 Tontine dividends with liquidity (90% C.I.)

Why? Because the lapsers are forfeiting the *interest and investment gains* they would have earned on the unreturned capital. Think about this carefully and perhaps even play around with different values of `dsc` so you can intuit how it impacts tontine dividends. Figure 5.3 displays the (90% CI) output from four different values of the surrender charge. In the upper left-hand corner you can see the case when the `dsc=0` and 100% of the UC is paid out upon lapse, and the lower right-hand corner represents the case when those who lapse get a mere 1% (i.e. almost nothing) back. Naturally, everyone else gains (more) from that. Finally, play around with different lapse rate assumptions and see what happens. Things get a bit wild towards the end, and you will see why I chose to display the 90% and not 99% confidence intervals.

Note that in this script we did not change κ to account for lapsation. Therefore the dividends in this model are not expected to be constant (as you can see in Fig. 5.3), like we would like to see in a proper tontine model. We did not create a new type of tontine that allows for lapsation, and we merely demonstrated what dividends would look like if we allowed lapsation but kept the original `v2.0` tontine dividend structure. To actually create a tontine that accommodates lapastion and keeps expected dividends constant, we may require numerical (non-analytic) methods.

5.7 The Final Act: Version 3.0

Finally, although some of this might seem repetitive, for the sake of completeness I now reproduce the entire and most recent version of the **R**-script in one central location. This version 3.0 represents the final structural evolution of the *modern tontine* simulation.

```
# Modern Tontine Fund (MoTo) Version 3.0
# Cash Refund at Death with Reserves and Lapses.

set.seed(1693)
# All base parameters are set here.
x<-65; m<-90; b<-10; GL0<-1000; TH<-30; N<-10000
EXR<-0.04; SDR<-0.03; r<-0.04; f0<-100

# Parameters that Govern lapsation and redemption.
# Replace 0.00 with appropriate rate during 15 years.
eta<-c(rep(0.00,15),rep(0,TH-15)); dsc<-0.0

# Calculates Gompertz survival probability.
TPXG<-function(x,t,m,b){exp(exp((x-m)/b)*(1-exp(t/b)))}

# Calculate probability of death between times t1 and t2.
TQXG<-function(x,t1,t2,m,b){TPXG(x,t1,m,b)-TPXG(x,t2,m,b)}

# Values temporary life annuity from age x to age y.
TLIA<-function(x,y,r,m,b){
  APV<-function(t){exp(-r*t)*TPXG(x,t,m,b)}
  sum(APV(1:(y-x)))}

# Values temporary life annuity with $1 reducing death benefit.
TLIA_RDB<-function(x,y,r,m,b,RDB){
  periods<-(y-x)
  value<-0
  for(i in 1:periods){
    PV<-exp(-r*i)
    # The next two lines should be combined.
    value<-value+TPXG(x,i,m,b)*PV+max(RDB-(i-1),0)
    *TQXG(x,(i-1),i,m,b)*PV}
  value}

# Function searches for RDB value that equals TLIA_RDB.
# This is the fixed point of the TLIA_RDB function.
# RTLIA Provides Declining Death Benefit, Replacing Basic TLIA.
RTLIA<-function(x,y,r,m,b){
  RTLIA1<-function(a){abs(TLIA_RDB(x,y,r,m,b,a)-a)}
  optimize(RTLIA1,interval =c(0,1/(r+0.0001)))$minimum}

# Placeholder for our data.
kappa<-c()
GLIVE<-matrix(nrow=N,ncol=TH)
GDEAD<-matrix(nrow=N,ncol=TH)
LAPSE<-matrix(nrow=N, ncol=TH)
```

```
TCPAY<-matrix(nrow=N,ncol=TH)
AGDEB<-matrix(nrow=N,ncol=TH)
AGLAP<-matrix(nrow=N,ncol=TH)
PORET<-matrix(nrow=N,ncol=TH)
DETFV<-matrix(nrow=N,ncol=TH)
TONDV<-matrix(nrow=N,ncol=TH)

# Compute lapses, deaths and survivors.
for (i in 1:N){
  LAPSE[i,1]<-rbinom(1,GL0,eta[1])
  GDEAD[i,1]<-rbinom(1,GL0-LAPSE[i,1],1-TPXG(x,1,m,b))
  GLIVE[i,1]<-GL0-GDEAD[i,1]-LAPSE[i,1]
  for (j in 2:TH){
    LAPSE[i,j]<-rbinom(1,GLIVE[i,j-1],eta[j])
    # The next two lines should be combined.
    GDEAD[i,j]<-rbinom(1,GLIVE[i,j-1]
                       -LAPSE[i,j],1-TPXG(x+j-1,1,m,b))
    GLIVE[i,j]<-GLIVE[i,j-1]-GDEAD[i,j]-LAPSE[i,j]}}

# Continuously Compounded Return is Normally Distributed.
# The Effective Annual Return is placed into PORET.
for (i in 1:N){
  PORET[i,]<-exp(rnorm(TH,EXR,SDR))-1}

# The initial payout rate for Modern Tontine fund.
# RTLIA (instead of TLIA) includes the declining death benefit.
kappa[1]<-1/RTLIA(x,x+TH,r,m,b)
# The next two lines should be combined.
for (j in 2:TH){kappa[j]<-
  1/TLIA_RDB(x+j-1,x+TH,r,m,b,max(1/kappa[1]-j+1,0))}
# Compute fund values and dividends.
for (i in 1:N){
  # First we address the first period.
  # Define first dividend (per individual).
  TONDV[i,1]<-kappa[1]*f0
  # Define cumulative dividends paid (per individual).
  TCPAY[i,1]<-TONDV[i,1]
  # Define aggregate death benefits paid (for fund).
  AGDEB[i,1]<-f0*GDEAD[i,1]
  # Define aggregate lapsation payout (for fund).
  AGLAP[i,1]<-f0*LAPSE[i,1]*(1-dsc)
  # Define total paid out by fund this period.
  outflow<-TONDV[i,1]*GLIVE[i,1]+AGDEB[i,1]+AGLAP[i,1]
  # Define fund value.
  DETFV[i,1]<-f0*GL0*(1+PORET[i,1])-outflow
  # Loop through remaining periods.
  for (j in 2:TH){
    # Define dividend per person in year j.
    TONDV[i,j]<-kappa[j]*DETFV[i,j-1]/GLIVE[i,j-1]
    # Define cumulative dividends paid to date per individual.
    TCPAY[i,j]<-TCPAY[i,j-1]+TONDV[i,j]
    # Define aggregate death benefits paid (for fund).
    AGDEB[i,j]<-max(f0-TCPAY[i,j-1],0)*GDEAD[i,j]
    # Define aggregate lapsation payout (for fund).
    AGLAP[i,j]<-max(f0-TCPAY[i,j-1],0)*LAPSE[i,j]*(1-dsc)
    # Define total paid out by fund this period.
    outflow<-TONDV[i,j]*GLIVE[i,j]+AGDEB[i,j]+AGLAP[i,j]
    # Define current fund value.
    DETFV[i,j]<-DETFV[i,j-1]*(1+PORET[i,j])-outflow}}
# Regression model on median dividend as a function of time.
mtd<-c(); t<-1:TH
for (j in 1:TH){mtd[j]<-median(TONDV[,j])}
fit<-lm(mtd~t)
# Intercept summary value, slope should be statistically zero.
summary(fit)
```

As with prior **R**-scripts that included commands that are too long, please remember to append them when you compile these scripts. They should be clear from the context.

5.8 Conclusion

With this chapter, you should now be able to create and offer a *modern tontine* that consumers might actually be interested in purchasing. So, while the initial yield and payout might not be as high, as lucrative or with full mortality credits, it should offer a proper balance between the best cold hard mathematical *economics* and behavioural *psychology.* Finally, the last section of this chapter provides users with the definitive updated **R**-script `v3.0` for simulating the *modern tontine.*

5.9 Test Yourself

1. Please generate a tontine dividend dashboard in which the initial seed is changed from (the year) `1693` to `3961`. Discuss how (all) the results change, and why.
2. Please create, report and display a tontine dashboard (similar to Table 5.1), but for the aggregate amount of death benefits paid to those who die in the years $T = 1, 5, 10, 15, 20$. That is the `AGDEB[i,j]` matrix. Explain qualitatively what happens in the (later year's) columns and discuss the statistical distribution of those death benefit payouts. Do they seem normally distributed around the mean value? Discuss.
3. Assume that a (nefarious or misguided) tontine sponsor *claims* they will offer a death reimbursement covenant, but instead decides to pay the beneficiaries of the deceased a tontine dividend until the entire $f_0 :=$ \$100,000 is returned to the heirs. The death benefit is stretched out over time, instead of being paid out at once. They say: *Don't worry, we will make your family whole again, but slowly…* Please modify the core simulation `v2.0` to account for this feature, discuss whether it makes any difference at all on the tontine dividend payouts and carefully explain your results.

4. Please generate simulation results and create a table similar to the canonical dashboard, in which the reimbursement covenant is weakened in the following manner. If at the end of the year the aggregate value of the decumulation tontine fund is *less than* the original investment minus dividends received, those who died during that year receive the lower of the two values. So, for example, consider the following scenario. Tontine dividends of exactly \$7000 are received for 3 years, and in the 4th year the investor dies. Under the normal covenant, the beneficiaries of the deceased should receive a death benefit of \$100,000 minus the \$21,000 already received, which is: \$79,000 at the end of year 4. But imagine that the total fund value happens to only be worth \$60 million, perhaps due to a (very) bad year in the markets, and there are 800 survivors at the end of year #4. This implies that the notional value of the fund at the end of the 4th year is \$75,000 per live survivor. One can think of this as the reserve per person. But, this is also \$4000 less than the death benefit promised under a *strong* covenant. Well then, under the *weak* one proposed here, it would be reduced to \$75,000. Obviously this situation is somewhat artificial, but nevertheless please generate dashboard results under this particular design *and* report the distribution of the number of people—from an original pool of 1000 investors—who die and do **not** get their entire money back. Discuss the qualitative impact of fund values and tontine dividends, when initially `RTLIA` is used to set payouts but then weakened when someone actually dies.
5. Modify the `v3` script to allow new members to join the tontine in the first 15 years. You can do this by adding `LAPSE` to our `GLIVE` values (instead of subtracting `LAPSE`). When a new person joins during the first 15 years, they pay a lower `f0` into the fund, their initial investment is reduced by 30% of `TCPAY` (30% of the dividends they missed out on by joining late). After 15 years, we no longer allow people to join. Also note that we do not allow people to lapse in this specific fund. Find the median tontine dividend for this fund.

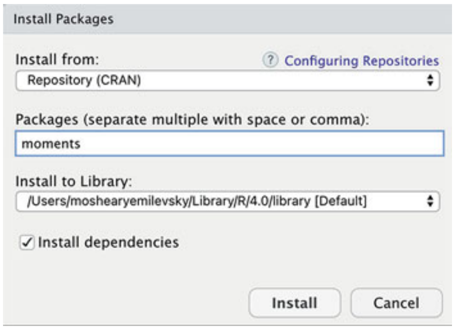
Goodbye LogNormal Distribution 6

The R-scripts presented and developed in the prior chapters assumed an investment return generating process that is both static and LogNormal, via the basic `rnorm()` function. Indeed, had this work been done in the 1970s, or perhaps the 1670s when national (versus *natural*) tontines were first launched, that might have been sufficient. But in the early twenty-first century, being normal doesn't cut it and mean with standard deviations aren't enough. In this chapter I explain (i) why that assumption is problematic and (ii) how to implement more robust models of portfolio investment returns. I will begin by returning to higher moments, such as skewness and kurtosis and their important role in testing data for normality. I then examine historical returns from the stock market during the prior half century and discuss how they deviate from normality. The chapter concludes by using a technique called *statistical bootstrapping* to simulate *modern tontine* dividends as an alternative construction method for the critical portfolio return matrix `PORET`. Overall the pedagogical objective is to help readers answer the basic question: *Does all this complexity make a difference?*

6.1 Statement of the Historical Problem

As noted above, the forward-looking simulations presented in the last few chapters have been predicated on LogNormal investment returns; that is that the logarithm of one-plus investment returns is normally distributed. That was occasionally referred to as the *continuously compounded* investment return (CC-IR), which is taken as Normal. In **R** they were simulated using the command `rnorm(TH,EXR,SDR)`, where `TH` was the number of years (e.g. 30 years), `EXR` was the mean return (e.g. 4% per year), and `SDR` was the standard deviation (e.g. 3% per year). That assumption, which is also labelled *Gaussian* or just bell curving should be familiar to readers from portfolio or option pricing theory. It states that future market investment returns can be described fully and exclusively in terms of their mean and standard deviation,

M. A. Milevsky, *How to Build a Modern Tontine*, Future of Business and Finance,
https://doi.org/10.1007/978-3-031-00928-0_6


Install Packages

Install from: Repository (CRAN)

Packages (separate multiple with space or comma): moments

Install to Library: /Users/moshearyemilevsky/Library/R/4.0/library [Default]

✓ Install dependencies

Fig. 6.1 How to install (missing) packages in R

a.k.a. the first two moments. Alas, bell curves are certainly convenient to work with, but they are a fiction in markets.

The facts are that historical (log) investment returns aren't really normally distributed, especially over short horizons such as trading days, weeks and even months. Just as importantly, historical (log) investment returns aren't independent and are somewhat serially correlated with each other. In reality they are not independent from month-to-month or year-to-year, which of course stands in contrast to generating *TH* independent and identically distributed (i.i.d.) random variables.

Now, this really isn't news to anyone who has worked with historical (and especially high frequency) stock return data. Moreover, this certainly isn't the proper place to review the voluminous scholarly literature on (i) the reasons they deviate from normality, and (ii) the many alternative non-normal models. Rather the point of this chapter is much more limited. I would like to show and explain how real-world historical (stock) returns have deviated from the normality, suggest some possible fixes and alternative simulation approach and conclude by discussing how this might impact the distribution of *modern tontine* dividends.

6.2 Measuring Skewness and Kurtosis

Let's go back to basics. I'll begin by simulating monthly (instead of annual) investment returns using the usual `rnorm(.)` method and display some summary statistics of those artificial returns. The plan is to compare those (artificial, laboratory grown) numbers to actual historical returns from the SP500 total return; the most widely known and quoted stock index in the world. To do this higher-moment analysis, you might want to install a new package into the **R-studio** environment, which computes statistical moments. Install via the command `install.packages("moments")`.

Depending on your **R** studio environment, or the software version you are using and perhaps even your operating system, you might have to tinker a bit and Fig. 6.1 is an alternative way to load this up (although don't use my directory!). You may also need to run the command `library(moments)` before you start your script.

In the end though, once the warnings lights have stopped flashing and the proper boxes have been ticked, you should get results that look like this next script. If you get error messages for the `skewness` and the `kurtosis` command, go back and try loading the libraries again. Alternatively, you can compute those two manually (without the moment packages) based on the formulas in Chap. 2, which I'll revisit in a moment.

```
set.seed(1693)
rv<-rnorm(612,0.0085,0.044)
mean(rv)
> 0.007751903
sd(rv)
> 0.04484798
median(rv)
> 0.00664067
skewness(rv)
> 0.04799061
kurtosis(rv)
> 2.607695
```

Note that I have simulated 612 monthly (log) investment returns, where the theoretical and assumed mean is 85 basis points (a.k.a. 0.85% per month) and the assumed standard deviation is 4.4% (per month). The rationale for these two very specific and rather odd numbers will become evident over the next few pages. But the first thing to notice is that if you (only) simulate 612 months, which is 51 years of returns, the *sample* mean can end up being different from the *theoretical* mean. In the above case, the sample mean was (only) 78 basis points and the sample standard deviation was closer to 4.5%. Neither of these differences should be cause for alarm, and generating this small sample with a different seed will obviously change the numbers. Just as importantly, the sample *median* is 66 basis points, which is 11 basis points lower than the *mean*, and well-within (small) sample variations.

Nevertheless, notice how the sample mean is just slightly higher than the sample median. This is a telltale sign of things to come and is related to the coefficient of skewness and kurtosis to which I would like to now direct readers attention. The formal and mathematical definition of those two variables was briefly given back in Chap. 2, but recall that for the normal distribution the coefficients should be zero (skewness) and 3 (kurtosis), respectively. In the above small sample the skewness was slightly higher than zero—consistent with a mean higher than a median—and the kurtosis was slightly lower than 3. Once again, these (small sample) results shouldn't be alarming or indicate any violation of normality. In fact, if you rerun or generate with `set.seed(1000)`, the sample skewness is −0.04792736 (on the other side of zero) and the sample kurtosis is 2.904078 (getting closer to 3). That's noise for you.

For those who are having difficulty using the built-in functions for higher moments, they can also be constructed and computed manually. The following script should replicate the skewness values:

```
set.seed(1693)
rv<-rnorm(612,0.0085,0.044)
# Sample Mean
mrv<-sum(rv)/(length(rv))
# Third Cumulant Defined
m3<-sum((rv-mrv)^3)/length(rv)
# Second Cumulant Defined
m2<-(sum((rv-mrv)^2)/(length(rv)))
# Fisher's Coefficient of Skewness
m3/m2^(3/2)
> 0.04799061
skewness(rv)
> 0.04799061
```

A similar script can be used to compute and confirm the kurtosis function values, except that instead of scaling the **third** cumulant by the **second** cumulant to the power of 3/2, the **fourth** cumulant is scaled by the (raw vs. sample) standard deviation to the power of four. The point of all this is to provide readers with a deeper understanding of the skewness and kurtosis values, what they represent and when they deviate too much from what is expected.

```
set.seed(1693)
rv<-rnorm(612,0.0085,0.044)
# Sample Mean
mrv<-sum(rv)/(length(rv))
# Fourth Cumulant Defined
m4<-sum((rv-mrv)^4)/length(rv)
# Raw Standard Deviation Defined
sdrv<-sqrt(sum((rv-mrv)^2)/(length(rv)))
# Coefficient of Kurtosis
m4/sdrv^4
> 2.607695
kurtosis(rv)
> 2.607695
```

In fact, while on this minor statistical digression I shall take this opportunity to remind readers that the `sd(.)` function built into **R** computes the *sample* standard deviation, not the above-noted (and christened) *raw* standard deviation. The difference between them will be well-known to readers from basic statistics, but I'll re-iterate here that the *sample* estimate divides all sums by $(N - 1)$ and what I called *raw* divides by the original N. The following script should clarify any confusion in this matter. In particular notice and pay attention to the first few digits after the decimal sign, which in both cases represents a volatility of 4.5% per month. Indeed, the differences are small (to an empiricist).

```
set.seed(1693)
rv<-rnorm(612,0.0085,0.044)
# Sample Mean
mrv<-sum(rv)/(length(rv))
```

```
# Raw Standard Deviation
sqrt(sum((rv-mrv)^2)/(length(rv)))
> 0.04481132
# Sample Standard Deviation
sqrt(sum((rv-mrv)^2)/(length(rv)-1))
> 0.04484798
# As computed by R.
sd(rv)
> 0.04484798
```

6.3 Continuously Compounded vs. Effective Annual

This might be a good point to digress just a bit and revisit the statistical distinction between the *effective annual* return as modelled and captured in the `PORET[i,j]` matrix, versus the *continuously compounded* return, which is computed via `log(1+PORET[i,j])`. During the last few pages and chapters, I might have sloppily used the term *returns* when referring to either of these two variables, even though they are technically different from each other. One is an effective annual rate (EAR), and one is a type of annual percentage rate (APR).

Recall that the basic simulation script for generating `PORET[i,j]` begins with a normally distributed random vector with mean `EXR` and standard deviation `SDR`, a.k.a. the *volatility*. The number *e*, or approximately 2.718, as the base raised to the power of that vector using the built-in `exp(.)` function. Finally, the numerical value of 1 is subtracted from that exponentiated value to convert the entire vector into an effective annual rate. My point here is that the exponentiation process will distort the higher moments that I introduced and explained in the prior paragraph, and especially the skewness. Here is a detailed numerical example that should help illuminate this insight.

```
set.seed(1693)
rv0<-rnorm(100000,0.04,0.06)
rv1<-exp(rv0)-1
# Compare Means
mean(rv0)
> 0.03991409
mean(rv1)
> 0.04258999
# Compare Deviations
sd(rv0)
> 0.0599022
sd(rv1)
> 0.0624973
# Compare Skewness
skewness(rv0)
> -0.005790745
skewness(rv1)
> 0.1724074
# Compare Kurtosis
```

```
kurtosis(rv0)
> 2.9789
kurtosis(rv1)
> 3.033556
```

I have generated 100,000 numbers from a normally distributed random variable with a theoretical mean of 4% and a standard deviation of 6%. Of course, the sample statistics will differ from the theoretical values, and especially after exponentiation to convert from *continuously compounded* to *effective annual*. By construction the baseline `rv0` is normally distributed, but `rv1` is created by exponentiating `rv0` and then subtracting one.

Per the results, which of course depend on the seed, the mean of the 100,000 numbers in the *continuously compounded* investment return (CC-IR) vector `rv0` is indeed very close to 4%, but the mean of the *effective annual* return vector is higher than 4%, as indeed it should be. Also, the standard deviation of the *continuously compounded* return is close to 6%, but for the *effective annual* return it's higher than 6%, once again as it should be. More importantly, the skewness of the *continuously compounded* return is slightly negative but zero for two digits after the decimal point, again consistent with the symmetry (and zero theoretical skewness) of the normal distribution. But, when the `rv0` vector is exponentiated to create the `rv1` vector, which recall is the basis for the `PORET[i,j]` matrix, raising *e* to the power of these values, induces statistical skewness. This is important to remember. Skewness in *effective annual* investment returns is **not** a sign of abnormality. Finally, the kurtosis of the *effective annual* return `rv1` is just slightly higher than the corresponding value for the *continuously compounded* return `rv0`, although both are quite close to the anticipated value of 3. No big change there.

The important takeaway (or reminder) is threefold. First, the *effective annual* investment return as captured by the `PORET[i,j]` matrix and the foundation for the simulation algorithm will have a small amount of skewness due to the exponentiation, even if the underlying return generating process was perfectly normal. Second, if you (blindly) compute the sample standard deviation of any `PORET[i,j]` matrix—regardless of how it was obtained—it will likely be just a tad higher than the standard deviation `SDR` used to generate the underlying returns, and what I have called *volatility.* Third, and just as importantly, if-and-when I happen to be sloppy and use the term *expected return* or *standard deviation* without specifying whether I'm referring to the *continuously compounded* `rv0` or the *effective annual* `rv1`, I should apologize in advance. But, more often than not I'll be referring to the former `rv0` and not the latter `rv1`. Or, using the language or **R**, it's the second and third argument in the `rnorm(.)` function. Go back to Chap. 2 for a refresher.

6.4 Quantile Plots of Investment Returns

Back to our main objective, another perspective on the normality (or lack thereof) of a particular dataset is obtained by generating and creating a simple quantile plot against the normal distribution. This process compares the number of data points within different quantile ranges or bins relative to *what they should have been* if they were normally distributed. This is a very well-known procedure for quickly visualizing and summarizing the distribution of the data.

Figure 6.2 displays a quantile plot of the normally distributed `rv` data using the `qqnorm(rv)` and then `qqline(rv)` command, which adds the diagonal line. What this picture is telling us by virtue of the points falling almost perfectly on the diagonal is that our sample (612 normal numbers generated using the 1693 seed) appears normally distributed. Ok, yes, in the lower-left corner a few data points are above the diagonal and in the upper-right corner a few points are under the diagonal, but this is consistent with the sample kurtosis value being under (the theoretical, perfect) 3. And for those readers who now worry about the accuracy of `rnorm(.)`, if you want to convince yourself **R**'s random number generator is functioning properly, generate 612,000 monthly returns and run the same procedure. It might take a while to sort and plot them, but the points will all rest nicely on the diagonal and barely a dot will deviate. So, I might be a bit cavalier with confidence intervals here, but the point is to (i) develop some quick and easy intuition for

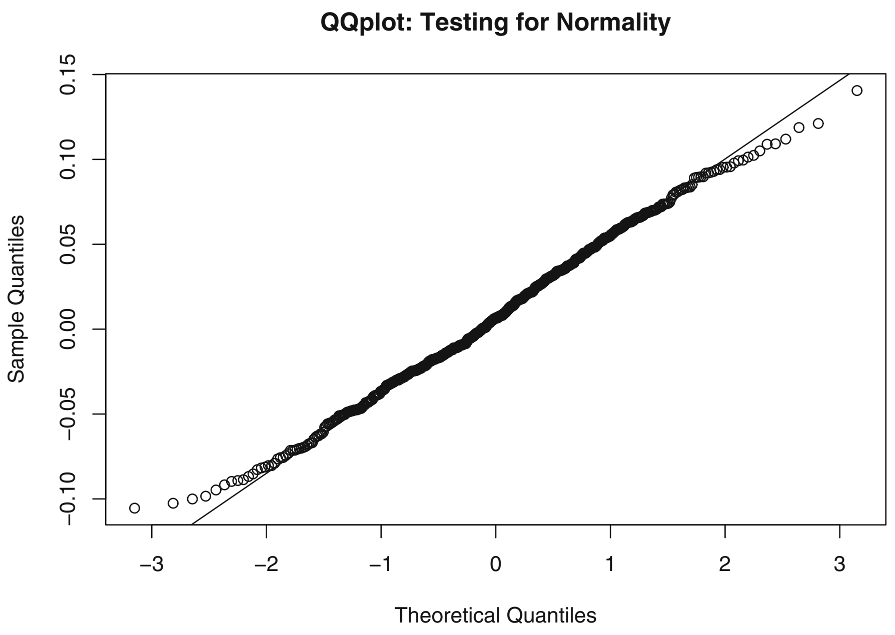


Fig. 6.2 Testing for (simulated) normality in (hypothetical) returns

detecting or not being able to detect normality and (ii) remembering that it very much depends on the sample size.

6.5 Serial Autocorrelation of Returns

Another aspect of the artificial and simulated data that I used to generate forward-looking scenarios for *modern tontine* dividends is that the underlying periodic investment returns have been assumed independent of each other over time. That is, a large (or small) return in one year is unlikely to follow a large (or small) return in the next year. The random draws from the investment urns are uncorrelated with each other. The best way to see the absence of correlation over time (lags) is to compute the autocorrelation value and plot them via the command `acf(rv)` in **R.**

Figure 6.3 plots the autocorrelation function over lagging periods ranging from 0 to 25 and draws a 95% confidence interval around the point estimate for this correlation value. You can extract individual (lag) values using the next script and syntax. The first number displayed is technically lag zero, that is the correlation of the investment return with itself, which is obviously a perfect one. The second number displayed is the correlation of a particular month's return, with the prior month's return. So, for example, a correlation value greater than zero would indicate that positive (negative) months are followed by positive (negative) months, and vice versa. A number less than zero would indicate investment (and momentum) reversals. If there is no autocorrelation at all, they should be (statistically) zero.

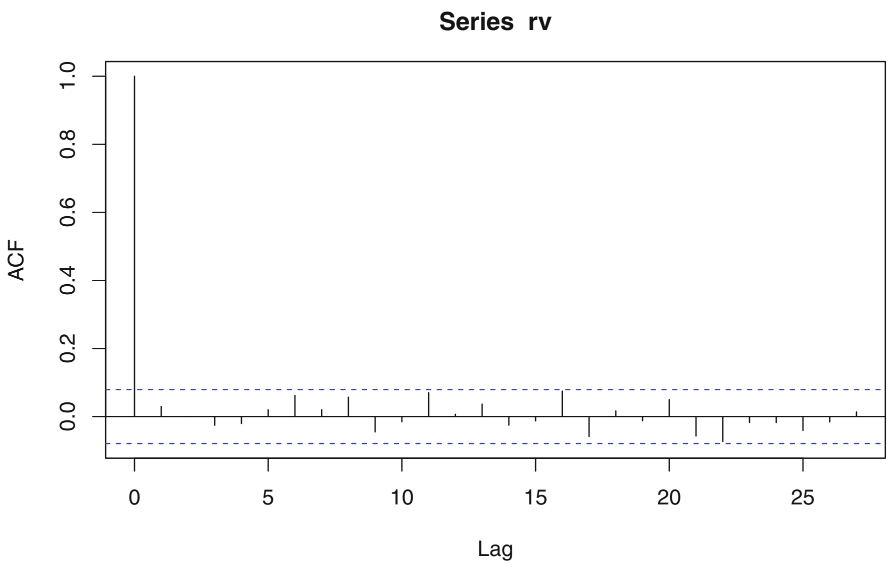


Fig. 6.3 The autocorrelation function: completely random returns

```
acf(rv)$acf[1:5]
> 1.000000  0.029472  0.000302 -0.025429 -0.020377
```

Looking at the above numbers, while the correlations for lag one and two are positive, and for lags three and four are negative, they are statistically indistinguishable from zero as per the confidence interval bands noted in Fig. 6.3. In fact what we have done is confirmed something we already knew by construction. These 612 monthly numbers were generated independently and identically distributed. Ok, let's see what happens when we filter *real world* investment returns thru the lens of the above-noted functions.

6.6 The SP500 Total Return

As part of the (online) material, on my website www.MosheMilevsky.com I have included a CSV file that contains realized historical monthly returns for the SP500 index, including dividends from January 1970 to the end of December 2020. This time series is also known as the SP500 Total Return, to differentiate it from the Index that does not include dividends. Those 51 years of historical returns are exactly 612 months, which might help explain the origins of that rather odd number earlier in the chapter. In the next script I import those numbers into **R**, which is something we have not yet done in this book, and I store those numbers in a vector called `SP500TR`. Simple importing can be done with a menu command in the graphical **R-studio** environment as well. In fact, the exact syntax you must use depends on where (on your desktop) you have downloaded and stored the CSV file. You also might want to ensure there are no extraneous entries or cells in the downloaded CSV. All these minor irritants can create error messages. Also, every time you see a + in the script below, it means that this line should be appended to the previous line.

```
library(readr)
SP500TR <- read_csv("~/SP500TR.csv",
+     col_types = cols(MONTH = col_date(format = "%m/%d/%Y"),
+         RETURN = col_number()))
```

Next, once you have `SP500TR` saved in memory, that is loaded into your environment, a simple calculation of (log one plus) mean and standard deviation should yield the following two numbers, which should further help clear up the rationale from my rather odd selection of parameters earlier in this chapter.

```
mean(log(1+SP500TR$RETURN))
> 0.008501958
sd(log(1+SP500TR$RETURN))
> 0.04436831
```

These are two rather important numbers. The historical average monthly (log) price return of all companies in the SP500 index, including all their dividends, has been about 85 basis points, or 0.85%. If you multiply that number by 12, which is what you should do with CC-IR numbers, then annual (log return) value is 10.2%, which is an astonishingly high rate of return that is unlikely to be repeated in the next half century. The basis for this phenomenal growth, and why prior chapters have used (much) more conservative numbers closer to 5%, is partially because of the dramatic and unprecedented decline in interest rates over the last half century. Now, my point here isn't to argue about forward-looking investment return assumptions (are you *bullish* or *bearish?*), but rather to focus on the higher moments. Moving right along, compute the skewness and kurtosis of the 612 historical monthly returns and compare them to and against the values for the artificial returns. Remember, `rv` and the SP500TR share the first two (log) moments, but what about the 3rd and 4th moments?

```
skewness(log(1+SP500TR$RETURN))
> -0.6962339
kurtosis(log(1+SP500TR$RETURN))
> 5.503288
```

The results are rather startling. The historical (realized, actual) skewness is extremely negative at (−0.696). The kurtosis is 5.5 and also much higher than the normally required 3. Now, to be precise I should also report a confidence internal on these two numbers—which would necessitate explaining how to compute those intervals using the sampling distribution—all which would take me even farther away from *modern tontines*. My point here is to make one simple and very important point, historical (log one plus) returns of the widest and most quoted stock index in the world aren't normally distributed. And, if the negative skewness and excess kurtosis don't convince you, then perhaps the QQplot will help. For the record, here is the precise **R** script to create the QQplot.

```
rv_history<-log(1+SP500TR$RETURN)
qqnorm(rv_history,main="QQplot: Is the SP500TR Normal?")
qqline(rv_history)
```

Figure 6.4 displays the earlier explained QQplot, but using the historical monthly SP500 total returns. Readers should quickly and easily be able to discern the excess kurtosis (standardized 4th moment) as evidenced by the many data points or outliers that are under the diagonal on the left-hand side of the figure (a.k.a. *fat left tails*) and the somewhat smaller number of data points above the diagonal on the right-hand side. Bottom line from all of this: *history isn't normal.*

So, while the prior discussion should convince you of the need to incorporate some (negative) skewness and (positive) excess kurtosis in forward-looking simulations, another issue I would like to address is serial correlation. Is there any persistence in the monthly (or annual) returns of the SPR500TR? Recall that a few pages ago I used the `acf(.)` command to display the autocorrelation function in **R.** Using the randomly simulated (612) investment returns, those serial correlation values were within the 95% confidence interval of zero—which is exactly what

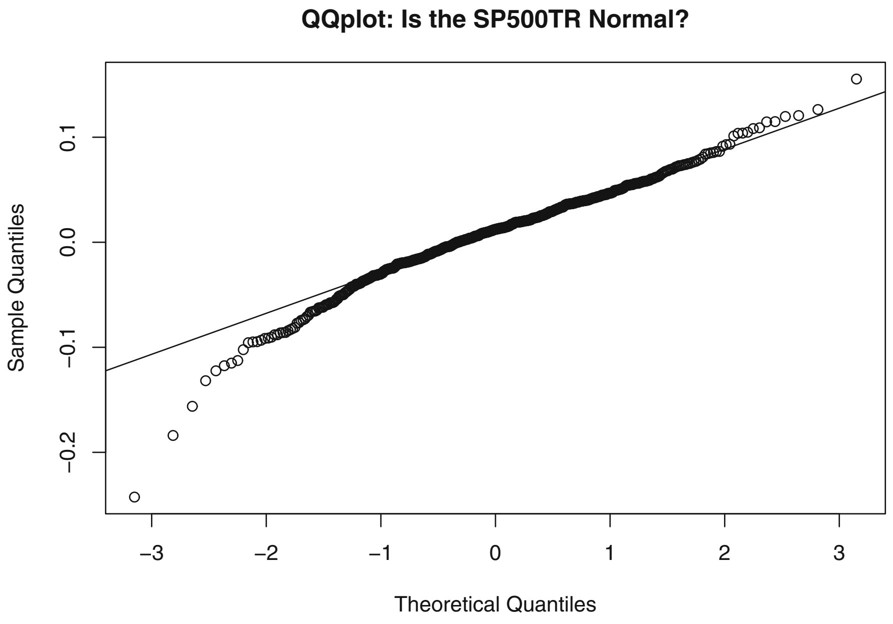


Fig. 6.4 Testing for historical normality in SP500 total returns

you would anticipate based on the underlying generating process. But what about the total monthly returns from the SP500 index over the last half century? Do they display any serial correlation? Do broad stock indices continue to go up (or down) after they have gone up (or down)? Is there (negative) momentum in monthly returns?

Interestingly enough it turns out that to a first order of approximation the original *independence* assumption isn't that bad and Fig. 6.5 provides the visual evidence. In particular, if you compute `acf(log(1+SP500TR$RETURN))$acf[2]`, which is the autocorrelation using a lag of one month, the resulting number is indeed positive with a point estimate of 0.0365. Remember, that should be interpreted as a correlation between this month and last month. But technically, one can't reject the null hypothesis that its value is indeed zero—given that it falls within the confidence intervals noted in Fig. 6.5.

Now, I must emphasize and remind readers that this statement or modelling position, namely that *"You can assume the serial correlation is zero"* is rather controversial in the financial econometrics literature and is partially driven by the fact the index averages returns of (close to) 500 stocks. Individual stocks or sub-indices or portfolios that use derivatives will behave differently and goes beyond the range of this chapter. Also, if we examined daily or hourly investment returns, the correlation patterns would deviate and might not fall within the acceptable range of zero. But this is key. Remember that modern tontines are **not** designed for continuous time trading. Rather, the assets are assumed to be invested in simple

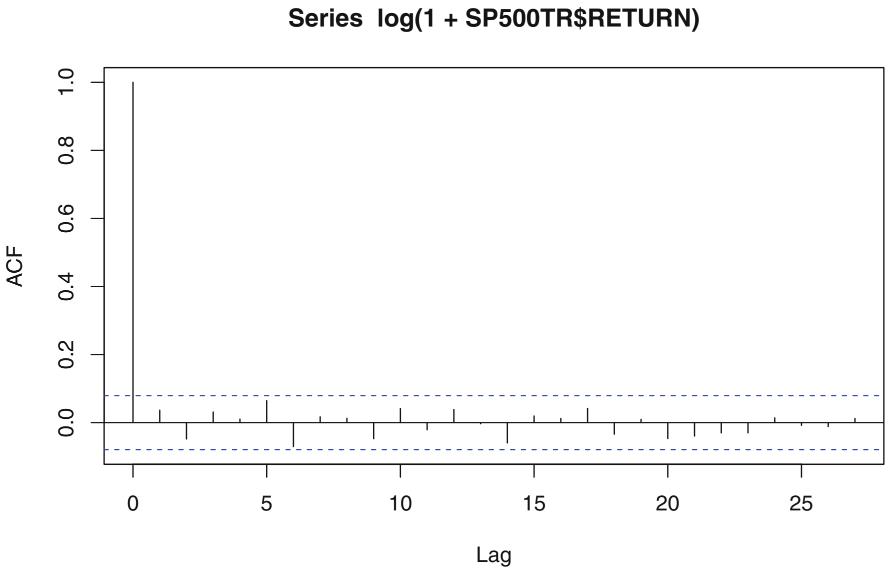


Fig. 6.5 The autocorrelation function: SP500 total returns

low-risk and linear instruments. The dividends are re-calibrated yearly, or perhaps quarterly at most. So, incorporating serial correlations in our (basic) forecasting model is less of an imperative. But, to be clear, the third and fourth moments can't be as easily ignored or dismissed.

6.7 Path Forward for Deviations from LogNormality

The natural question arises then, how does this (negative skewness, excess kurtosis) affect projected tontine dividends? The discrepancy in the higher (3rd and 4th) moments will obviously impact the statistical distribution of returns and the `PORET[i,j]` matrix. But how exactly? Is it economically material? The only way to actually find out is to simulate tontine values in which the first two moments—a.k.a mean and variance—remain the same, but the higher moments are modified. What I will now describe is a procedure that can be used to *replace* the `PORET[i,j]` matrix, as opposed to a completely different *modern tontine* simulation procedure. This will limit the surgery (in the code) and the overall work involved. There are two different ways to do this, and I will describe both. The first one is rather basic, and that is to use an external program or economic forecasting engine to create the 10,000 paths required by `PORET[i,j]`. From a business management point of view, this implies having someone other than the (tontine) quant who is designing the algorithm take responsibility for generating those asset return scenarios. I have nothing more to say about that first approach, other than to remind readers that `PORET[i,j]` should reflect an asset allocation that is suitable for the clientele investing in the *modern tontine*. The second approach is more organic, the *historical bootstrap*, and involves using a new function in **R.**

6.8 Basic Historical Bootstrap

I will start by explaining how to use the `sample(.)` command within **R**, which is a rather powerful tool for simulating forward-looking investment returns. The following script *samples* 4 numbers from the entire vector of 612 numbers (monthly returns) stored in the `SP500TR$RETURN` dataset. For variety, I will use another seed to generate this sample.

```
set.seed(1)
sample(SP500TR$RETURN,4)
> 0.0294 -0.0601  0.0876 -0.0364
```

The ubiquitous `set.seed` command ensures that everyone gets the same four numbers, no different than setting the seed before generating random numbers. In the above case, **R** selected the four numbers listed in the results. The first (random) number was a gain of 2.94% in the month, the second was a loss of 6.01% in a month, the third was a gain of 8.76%, and the final sample monthly return was a loss of 3.64%. Now, to look clever, I could have also sampled from the actual months themselves that are part of the `SP500TR` dataset. In this case the syntax would have been:

```
set.seed(1)
sample(SP500TR$MONTH,4)
> "1980-09-30" "2012-05-31" "2009-03-31" "1994-11-30"
```

Notice that (because the random seed was the same) the results were the same. Indeed, the monthly returns for the periods ending in the above four calendar dates were the same ones as listed above. Type in the following command and out pops the corresponding return:

```
SP500TR$RETURN[SP500TR$MONTH=="1980-09-30"]
> 0.0294
```

The next step is to use this method or command to sample a total of 30 years' worth of investment returns from the SP500TR dataset, to then create one possible path for the tontine fund and then tontine dividends. The 30 years is 360 months, and we want a total of 10,000 such samples, so there is a bit of work that must be done before we can replace the old `PORET[i,j]` matrix with a new one. First, since we will be sampling 360 months (from a total of 612) we probably want to sample-with-replacement and allow for multiple picks and repetitions of a given month. Generally the arguments for sampling with (or without) replacement are rather deep and philosophical in the context of forward-looking investment returns, but once again this isn't the venue for such debates. The key is that we must use a slightly modified version of the `sample(.)` command in **R.** Here is the next step on the path to creating the modified `PORET[i,j]` matrix. I will generate one possible path for the 30 years, using our original familiar 1693 seed.

```
set.seed(1693)
path<-sample(SP500TR$RETURN,360,replace = TRUE)
> summary(path)
 Min.      1st Qu.    Median    Mean   3rd Qu.     Max.
-0.215400 -0.017200 0.009600 0.007188 0.037675 0.110400
```

Notice that is quite the large and wide range of monthly returns. The worst month was a loss of 21.5% (which can be traced to October 1987). How exactly do I know this? Well, I can reverse the query noted above and ask **R** for the month ending date in which that return was observed.

```
SP500TR$MONTH[SP500TR$RETURN=="-0.2154"]
> "1987-10-30"
```

I note this (again) so that readers appreciate how this particular simulation methodology—that is sampling from historical returns—is simply another way of picking and scrambling random months, and assuming the investment returns from those months will repeat themselves. For those who are curious, the best of the 360 selected months corresponds to the month of August 1984.

```
SP500TR$MONTH[SP500TR$RETURN=="0.1104"]
> "1984-08-31"
```

Before we proceed to stitching together our many single `path` values into a large matrix to replace `PORET[i,j]`, it might be worthwhile to examine the statistical properties of the (log) one-plus investment return of this one `path` just created. The following script that computes the four moments should be familiar by now but is yet another check on your results. Confirm these numbers as well.

```
mean(log(1+path))
> 0.006101569
sd(log(1+path))
> 0.04651308
skewness(log(1+path))
> -1.152608
kurtosis(log(1+path))
> 6.991457
```

Notice how this particular collection of 360 values—from a Universe of 612 numbers, albeit with replacement—resulted in a lower mean return, slightly higher volatility, more negative skewness and a higher kurtosis, compared to the entire period 1970 to 2020. This is (very) non-normal, but also nothing more than random luck.

6.9 Monthly to Annual

The next step in this process—once we generate one path of 360 months—is to convert that into a path of 30 annual returns and then repeat that process again (and again) so that we create 10,000 such paths for our *modern tontine* simulation code. The process of converting from monthly to annual can be done in a variety of manners, but here is one possible **R**-script that will take you from 360 months to 30 years.

```
set.seed(1693)
# Generate Vector of 360 Monthly Investment Returns.
vector1<-rnorm(360,0.01,0.05)
# Create Cumulative Investment Values.
vector2<-cumprod(1+vector1)
# Create Vector of Annual Values.
vector3<-vector2[seq(0, length(vector2), 12)]
vector3<-append(vector3,1,0)
# Extract the Annual Returns.
vector4<-vector3[-1]/vector3[-length(vector3)]-1
round(vector4,6)
 [1]  0.073581  0.121747  0.411016  0.580719 -0.022159
 [6] -0.053635 -0.150397  0.089884  0.254365  0.222345
[11]  0.411773 -0.183681  0.102055  0.117695  0.226608
[16]  0.319082  0.128694  0.015303  0.064975  0.399443
[21] -0.022303  0.084906  0.411599  0.035348  0.005110
[26] -0.274844  0.030719  0.176580  0.269325  0.343089
```

What the (rather elaborate and cumbersome) script is doing in four stages, is generating a vector of 360 (random) monthly returns and then converting them to the 30 annual returns displayed and listed above. It does this by (first) cumulating the monthly returns to create total returns, then (second) selecting the year-end values of those total returns, and (third) extracting those values and dividing by the prior year-end values to end with an actual return. The end result is 30 pristine numbers. I will now do the exact same thing within a formal function (without all the explanations), which generates monthly samples and then converts the `path` into annual numbers.

```
ANPATH<-function(hist_month_data,TH){
path<-sample(hist_month_data,TH*12,replace = TRUE)
path2<-cumprod(1+path)
path3<-path2[seq(0, length(path2), 12)]
path3<-append(path3,1,0)
path4<-path3[-1]/path3[-length(path3)]-1
path4}
```

I'm performing two separate tricks or calculations within this `ANPATH` function. First, I am sampling from the historical monthly return data that is captured in the first argument of this function. In particular, since `TH` is measured in years, I'm sampling `TH *12` months with replacement. That is the first operational line in the script. Then, once that path of monthly values has been created, the remainder of the script converts from monthly to annual, all as explained above. Here is one particular run of the `ANPATH(.)` function.

```
set.seed(1693)
rv<-ANPATH(SP500TR$RETURN,30)
> summary(rv)
    Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
-0.30730 -0.01739  0.11606  0.09000  0.20254  0.42745
prod(1+rv)
> 8.994063
```

Notice how for this sample, the worst total annual return (over 30 years) was a loss of 30.7%, and the best annual return was a gain of 42.7%. The (arithmetic) mean of those 30 years was 9.0%, and the inter-quartile range ranged from a loss of 1.7% to gain of 20.3%. To repeat, this was one annualized sample path for 30 years.

The final step then is to place this particular function into a loop and create the 10,000 paths to replace `PORET[i,j]`. Here is that final piece, of which the loop is lifted directly from the script.

```
TH<-30; N<-10000
set.seed(1693)
PORET<-matrix(nrow=N,ncol=TH)
for (i in 1:N){
  PORET[i,]<-ANPATH(SP500TR$RETURN,TH)
  }
mean(log(1+PORET[,10]))
> 0.1018501
sd(log(1+PORET[,10]))
> 0.1528116
skewness(log(1+PORET[,10]))
> -0.2019967
kurtosis(log(1+PORET[,10]))
> 3.238949
```

So, at the end of this process we have simulated 10,000 paths of 30 annual returns by bootstrapping the SP500 total return over the last half century. The resulting mean log one-plus (a.k.a. continuously compounded) investment return **in year 10** was 10% per year, and the standard deviation (or realized volatility) in that year was 15% per year. Of course, the objective of this entire chapter or exercise was not those two moments, but the skewness of −0.202 and the kurtosis of 3.24, well beyond the normal. In the end we have a new type of `PORET[i,j]` matrix, one that resembles the historical SP500 total return index—with negative skewness and higher kurtosis—which will be used to generate tontine fund values.

6.10 How Do Higher Moments Affect Tontine Payouts?

Regardless of how you construct and populate forward-looking investment returns—the procedure described in the last few pages, or by importing your own scenarios—the final step and objective of this chapter is to investigate the impact of that (new) matrix on the evolution and stability of *modern tontine* dividends. At this point in the narrative, I'll refrain from copying and pasting the entire Monte Carlo simulation script yet again and will simply remind you to remove (or comment out) the two lines in which the `PORET[i,j]` matrix is defined in terms of the `rnorm(.)` function. Instead, replace that variable with the modified `PORET` numbers and then generate the *modern tontine* simulation code over a 30-year horizon with the usual Gompertz mortality parameters. The only remaining *tricky* and sticky point is to figure out what discount rate r to use. That determines the initial tontine dividends as well as the factor used to distribute cash in later years. Recall that when returns are simulated from a LogNormal distribution and the fund incurs no expenses or extra costs, the discount rate **is** the expected continuously compounded investment return (CC-IR).

But, when the `PORET` matrix is generated *exogenously*, we must manually compute the sample average and hope that using that as r stabilizes the dividends. For the `PORET[i,j]` matrix computed above, the arithmetic average was approximately 10%, which is what I'll use for the r value in this one simulation. The expectation is that like the LogNormal case using that value of r will assure a stable tontine dividend profile.

Table 6.1 displays the results from a simulation run using the historically bootstrapped `PORET[i,j]` matrix, implemented in the standardized tontine dashboard. The first item to notice when compared against dashboards from prior chapters, is that the median *modern tontine* dividend is much higher than in previously reported simulations. The initial payout of $12,550 per year far exceeds the $7074 estimated and displayed in the dashboard of Chap. 5. The reason for this almost doubling of the dividend is driven entirely by the (much) higher discount rate of 10% versus the

Table 6.1 Tontine dividend using bootstrap SP500 total returns; `set.seed(1693)`

Modern Tontine (MoTo) Fund: Simulation Dashboard					
Lifetime Income with Refundable Death Benefit					
Statistical	**DIVIDENDS: End of Year Number…**				
Outcome	$T = 1$	$T = 5$	$T = 10$	$T = 20$	$T = 30$
1 pct. (worst)	$12,550	$5,276	$3,232	$1,247	$0
25 pct.	$12,550	$9,940	$8,693	$7,179	$5,363
Median	$12,550	$12,619	$12,637	$12,475	$11,856
75 pct.	$12,550	$16,045	$18,205	$21,471	$24,650
99 pct. (best)	$12,550	$27,697	$42,803	$76,144	$143,983
St. Dev.	$0	$4,783	$8,271	$15,464	$30,633
Assumptions:	Historical SP500TR Bootstrap with: $r = 10.0\%$				
$N = 10000$	Gompertz: $x = 65$, $m = 90$ and $b = 10$.				
`TH=30`	Investors: $GL_0 = 1000$ with $f_0 = \$100{,}000$.				

4% used in Chap. 5. Now, it's not that I have changed my mind about a suitable discount rate for setting the initial tontine payout. In fact, the number 10% is woefully inappropriate and much too high. Rather, what's driving results and this choice of r, is that the underlying assets of the tontine fund are entirely invested in the (historical) SP500 index, which experienced a growth rate of 10% over time. I had no choice but to discount and value the embedded temporary life annuity at 10%, if I wanted stable tontine dividends when the assets are allocated to SP500TR.

Needless to say, it's highly unlikely that growth rate will persist going forward into the future. More importantly, the very wide dispersion in *modern tontine* dividends over time in Table 6.1 is yet another reason it's woefully inappropriate to allocate the entire funds into equity with a 15% annual volatility. More worrisome than the wide dispersion is that the worst case (1%) scenario after 30 years is no tontine dividends at all!

Again, the point isn't to suggest or even hint that a 100% allocation to the SP500 total return would be suitable, so as to increase initial payouts. This particular assumption within the context of defined benefit pension plan contributions has been quite costly for the actuaries! Rather, the main technical objective of this chapter was to examine how the negative skewness and excess kurtosis of (real world) stock returns affects *modern tontine* dividends. We are halfway there. To make that an apple-to-apples comparison I must now go back to the same basic model underlying the dashboard of Chap. 5, and assume *LogNormality* but with an expected return of 10% and a standard deviation of 15%. Note that the LogNormal assumption forces the skewness to be close to zero and the kurtosis to be close to 3, regardless of the assumed value of first two moments. Recall though that for the historically bootstrapped `PORET` matrix underling Table 6.1 the skewness (of the year 10 investment return, for example) was −0.20 and the kurtosis was 3.24.

Finally, Table 6.2 provides a dashboard summary and concludes this particular horse race. Again, the underlying `PORET` matrices share that same first two moments (at least approximately) but differ in their skewness and kurtosis. Can you see any differences in results between a parametric simulation (with LogNormal returns) and a non-parametric (with historical bootstrap returns)? Pay specific attention to the standard deviation of the dividends in year $T = 10$, as an example.

Alas, you would be quite correct if you concluded that the volatility or risk for the *modern tontine* dividends is just slightly higher—but not substantially so—when the underlying return generating process is *historically bootstrapped* instead of moment matched to the SP500 total return index. The median dividends are well within a few hundred dollars of each other—when comparing both dashboard tables against each other—although the 99% percentile at the long horizon appears to be a tad higher when returns are bootstrapped. In some sense, this is a manifestation of negative

Table 6.2 Tontine dividends using moment matching LogNormal returns; `set.seed(1693)`

Modern Tontine (MoTo) Fund: Simulation Dashboard Lifetime Income with Refundable Death Benefit					
Statistical	**DIVIDENDS: End of Year Number…**				
Outcome	$T = 1$	$T = 5$	$T = 10$	$T = 20$	$T = 30$
1 pct. (worst)	$12,550	$5,517	$3,316	$1,274	$240
25 pct.	$12,550	$9,912	$8,499	$7,037	$5,095
Median	$12,550	$12,486	$12,261	$11,960	$10,850
75 pct.	$12,550	$15,626	$17,665	$20,257	$22,485
99 pct. (best)	$12,550	$28,684	$40,990	$69,499	$129,557
St. Dev.	$0	$4,740	$7,931	$14,343	$29,199
Assumptions:	Financial: $r = 10.0\%$, $\nu = 10.0\%$, $\sigma = 15.0\%$				
$N = 10000$	Gompertz: $x = 65$, $m = 90$ and $b = 10$.				
`TH=30`	Investors: $GL_0 = 1000$ with $f_0 = \$100{,}000$.				

skewness and excess kurtosis in returns, although most of the blame for the wide range band sits squarely in the hands of mortality.

6.11 Conclusion: How Much Should We Worry?

To be very clear—and to conclude this chapter—I am not advocating that the underlying tontine fund assets be 100% allocated or linked to the SP500 total return, or any other broad-based equity index. Rather, what I have shown is that using a LogNormal model in which annual investment returns moment match history provides reasonably close results to the historically bootstrapped methodology. The unacceptably high spread or variability of the tontine dividends in later years is slightly higher in the bootstrapped case, but not enough to reject or discard the insights from and use of the LogNormal model. And, just as importantly, I'll leave this as a concluding thought. The assumed mean and standard deviation of forward-looking investment returns will have the greatest impact—and are most critical—for setting *modern tontine* dividends. It's absolutely essential to get those right, and unfortunately I don't have that crystal ball.

6.12 Test Yourself

1. Please generate a `PORET[i,j]` matrix in which 40% of the tontine fund assets are placed in a fund resembling the historical SP500 total return index, and the remaining 60% of the fund is invested in fixed income bonds that are normally distributed with a mean return of 2%, and volatility of 4%, per annum. There is no need to generate the entire *modern tontine* simulation, but please report the summary mean, standard deviation, skewness and kurtosis of the `PORET[i,j]` matrix in the tenth year of the fund.

2. Create a `PORET[i,j]` matrix in which investment returns are based entirely on the historical SP500 total return, but the annual returns are both floored and capped. The floor and cap are located at the upper and lower 15th percentile. What this means is that 15% of the *worst months* are not used or experienced by the fund, in exchange for giving up or sacrificing the 15% of *best months*. To be very clear, your bootstrap procedure should only use 100% or all of 612 months, but replace the extreme returns with their floored and capped values. Again, there is no need to simulate tontine dividends. Rather, report the summary statistics (a.k.a. moments) of this `PORET[i,j]` matrix in the 10th year.
3. The discussion of skewness and kurtosis has been devoted exclusively to the investment returns, but the same computation and summary statistics could be applied to the tontine dividends. Please generate the numbers underlying Tables 6.1 and 6.2 and compare the skewness and kurtosis of the *modern tontine* dividends in the 10th year of the fund. Note that the 10th year is rather arbitrary, but a horizon or time period must be fixed whenever the summary statistics are computed. Averaging all of the dividends or all of the returns would be mixing too many (calendar) apples and oranges.
4. The previous question explored the effect that skewness and kurtosis of the `PORET` matrix have on the skewness and kurtosis of `TONDV[,10]`. This question looks at how `PORET` volatility affects the skewness and kurtosis of `TONDV[,10]`. Generate a `PORET` matrix where volatility is 0, and the expected annual return and discount rate are 10%. Use this returns matrix to simulate modern tontine dividends and measure the skewness and kurtosis of the dividends in the 10th year.

7 Squeezing the Most from Mortality

The R-scripts and numerical results in the prior chapters assumed that investors in the modern tontine live and die by the famous Benjamin Gompertz (1825) model of mortality. And, although that *natural law* is about to celebrate its bicentennial and has withstood the test of time and longevity, there is a growing body of (controversial) evidence the law is violated at advanced ages. More importantly, practising real-world actuaries are accustomed to using a vector of discrete q_x mortality tables and many are sceptical the force of mortality can be treated as smooth and differentiable. (Even more distressing, some have never heard of Gompertz.) The good news is that almost any discrete mortality assumption can be easily and quickly *plugged into* the simulation R-scripts presented in the earlier chapters—and that will be explained. Another important issue addressed in this chapter is how and when changes to mortality models themselves might affect the evolution and stability of tontine dividends. Essentially, the plan in this chapter is to kill people very differently, otherwise known as model risk. To start though, we won't kill anyone at all and I'll begin by introducing the non-tontine version of a *decumulation* fund.

7.1 Assumptions Versus Realizations

As noted above, the focus and purpose of this chapter is to measure and monitor the impact of different mortality *assumptions* and *realizations* on the long-term behaviour of the modern tontine fund and its dividends. To be more specific, if the sponsor believes investors will be healthier, dying later and living longer, the initial payout rate must be reduced in order to maintain tontine dividend stability. In our language, they should use a higher m value. Indeed, if an incorrect initial assumption is made about mortality and the initial payout rate is set too high, the realized tontine dividends will be forcefully adjusted downwards over time via the

M. A. Milevsky, *How to Build a Modern Tontine*, Future of Business and Finance,
https://doi.org/10.1007/978-3-031-00928-0_7

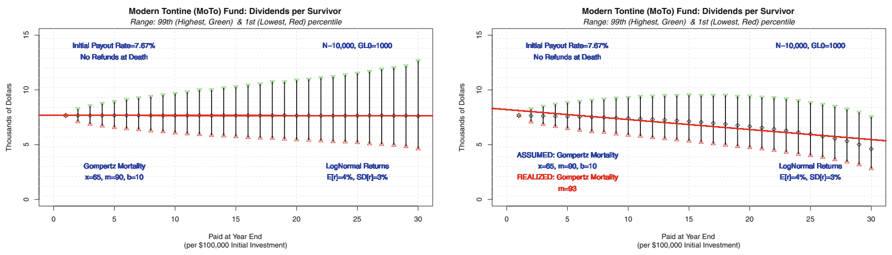


Fig. 7.1 Mortality models: assumed versus realized (98% C.I.)

thermostat mechanism. This is what I mean by the term initial *assumptions* versus actual *realizations*. The distinct between ex ante and ex post is critical.

Indeed, it's hard to over-emphasize the importance of "getting" the mortality and longevity parameters "just right", and the following graphical example should help illustrate what happens when wrong (or bad) assumptions are made. The left-hand panel in Fig. 7.1 displays the (by now standard) evolution of tontine dividends *assuming* that—and *realizing* a future in which—mortality evolves according to a Gompertz model with modal value $m = 90$ and dispersion value $b = 10$ years. Under a valuation (and expected return) of 4%, the tontine dividends propagate in a relatively symmetric manner over time with an expanding band of uncertainty caused by the diminishing number of survivors. The relatively (high) initial payout rate of $\kappa_1 = 7.67\%$ assumes no cash refund at death or any other covenants. This is the original `v1.0` introduced and explored many chapters ago and should be routine.

In contrast to the usual picture, the right-hand panel of Fig. 7.1 makes one small and minor change to the canonical R-script. Although the initial payout rate and the κ curve is computed based on the parameter: $m = 90$, the simulation of deaths via the R-script line: `rbinom(1,GLIVE[i,j-1],1-TPXG(x+j-1,1,93,b))` assumes that mortality rates are (lower) and driven by a modal value of $m = 93$. In other words, people die at a slower rate, there are more survivors than expected and tontine dividends must gradually be reduced relative to what was projected and promised.

Notice how the cone and envelope of uncertainty begins to trend downwards over time as more people (than expected) survive. While the total amount (numerator) paid out to survivors as a fraction of the fund value remains the same—determined by the vector of κ values—the number of survivors (denominator) is larger than anticipated and the tontine dividends shrink. Either way this is a simple example and illustration of what happens if life expectancy (or more precisely, the modal value of life) ends up being a mere 3 years greater than anticipated and mortality rates are systematically lower across the board. The death rate at age y was assumed to be: `(1-TPXG(y,1,90,b))`, when it should have been: `(1-TPXG(y,1,93,b))`. Mortality modelling is quite important to the life of a *modern tontine* fund, and you already heard something similar in Chap. 4.

7.2 No Mortality: Natural Decumulation

And yet, before we continue along the journey of investigating the sensitivity of the modern tontine to other mortality assumptions and forms, I would like to proceed with a hypothetical in which there is no mortality assumed or realized at all. In other words, I would like to examine the operations of the *anti-tontine fund* in which the initial capital and all market returns are returned to all investors (who never die) over the entire horizon of the fund. I'll call this the *Natural* Decumulation (NaDe) Fund to contrast it with the *Modern Tontine* (MoTo). Operationally this will allow beneficiaries or their estate to inherit shares of the *natural* decumulation fund and continue the same cash-flow stream. The point of this odd diversion is to investigate how a twin fund without any longevity risk sharing or pooling might behave, to then help contrast with its super-charged sibling, shedding a new unique light on mortality assumptions. Alas, it's not inconceivable that such a fund might be offered to help highlight the greater longevity and mortality credits embedded within its sister.

A quick-and-easy way to simulate a *natural decumulation* (NaDe) fund would be to use our existing R-script and replace the modal age at which people die, that is the Gompertz *m* parameter with a ridiculously high number (like a million years). That would certainly ensure that the *force of mortality* in the next few centuries is forced down to zero and nobody is *assumed* to die or *actually* dies. Although that fix would certainly do the trick, it would create a very large and unnecessary number of matrices, tables and variables. It would also slow down the code. After all, what's the point of having a `GDEAD` full of zeros, or `GLIVE` with the same people as there were on day one. Instead, I will take this opportunity to create a slimmed down R-script, one that allows us to compare and contrast *modern tontines* against *natural decumulation* funds. Here is the script.

```
set.seed(1693)
# Natural Decumulation Fund (non-tontine.)
RGOA<-function(g,v,N){(1-((1+g)^N)*(1+v)^(-N))/((v-g)/(1+g))}
TH<-30; N<-10000; EXR<-0.04; SDR<-0.03;
r<-0.04; f0<-100; GL0<-1000
kappa<-c()
for (i in 1:TH){kappa[i]<-1/RGOA(0,r,(TH-i+1))}
PORET<-matrix(nrow=N,ncol=TH)
DECFV<-matrix(nrow=N,ncol=TH)
DECDV<-matrix(nrow=N,ncol=TH)
for (i in 1:N){PORET[i,]<-exp(rnorm(TH,EXR,SDR))-1}
for (i in 1:N){
  DECDV[i,1]<-f0*kappa[1]
  DECFV[i,1]<-f0*GL0*(1+PORET[i,1])-DECDV[i,1]*GL0
  for (j in 2:TH){
    DECDV[i,j]<-DECFV[i,j-1]*kappa[j]/GL0
    DECFV[i,j]<-DECFV[i,j-1]*(1+PORET[i,j])-DECDV[i,j]*GL0}}
```

I'll now go through the R-script line-by-line and explain what each piece is doing. Many of the ingredients should be familiar by now. The very first line defines a new function RGOA, which is an abbreviation for regular growth ordinary annuity. It represents the present value of \$1, growing at the rate of g per period, received over N periods, where the discount or valuation rate is v. So, for example, `RGOA(0,0.04,30)` leads to a present value factor of: \$17.292, and the more important inverse of this number is 5.783%. What this effectively means is that an initial investment of \$100,000 is economically equivalent to a constant payment of \$5, 783 per year for 30 years. The present value of \$5, 783 annually for the next 30 years is equal to \$100,000. If this reminds you of the κ function and it's inverse the annuity factor, that is exactly my point. The payout rate of the non-tontine will be based on this function so when valuation rates are 4%, the initial payout rate from the *natural decumulation* fund will be 5.783%, versus the above-noted 7.67% for the *modern tontine*. The 189 basis point difference between these two numbers and payouts is due to: *mortality credits*. In some sense, the gap or spread—regardless of the parameters—highlights what tontine participants are getting in return for sacrificing their principal.

With that one line of R-script out of the way, we can move a bit faster down the remaining ones. The parameters are defined (with no mortality!) and then the κ vector is defined as the reciprocal of the `RGOA` function, instead of the Gompertz-based `TLIA` or `RTLIA` functions. The code names the same (old) `PORET` matrix to capture and simulate the future investment returns but defines new `DECFV` and `DECDV` to keep track of the natural decumulation fund value and the dividends, which should probably be called liquidation payouts or blended return of principal and investment gains. The word dividend is pushing the nomenclature a bit far. Nevertheless, the process of populating those two matrices is quite similar to its twin tontine sibling, except that the distributable quantity `DECFV[]*kappa[]` is divided by the same original number of investors `GL0`, not the shrinking `GLIVE[]` matrix.

For example, let's use a seed of `1693` and compute the year-20 mean and standard deviation of the modern tontine dividends using `v1.0` of the R-script and compare those with the mean and standard deviation of the non-tontine, a.k.a. *natural decumulation* fund. Use the familiar $m = 90, b = 10$, as well as $r = \nu = 4\%$ with a standard deviation $\sigma = 3\%$, and make certain you get these exact numbers, in the process confirming your (new and old) code is working correctly. Recall that in the later part of this book we have migrated towards a higher value of σ, to be able to afford (and justify) a higher portfolio expected return and discount rate. The mean tontine dividend in year 20 is \$7,750 per initial \$100,000 investment, and the standard deviation (scaled by the mean) is: `sd(TONDV[,20])/mean(TONDV[,20])`, which is: 14.76%. Alas, for the non-tontine fund, the mean dividend in year 20 is: `mean(DECDV[,20])=5.9115` thousand dollars, and the standard deviation is: `sd(DECDV[,20])/mean(DECDV[,20])`, or 14.12%, a much lower mean and a slightly lower standard deviation.

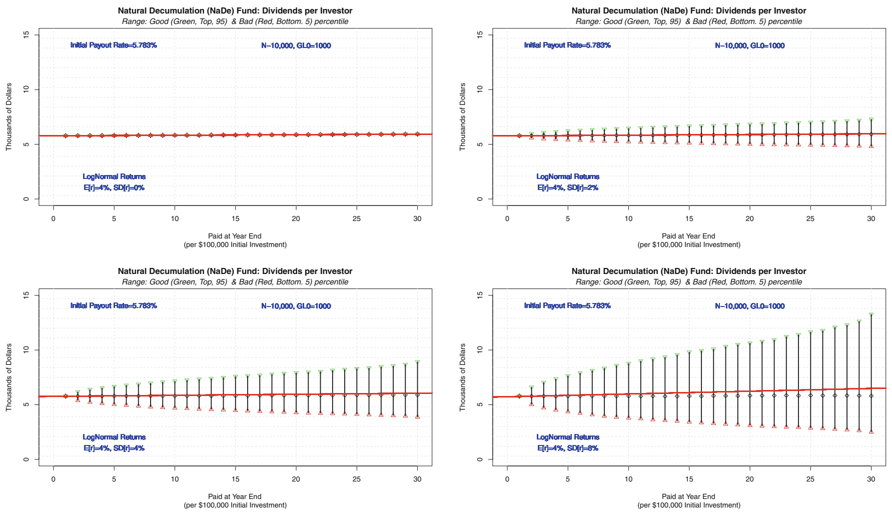


Fig. 7.2 Natural decumulation fund dividends: 0%, 2%, 4% and 8% vol

Finally, Fig. 7.2 displays the non-tontine dividends over time assuming the usual stochastic (LogNormal) investment returns, but with no other source of uncertainty; certainly no deaths or lapsation. Rather, every single year the entire value of the fund is effectively *divided* by the relevant annuity factor and distributed to all investors. Although I have displayed the range of payouts under four different volatility levels, one thing that does seem rather clear is that *even* without any mortality uncertainty—or any relevance of death—there is an increased variability at advanced ages and periods. You can't blame a small surviving pool for that. That band is due to investment returns and markets. To wrap this all up, at the very end of the script you could also add the usual regression to test stability of the *average* payouts over time.

```
mtd<-c(); t<-1:TH
for (j in 1:TH){mtd[j]<-median(DECDV[,j])}
fit<-lm(mtd~t); summary(fit)
```

7.3 Isolating Mortality Credits

The point of a *natural decumulation* fund isn't simply to act as a foil or baseline for the *modern tontine*, but in fact might be a viable product that is offered for sale at the same time. Retirees—or perhaps better described as decumulators—would have a choice of investing in either the *modern tontine*, which I'll abbreviate with MoTo or the *natural decumulation* fund NaDe. The point of offering these two funds together and at the same time would be to highlight the value or benefits from pooling longevity risk. Of course, the initial yield or payout from MoTo would be higher than NaDe, and the exact difference between the two would (obviously) depend on mortality assumptions, which is something I'll return to in just a bit.

7.4 Fitting Gompertz at the Table

The time has come for me to address the elephant in the room (or this book), which is my continuous use of the historic 1825 Benjamin Gompertz law of mortality, when most actuaries in the twenty-first century use discrete mortality tables—and many of them—to value, price and reserve against life annuity liabilities. Indeed, industry actuaries are only vaguely familiar with or aware of Gompertz, last seen during their actuarial exams in school. Rather, the common practice in the insurance industry is to fix a proper *mortality basis* by selecting a (i.) suitable mortality table with possible (ii.) improvement factors, and then combining those with a valuation rate assumption to compute actuarial present values. So, I now discuss how to reconcile those two distinct approaches within the context of *modern tontine* simulations.

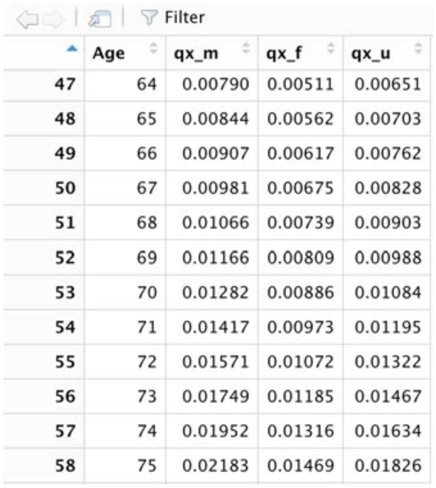
For those who might be new to matters of life and death, a mortality table is a set of numbers ranging from zero to one, which represent the fraction of individuals—alive on their birthday—who are *expected* to die prior to their next birthday. So, for example, one element in a mortality table might be $q_{65} = 0.01$, which should be interpreted to mean that one percent of a sufficiently large group of 65-year-olds is expected to die before their 66th birthday. A full mortality table contains many such mortality rates, ranging from a very young age to an advanced age. These q_x numbers increase and eventually hit one. Eventually everyone dies.

In what follows I will work with a particular mortality table, the CPM2014 table, although everything I do over the next few pages can be applied to anyone of the hundreds (perhaps even thousands) that are used by actuaries on a daily basis. I have selected the CPM2014, which again can be obtained from my website www.MosheMilevsky.com if you can't find it anywhere else, because it is often used to model the mortality of pensioners and future retirees in Canada. It's suitable for the audience who might be interested in *modern tontines.*

Once you have located the data (online) and saved (in your own unique folder), import the mortality table by running the following R-script or using the standard graphic interface in R-studio. Please `view` the file once it has been imported, a part of which I have displayed in Fig. 7.3.

```
library(readr)
CPM2014 <- read_csv("~/CPM2014.csv")
View(CPM2014)
```

After the row index counter in **R**, you will see four columns. The first is an age that ranges from 18 to 115. The second column is the one-year mortality rate for males, the third column is for females and the fourth column is a unisex blended average of the two genders. The unisex mortality rates are the *equally* weighted



	Age	qx_m	qx_f	qx_u
47	64	0.00790	0.00511	0.00651
48	65	0.00844	0.00562	0.00703
49	66	0.00907	0.00617	0.00762
50	67	0.00981	0.00675	0.00828
51	68	0.01066	0.00739	0.00903
52	69	0.01166	0.00809	0.00988
53	70	0.01282	0.00886	0.01084
54	71	0.01417	0.00973	0.01195
55	72	0.01571	0.01072	0.01322
56	73	0.01749	0.01185	0.01467
57	74	0.01952	0.01316	0.01634
58	75	0.02183	0.01469	0.01826

Fig. 7.3 A corner section of the CPM2014 (mortality) table

average of the male and female q_x rates, which can be confirmed by running the following script and should result in a large collection of zeros:

```
round((0.5)*CPM2014$qx_m+(0.5)*CPM2014$qx_f-CPM2014$qx_u,4)
> 0...
```

In the future we might want to generate our own *biased* unisex averages that differ from the fourth column, perhaps based on realized purchase experience. That average might tilt the genders in one way or the other depending on the distribution of investors in the tontine scheme. For now though, I will work with what's given—and focus on the unisex vector—and simply ask you to keep in mind that it doesn't necessarily have to imply a 50/50 split. Next, I will reduce the length of the name of the variable by defining a new `qx` to equal the longer named `CPM2014` and focus on the ages from 65 to 94, which are the 48th to 77th row. Here is the script:

```
qx<-CPM2014$qx_u[48:77]
length(qx)
> 30
min(qx)
> 0.00703
max(qx)
> 0.19541
```

Notice how the lowest (dying between age 65 and 66) mortality rate is 0.7% and the highest (dying between age 94 and 95) is 19.54%, or nearly one in 5. Remember, that q_x value is an average, assuming a large group of individuals. It's not a guarantee, or promise. This is precisely why I have randomized the number of deaths in the `GDEAD` matrix, using the Binomial distribution.

Next it's time to value temporary life annuities and compute their inverse the *initial payout yield* using the above CPM2014 mortality table. The point of this exercise is to examine how that number compares and contrasts with an *initial payout yield* under the Gompertz model, which was our `TLIA(.)` function in Chap. 3. Without repeating the theory again, this involves computing the conditional survival probability (from age 65) to the end of the year at which $ payments take place and then discounting those $ payments by the valuation rate. The survival probability is the product of the one-minus mortality rate q_x, and the discounting can be achieved by creating a vector of present values. I multiply those two vectors together, sum up the discounted conditional cash-flows and compute the inverse to obtain the all-important *initial payout yield.* Here is the (short) script under a 4% effective annual valuation rate.

```
r<-0.04
# Probability of Survival to each age from the present.
ps<-cumprod(1-qx)
# Discount Rate for each year to the present.
dr<-cumprod(rep(1/(1+r),length(qx)))
# Reciprocal of Actuarial Present Value of annual $1 payments.
1/sum(dr*ps)
> 0.0741556
```

Under a 4% effective annual valuation rate, the *initial payout yield* for a 30-year temporary annuity using discrete annual and unisex mortality rates within the CPM2014 mortality table is 7.4%. This is 340 basis points above the valuation rate of 4%. That assumes annual end-of-year payments, which begin on the annuitant's 66th birthday and end on his/her 95th birthday, assuming he/she is alive on that date. For those who might be puzzled by the new commands in the short script, the `rep(.)` creates a vector that repeats $1/(1+r)$ a total of `length(qx)` times. The `cumprod(.)` creates a new vector of the same length, in which the individual elements are cumulatively multiplied and added together. Think of that as the yield curve or term structure of discount rates. Finally, `sum(.)` adds them all together. This process can be repeated for any effective annual interest rate, and under $r = 2\%$ the *initial payout yield* is 6%, all which you should confirm for yourself. In fact, you can also compute *initial payout yields* for other starting ages (e.g. 60 or 70) by setting the q_x vector to equal that subset of the `CPM2014` table.

Now, let's compare those 7.4% and 6.0% *initial payout yields*—using the CPM2014 unisex mortality table—against the Gompertz law of mortality. For convenience, I will recreate the original script explicitly in terms of the Gompertz survival curve (from Chap. 3) with parameters (m, b).

```
TLIA<-function(x,y,r,m,b){
  APV<-function(t){exp(-r*t)*exp(exp((x-m)/b)*(1-exp(t/b)))}
  sum(APV(1:(y-x)))}
```

It's worth looking at that **R**-script carefully once again and noting the symmetry between the `APV` function, which is a product of the discount factor and the survival probability, and the `sum(dr*ps)` segment in the earlier script using the discrete mortality table. Both are doing the same thing. I will now evaluate the `TLIA` function using the same 4% and 2% interest rates, assuming the usual and familiar ($m = 90, b = 10$) parameters. Note that the `TLIA` function takes as input a *continuously compounded* interest rate r, so I will be using ln[1.04] and ln[1.02], which recall is the `log` command in **R.** Finally, here are the results:

```
1/TLIA(65,95,log(1.04),90,10)
> 0.07610133
1/TLIA(65,95,log(1.02),90,10)
> 0.06177168
```

Under the 4% valuation interest rate, the *initial payout rate* under the Gompertz model—relative to unisex CPM2014 mortality—is approximately 19 basis points higher, and 7.61% versus 7.42%. Gompertz is more generous and pays more because he is more deadly. Under the lower and more conservative expected rate of return of 2%, the *initial payout rate* is about 16 basis points higher under Gompertz. Once again, the mortality credits are more generous. Slightly more people are assumed to die under the Gompertz ($m = 90, b = 10$) assumption. So, is this good news or bad news? Is this gap of 0.2% small enough?

Well, when one considers the many other sources of uncertainty including the assumed mix of genders, payment frictions and management costs, I would say the two *initial payout yields* are close enough for comfort in Gompertz. To be more precise, the Gompertz mortality assumption with parameters ($m = 90, b = 10$) will slightly over-estimate mortality, assuming slightly more deaths—relative to the CPM2014 table—and earlier on, thus leading to the higher and more lucrative mortality credits. But to be very clear, and certainly from a pedagogical point of view, the results are close enough to vindicate Benjamin Gompertz within the context of annuity factor valuation. I do not think the 20 basis point difference is material.

Now, this result or effect directly depends on the Gompertz parameters themselves. For example, if I were to (erroneously) assume that (say) the modal value of life is (only) $m = 80$ years and that the dispersion coefficient was a lower $b = 8$ years, the *initial payout yield* using these Gompertz parameters in the `TLIA(.)` function would be quite different, and much higher than the discrete mortality table values.

```
1/TLIA(65,95,log(1.04),80,8)
> 0.1057475
1/TLIA(65,95,log(1.02),80,8)
> 0.0909322
```

Intuitively, a lower assumed life expectancy and a higher force of mortality, which recall is $(1/b)e^{(x-m)/b}$ results in 250 to 300 more basis points of mortality credits. But those numbers aren't consistent or aligned with the 7.4% and 6% initial payout rates from the discrete CPM2014 mortality table.

So, to phrase the question once more, is the *continuous* Gompertz law of mortality consistent with the CPM2014 *discrete* mortality table for the purpose of computing *initial payout yields*? Well, it depends on the actual parameters selected within the simulation model. For ($m = 80, b = 8$), the answer is a definite no, as you can see. The *initial payout yields* are quite different. But, for the above-noted parameters of $m = 90$ and $b = 10$ the payout factors are close and the Gompertz law of mortality is a good approximation. The bad fit is due to a poor choice of *parameters* versus a poor choice of *model.* That distinction—model versus parameters—is something that's important to remember at all times.

7.5 A Look Under the (Mortality) Table

Now, to help understand why the initial payout rates (a.k.a. the inverse of the annuity factors) are so close (or so far) from each other, one has to dig a little deeper and examine the underlying one-year mortality rates q_x and their cumulative survival probabilities. Recall that the vector `ps` contains the survival probabilities from age 65 to age 95. For completeness, I display them here to 3 digits.

```
round(ps,3)
0.993 0.985 0.977 0.968 0.959 0.948 0.937 0.925 0.911 0.896
0.880 0.862 0.842 0.820 0.796 0.769 0.740 0.708 0.672 0.634
0.593 0.550 0.504 0.456 0.407 0.357 0.308 0.261 0.215 0.173
```

Under the CPM2014 assumption, if we begin with 1000 (unisex) investors who are all 65 years old, we can expect to lose 7 of them in the first year, leaving 993 at the age of 66. Then, we *expect* to lose another 8 before the age of 67, leaving 985, etc. This process continues for 30 years at which point we *expect* to have 173 remaining at the age of 95. That is expected under the CPM2014 assumption. Now let's examine the same numbers under the Gompertz ($m = 90, b = 10$) mortality assumption. For convenience, I will repeat the syntax for generating those values, as well as the results.

```
TPXG<-function(x,t,m,b){exp(exp((x-m)/b)*(1-exp(t/b)))}
round(TPXG(65,1:30,90,10),3)
0.991 0.982 0.972 0.960 0.948 0.935 0.920 0.904 0.887 0.868
0.848 0.827 0.803 0.778 0.751 0.723 0.693 0.661 0.627 0.592
0.555 0.518 0.479 0.439 0.399 0.359 0.320 0.281 0.244 0.209
```

Notice how the *expected* Gompertz survivor numbers are slightly lower in the first decade or so, compared to the CPM2014 numbers. At the end of the first year, there are 991 survivors (from an initial group of 1000) compared to the 993. At the end of the second year, there are 982 compared to 985, etc. By the end of ten years, that is age 75, the Gompertz assumption is more deadly, leaving 868 survivors compared to the CPM2014, which has 896. Stated differently, the CPM2014 assumption is expected to leave 28 more people alive.

Recall the emphasis on the word *expected* since our core simulation doesn't quite kill in a precisely Gompertzian manner but generates random deaths that are expected to match that curve. Also, and more importantly, the Gompertz assumption

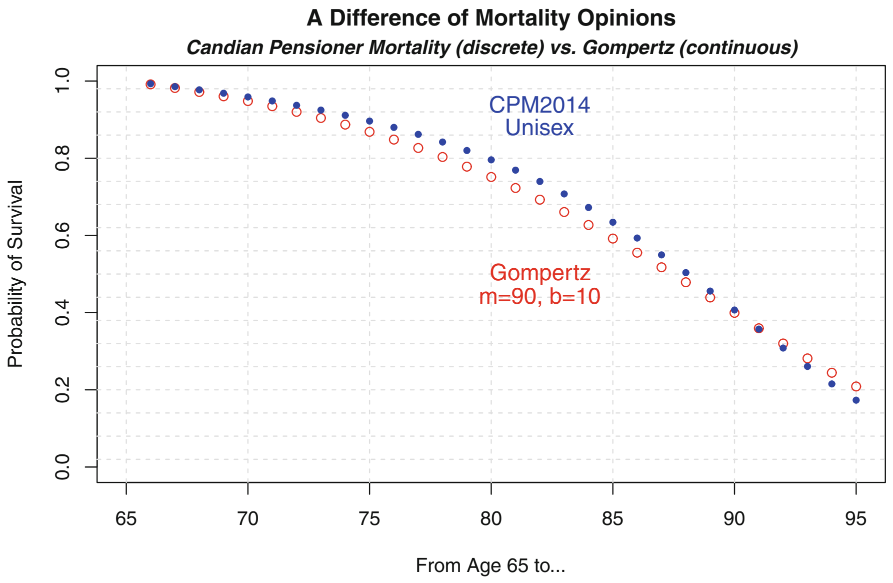


Fig. 7.4 Comparing mortality assumptions: tables versus laws

again with ($m = 90, b = 10$) lightens up a bit on the killing as time goes on, and you will notice that by age 95 there are 209 expected survivors under Gompertz (a 20.9% survival probability), versus the 173 survivors under the CPM2014 table (which is a 17.3% survival probability). All of this can be visualized with the help of Fig. 7.4, using the following script, which I have abbreviated to preserve space.

```
plot(c(65,95),c(0,1),type="n",
     xlab="From Age 65 to...", ylab="Probability of Survival")
grid(ny=18,lty=20)
for (i in 1:30){
  points(65+i,TPXG(65,i,90,10),col="red")
  points(65+i,ps[i],col="blue",pch=20)}
```

Notice how the CPM2014 table sits above the Gompertz curve for the first two decades and then falls under in the final ten years. Now, overall it's the present value of these curves that (really) matter for valuation and payouts, and that obviously depends on the valuation rates, which is another dimension to consider. As you saw earlier, the difference in the actuarial present value was no more than 20 basis points.

In some sense the above not-only vindicates using the (simple, parametric) Gompertz model but is the basis for selecting the specific ($m = 90, b = 10$) assumption within the context of modern tontines offered to (Canadian) retirees. Now, to be very clear, if these funds were to extend their horizons to 35, 40 or to the very end of the human lifecycle, the gap between these mortality assumptions would be greater. One would certainly not be entitled to set actuarial reserves or capital requirements—especially at advanced ages—using the Gompertz assumption, but for our purposes it's sufficient. Remember if-and-when realized mortality and the number of investors dying deviates from our initial assumptions, those losses (or gains) will be accounted for in the *thermostat* design of the fund.

7.6 Projection Factors: Today vs. the Future

The number or date 2014 in the title of discrete mortality tables I have been using over the last few pages wasn't coincidental. In fact, it's meant to remind users that the numbers are *period* mortality rates for a specific group (pensioners) in a specific year (2014). Thus, the number q_{65} denotes the mortality rate for someone who is 65 years old in the year 2014. The number q_{66} is meant to denote the mortality rate for a 66-year-old in 2014, etc. That seems reasonable enough, but our simulations require probabilities of survival and mortality rates q_x for investors who are 65 today and will be 66 next year, 67 the year after, etc. There is a subtle difference between the two because a 66-year-old today is different from a 66-year-old next year, and the underlying q_x values might be different. Intuitively one might expect mortality rates for (say) 66-year-olds to decline ever so slightly over time, as newer cohorts (born later) are likely to be slightly healthier.

What all this means is that current (a.k.a. period) mortality rates have to be projected or reduced into the future using *improvement factors*, which is another set of numbers used by practicing actuaries together with the basic mortality tables discussed above. Now, this is certainly not the venue for an in-depth and detailed discussion of mortality improvement factors, how they are created and when they are used, but I shall say the following. It's rather easy to implement those into the existing simulation code, but they will change the results. In fact, if one assumes very large improvements—for example mortality rates declining by 5% every single year—then survival rates will be much higher, less people will die over the next 30 years and dividends will be reduced. To get a sense of how these *improvement factors* can impact mortality rates q_x and survival rates, I offer the following example.

Before you review and run the script, allow me to explain *how* I'm projecting mortality improvements on the basic CPM2014 table. I'm not using any predetermined projection scales or factors, but I'm artificially assuming the following. Mortality rates for anyone between the age of 65 and 75 will improve (that is decline) by 3% every single year. Mortality rates for individuals between the age of 75 and 85 will improve by 2% per year, and for those between age 85 and 95 it will improve by 1% per year. Stated differently. In 5 years from now the relevant q_{65} will decline to $q_{65}(1-0.03)^5$. That is a decline from (current) 0.00988 to (future) 0.00848. So, the elements in the conditional survival probability `prod(1-qx)` must be adjusted accordingly. That is what this script is doing.

```
imfa<-c(rep(0.03,10),rep(0.02,10),rep(0.01,10))
qx<-CPM2014$qx_u[48:77]*(1-imfa)^(1:30)
ps<-cumprod(1-qx)
round(ps,3)
```

```
0.993 0.986 0.979 0.971 0.963 0.954 0.945 0.935 0.924 0.913
0.900 0.885 0.870 0.853 0.834 0.814 0.792 0.768 0.742 0.714
0.676 0.636 0.594 0.550 0.504 0.457 0.409 0.361 0.314 0.269
```

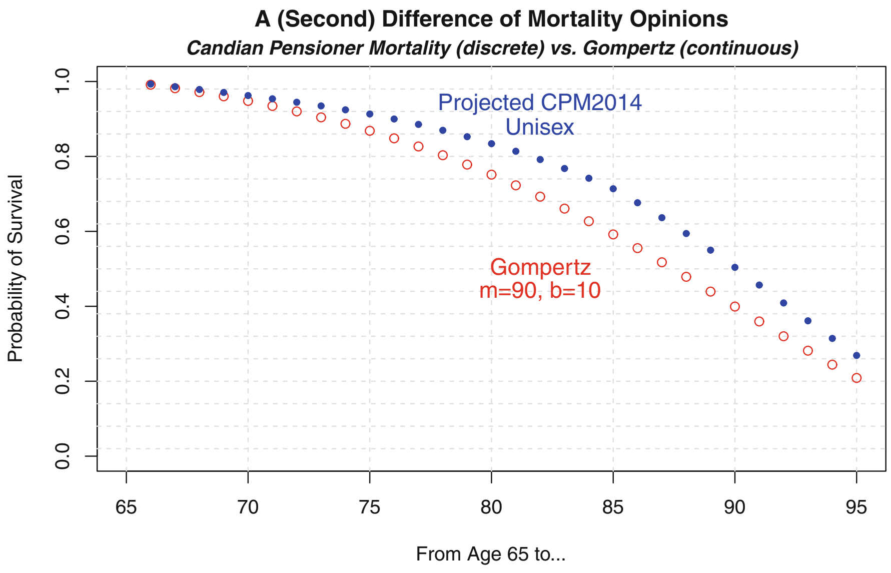
If you compare the number of survivors under the *dynamically projected* CPM2014 to the number of survivors noted earlier under the *static period* table, you will notice more survivors (starting with 1000) after year number one. For example, the above implies 986 survivors, versus 985, by the age of 67. In fact, if you look at the very last and final number, it represents 269 survivors versus the 173 without this *dynamic projection* of improvement factors. Clearly, reducing future mortality rates will have an impact on the fraction of the original 1000 that make it to the end of the 30-year period. More importantly, the Gompertz assumption with ($m = 90$, $b = 10$) will no longer prove accurate. The *initial payout yield*, which is the inverse of the annuity factor, is now:

```
r<-0.04
# Using a Projection Scale
ps<-cumprod(1-qx)
dr<-cumprod(rep(1/(1+r),length(qx)))
round(1/sum(dr*ps),3)
> 0.071
```

This is approximately 30 basis points less than when the *static* CPM2014 was used, which is consistent with the idea that less people are dying because mortality is improving. But when compared against the `1/TLIA(65,95,log(1.04), 90,10)` values, we are now a full half a percent (50 basis points) under the Gompertz values. Something must be done, or to be more specific and practical, the Gompertz parameters will have to be modified.

Figure 7.5 compares the *dynamically projected* values of the underlying q_x vector against the original ($m = 90, b = 10$) parameters, showing the poor fit. But right under the top figure, I have plotted the Gompertz survival probability using modified parameters ($m = 92.17, b = 8.73$) and now the two curves are much closer together. Intuitively if we increase the modal value of the Gompertz distribution and slightly reduce the dispersion, it has the effect of reducing mortality rates and (more) closely matching the *dynamically projected* CPM2014 table.

Now, if you happen to be wondering how (and where in the world) I came up with those two revised values of (m, b) and how I knew they would fit better, the answer is not blind trial and error and certainly isn't divine intervention. Rather, I used a short procedure that has been fully described elsewhere—in the book *Retirement Income Recipes in R*, Chap. 8—on which I will not elaborate or repeat myself. Suffice it to say, the algorithm I used is encapsulated in the following **R**-script that uses a simple regression procedure to locate the best fitting (i.e. optimized) parameters:



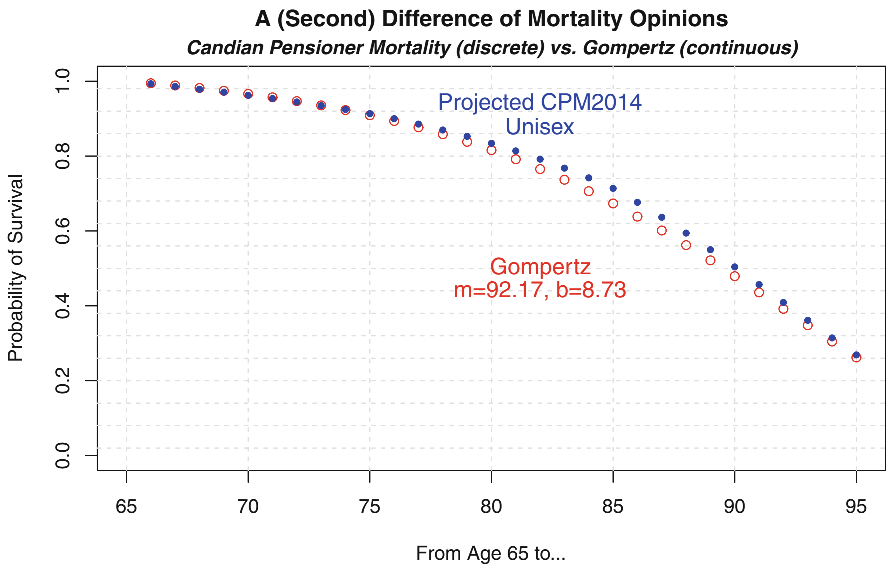


Fig. 7.5 Comparing dynamic mortality assumptions: different (m, b)

```
x<-65:94; y<-log(log(1/(1-qx)))
fit<-lm(y~x)
h<-as.numeric(fit$coefficients[1])
g<-as.numeric(fit$coefficients[2])
```

```
m<- log(g)/g-h/g; m
> 92.16698
b<- 1/g; b
> 8.728464
```

The essence of this script or its secret sauce is to linearize the mortality rate q_x via the double log calculation and then regress that number on age to obtain the Gompertz parameters based on the linearity of the term structure of mortality. If you want to understand *why* this works, the above-noted reference is the place to start. But if all you want is a quick-and-easy procedure for locating the best fitting Gompertz parameters to **any** mortality vector, then those few lines should do the trick. This trick will work whether you want to use a basic table such as the CPM2014 without any *dynamic projections* or whether you want to (first) reduce the mortality rates with unique improvement factors. The key is to fix the q_x vector and then run the script.

In fact, if you look back at Fig. 7.4 and the small gap between the CPM2014 (w/o any projection) and the Gompertz curve with parameters ($m = 90, b = 10$), you might now ask yourself if we can do better and reduce the gap even further using the above-noted optimization procedure. The answer is a definite yes, and I will leave that as an end-of-chapter question. In fact, once you locate those best fitting parameters and use them to compute the *initial payout yield* using the `TLIA(.)` function, you will stumble across another extra 10 basis points, bringing you even closer to the discrete values.

7.7 Working Discretely

If I can sum up the main point of the last few pages and to conclude this chapter, it's as follows. For those users and readers who are reluctant to embrace the Gompertz assumption (and lifestyle) and would rather simulate *modern tontine* payouts using a known discrete mortality table such as the CPM2014, the solution is one line away. Instead of using and defining the survival function `TPXG`, you can define the following:

```
# Alternative to Gompertz
TPXD<-function(qx,t){prod(1-qx[1:t])}
```

Then, replace the `TPXG(.)` function, which appears in various places within the main **R**-script, with the new `TPXD(.)` script, and make sure to include the new mortality vector `qx` as one of your inputs. You can then delete and remove the `m,b` parameters within the script and forget about Gompertz altogether. In fact, the process of generating the `GDEAD[i,j]` matrix can be simplified even further, using the following substitution:

```
# Using the Gompertz Model
GDEAD[i,1]<-rbinom(1,GL0-LAPSE[i,1],1-TPXG(x,1,m,b))
```

```
# Using Discrete Mortality Table
GDEAD[i,1]<-rbinom(1,GL0-LAPSE[i,1],qx[1])
```

The reason for this is that the expression `(1-TPXG(.))` at the end of the `rbinom()` command is just the first year's mortality rate, which is precisely `qx[1]`. There is no need for the `TPXD(.)` function altogether when it comes time to kill people. The same thing would apply to the simulation of survivors in future years, in which `(1-TPXG(x+j-1,1,m,b))` , which is the one-year mortality rate at age `(x+j-1)` is replaced with `qx[j]`, which does the same thing discretely. Now yes, modifying the script to run under discrete mortality will also mean removing some arguments from the `TLIA(.)` and `RTLIA` function, replacing the `(x,m,b)` with the vector of mortality rates `qx`, but that is mostly cosmetics.

In sum, think back to the triangle I presented at the very beginning of Chap. 2. While I prefer to use the Gompertz model in the *modern tontine* simulations, the upper-right corner, possibly calibrated to a mortality table using the regression procedure described earlier, those users who prefer to work directly with a discrete (and very specific) mortality table can do so directly, with just a few changes to the **R**-script. But, the main output, that is the `TONDV[i,j]` matrix, and the underlying fund values are unlikely to change very much. Of course, if you plan to create a *modern tontine* for 95-year-olds, a region of the Gompertz curve in which the fit to discrete mortality tables isn't as smooth, you might want to stick to q_x values. Then again, there are many other things to worry about if the *modern tontine* is sold to nonagenarians.

7.8 Test Yourself

1. Using the non-tontine *natural decumulation* fund, please generate a table with the median as well as the (top) 99th and (bottom) 1st percentile of the fund value in 5, 10, 15 and 20 years, assuming a 4% expected return and valuation rate, and a 3% standard deviation, for a 30-year time horizon. Compare with the results for a *modern tontine* (version 1.0) and confirm that the volatility of the payouts (a.k.a. dividends) as a percent of the expected value is indeed higher for the tontine fund. Finally, force the m parameter to be (astronomically) high for the *modern tontine* script and confirm you get the same exact numerical results as the *natural decumulation* fund.
2. Although the pattern of *dividends* or better described as payouts for the *natural decumulation* fund was presented within the chapter, please generate the relevant figures for the underlying fund value (using the same parameter values) and then discuss and explain their qualitative pattern.
3. Investigate the impact of getting the dispersion coefficient (b) wrong, like we did for the modal value coefficient (m). In particular, assume some reasonable investment returns and that ($m = 90, b = 10$) for pricing purposes, but that realized mortality is such that $b = 7$. In other words, the dispersion of lifetimes

is lower, and the rate at which mortality accelerates $1/b$ is higher than 10%. What is the qualitative impact on the pattern of modern tontine dividends over time?

4. Use the (magic) script to locate the best fitting (m, b) parameters for the basic CPM2014 mortality table at age 65, and then compute the *initial payout yield* using the Gompertz `TLIA(.)` function under a 4% and 2% valuation rate assumption. Compare that to the 7.4% and 6.0% yields derived and explained in the chapter.

8 Managing a Competitive Tontine Business

This chapter reviews some practical business issues that sponsors of a *modern tontine* might encounter when launching this venture, with particular emphasis on the competitive pressures from offering a non-guaranteed solution, the regulatory issues around the provision of securities versus insurance and some other administrative matters that might arise. The chapter begins by discussing how success and failure might be measured within the context of achieving certain return objectives.

8.1 Floors & Failure

Since the *modern tontine* competes head-on with the traditional life annuity issued by a conventional insurance company it's very tempting for the sponsor to go beyond the *sharing* of investment gains & losses and offer some investment guarantees. Of course, if you have reached this stage (and page) you know that the entire *raison detre* of the *modern tontine* is to avoid guaranteeing or promising anything. That dire warning was issued back when death benefits as well as surrender features were introduced. Nevertheless, this next section ponders what might happen if the sponsor were to implement a rule—not necessary a guarantee—that actual tontine dividends would never be less than some pre-determined floor. What would be the implications on the evolution of the fund and the dividends if that (hard) floor were mercilessly adhered to regardless of markets? Could this ruin the fund? And if so, when?

The next bit helps shed light on this precise question. It should be used *after* the user has generated a simulation run in which the tontine dividend has indeed been *forcefully floored* by augmenting the relevant line in the standard script to read: `TONDV[i,j]=max(,kfloor*f0)`. The argument to the left of the comma would represent the un-floored dividend and the parameter `kfloor` itself would be set as a fixed percentage of the initial investment `f0`. For example, echoing the above discussion, the sponsor might include a provision that annual tontine dividends will

M. A. Milevsky, *How to Build a Modern Tontine*, Future of Business and Finance,
https://doi.org/10.1007/978-3-031-00928-0_8

never be declared or distributed at less than a `kfloor` value of 4%, or more precisely \$4 per original \$100 investment. The floor would *never* be applied to the current fund value `DETFV` or the current net asset value, which would make no sense. Rather, the floor would be relative to the original investment `f0`. The **target** payout would remain `kappa[1]`, which would obviously be greater than the floor. Notwithstanding the inability of a fund company to guarantee such a policy outside of an insurance (partnership or) environment, the following R-script computes the failure rate (a.k.a. ruin probability) when implementing this particular feature in a modern tontine.

Now mechanically, when *forcefully flooring* the tontine dividend the "future" might generate scenarios in which the (new) dividend policy creates enough downward pressure on the underlying fund to force a premature collapse. That eventuality must also be dealt with in the R-script using `max(,0)` within the construction of `DETFV[i,j]`. Max mitigates any investment shocks that could lead to negative fund values, which is not allowed. The fund can't borrow against non-existent assets. Thus we are now ready to look for failures. The next R-script searches year by year over the horizon `TH` and keeps track of the number of cases in which the fund value hits zero (or less, just to be safe). Remember: true tontines never die prematurely, they just fade away! Nevertheless, the following run uses `kfloor=0.04`, which is 50 basis points higher than the investment return of $r = \nu = 3.5\%$; all to see what happens when floors are higher than discount rates. That said, the initial value: $\kappa = 6.658\%$ because of the mortality credits. The anticipated tontine dividend yield is 266 basis points above the floor. Here are the parameters I am using for this simulation.

```
x<-65; m<-90; b<-10; GL0<-1000; TH<-30; N<-10000
EXR<-0.035; SDR<-0.07; r<-0.035; f0<-100;
kfloor=0.04
# Parameters that Govern lapsation and redemption.
eta0<- rep(0.02,15)
eta<-c(eta0,rep(0,TH-length(eta0))); dsc<-0.03
```

Here finally is the script and results.

```
ruin4<-c()
for (i in 1:TH){fund<-DETFV[,i]
  ruin4[i]<-length(fund[fund<=0])/length(fund)}
ruin4
```

```
 [1] 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000
 [9] 0.0000 0.0000 0.0000 0.0000 0.0000 0.0006 0.0015 0.0040
[17] 0.0068 0.0102 0.0160 0.0226 0.0293 0.0369 0.0459 0.0557
[25] 0.0642 0.0745 0.0867 0.1012 0.1205 0.5780
```

In terms of the (simple) mechanics, the loop counts the number of elements in the newly defined `fund` vector that are less than or equal to zero via the `length` command and divides that by the total number of simulation runs (yes, usually

10,000) via another call to `length`. The loop starts with the `i=1` fund value at the end of the first year of the modern tontine and ends at the final year `TH`. There are a number of interesting (business) takeaways from this particular run, in which I used the usual ($x = 65, m = 90, b = 10$) demographic parameters, as well as a 2% lapse rate, a 3% surrender charge penalty on lapsing in the first 15 years, plus a full death benefit of the unreturned premium.

Notice how for the first 13 years the failure rate is effectively zero (to four digits). Unsurprisingly and despite the 4% floor on payouts, the impact of occasionally extracting more than one really should isn't felt until a decade and a half after the fund is created. By then of course, it's too late to fix a poor policy and there is now a 6-in-10000 chance the fund is ruined by year #14, or during year #13 to be precise. There were no failures before year 13. But from that point onwards things do and will get worse. The cumulative failure by year #18 is now over one percent and by year #28, the cumulative failure rate has hit 10.1% of cases. Of course that assumes our heavy-handed and overly generous `kfloor` of 4% is maintained regardless.

Figure 8.1 provides another perspective on the financial implications of this floor, plotting the tontine dividends (left panel) and fund value (right panel). Notice that in the first few years or so, there are no (statistical) scenarios in which the tontine dividends fall to the floor. In the language of mathematical optimization the constraint isn't binding in that region. *Why?* Well, the fund value `DETFV[i,j]` multiplied by the relevant `kappa[j]` value and divided by the number of survivors `GLIVE` is greater than above-noted $4,000 per survivor. The problems start to appear (on the left) towards the end of the first decade. We start to see some scenarios in which the floor is binding and the counterfactual (nonfloor) would have been pierced had that *pledge* not been made.

The panel on the right of Fig. 8.1 provides yet another perspective on the failure of the tontine fund when floors are (gratuitously) offered without any further modifications or restrictions. Notice how in this panel the problems become visible between year 15 and 20, when the 1% lower-bound curve hits zero. To be more precise that happens between year #17 and #18, when the above-noted R-script produced `ruin4[18]` above 1%. The large range of values for both the fund value `DETFV` as well as the `TONDV` is typical of high-volatility investments (reminder: $\sigma = 7\%$), which was done deliberately to accentuate and accelerate ruin. What is

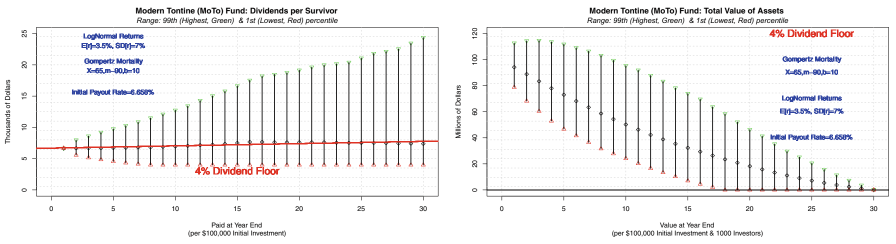


Fig. 8.1 Examining the failure of the 4% floor: $\nu = 3.5\%, \sigma = 7\%$

not typical or standard for the *modern tontine* is the lower bound of the right-hand panel hitting zero before the end of life.

Regardless of what these two pictures in Fig. 8.1 might be saying about the importance of year #15 (dividends), or year #18 (fund values), in the unravelling and collapse of the tontine, the blame—if there is any beyond the sponsor's misguided generosity—lies in the investment returns during the very early years of the fund. Using the `set.seed(1693)`, now users should be able to reproduce the following fund values and standard deviations. (Note the occasionally I have generated results or numbers by using a different seed, which is why I note this here.)

```
mean(DETFV[,25])
> 7594.718
sd(DETFV[,25])/mean(DETFV[,25])
> 0.5653603
mtrx<-cbind(PORET[,1],DETFV[,25])
> cor(mtrx)
          [,1]      [,2]
[1,] 1.0000000 0.2945066
[2,] 0.2945066 1.0000000
```

The average fund value at the end of year #25 is \$7.6 million—remember that `DETFV` is in thousands—but the standard deviation (risk, dispersion, variability) of the fund value in its 25th year of life is almost 57% of the expected value. More importantly and relevant to the question of assigning blame for failure, the correlation between the vector of year #25 values and the vector of year #1 investment returns (separated by a quarter of a century!) is almost 30%. Notice the very high (and alarming) impact of the first few years of the fund on the long term evolution of tontine dividends and the failure rate. Alas, this devil has another name: *sequence of returns.*

This rather alarming situation is driven by the unrealistically high constraint imposed by a 4% floor, and it's unlikely any (rational) sponsor would go near such a number (or promise). But what about a 3% or 2% floor? The **cumulative** failure rates should be lower with looser floors, and indeed results are consistent with intuition. Using the `ruinx[j]` to denote the cumulative failure rate in year `[j]` under a floor of `x`, the above-noted script can be easily modified with minimal surgery.

```
ruin4[25]
> 0.0642
ruin3[25]
> 0.0348
ruin2[25]
> 0.0228
```

Note that depending on what other things you have been calculating and generating at the same time, your simulation results might differ slightly from mine. By I should note (again) that in order to calculate the `ruinx` values, you

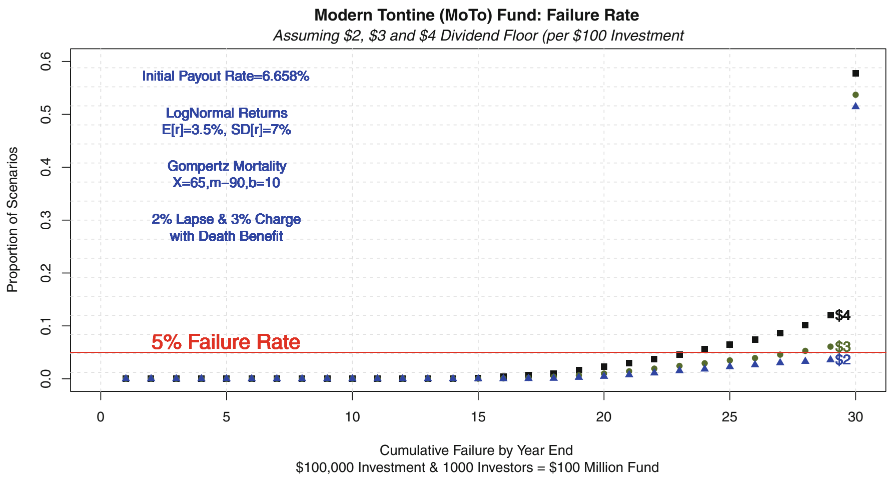


Fig. 8.2 Fund failure frequencies over time and promises

should rerun the entire script and make sure to set the `kfloor` value to percentage representing that rate.

I selected year #25 and the **cumulative** (vs. yearly) failure rate as a baseline for comparison across strategies and floors. The `ruinx[j]` values for the first decade or two of life would be rather boring and close to zero. Hence, if a particular year is selected for benchmarking, it shouldn't be too early—or too late either. Why not the entire horizon? Well, at the very final year of the *modern tontine*'s life, the fund is actually designed to be ruined. So, there should be zeros lurking around that vector. It's the years before `TH` in which zeros shouldn't be visible. To sum up, Figure 8.2 plots the **cumulative** failure rate over the life of the *modern tontine* fund under a variety of floor values, ranging from 2% (not very binding) to 4% (much too tight), with investment assumptions and parameters noted within the figure. The 5% failure rate is noted prominently on the horizon.

To sum up this discussion, in my opinion failure rates that exceed 1%—or for that matter even a quarter of that—should simply be unacceptable to the sponsor, which means that offering such a promise, guarantee or consumer expectation contravenes the entire purpose of the fund: *guarantee nothing*. Nevertheless, if a tontine sponsor does want to create a meaningful *flooring* covenant for its members—similar to the death benefit or lapse value covenant from Chap. 5, there are a few possible paths forward that one could take. The first is to mandate risk-control for the asset allocation and investments within the fund, especially in the early years; mitigating sequencing risk. The costs and benefits of that strategy, within the context of dividend stability, is something worthy of quite a bit of discussion on the business level.

Another possible solution to the *failure rate dilemma* is to place an informal **cap** on the tontine dividend in addition to the above modelled floor; thus building a financial reserve for the lean years. After all, the left-hand panel of Fig. 8.1 shows quite a bit of upside that could perhaps be retained by the fund. Why not hold some back? Indeed, modifying the R-script to incorporate the ceiling involves no more than a few characters and I leave that as a simple assignment. Jumping ahead I can attest that ceilings will work, and do mildly reduce the failure rates, but just barely. In some simulations they don't help at all. But in all scenarios they leave the tontine sponsor with another moral dilemma: *what do we do with leftovers?*

Without getting too caught-up in thick simulation *weeds*, I should note that scenarios in which we observe *failure* or *ruin* are associated with dividend paths that are continuously under the initial κ value. In other words, this is yet again a manifestation of a bad sequence of returns in the early years of the *modern tontine*. My point is that placing a cap on dividends with the hope you might be able to build-up some reserves for the bad times might sound like a good idea, but in practice isn't as helpful as you might think.

Finally, one last possible floor strategy is as follows. What if the sponsor held-back and *skimmed the top* from the Gompertz driven payout rates? Instead of payouts based on the pure `kappa[j]` value vector, they would subtract off a few basis points and keep that as a reserve. Could that create a cushion for the lean years and reduce the failure rate? Preliminary results indicate that it helps, and more so than ceilings which only offer an illusionary benefit. Numerically, shaving 50 basis points off the κ curve would reduce the year-25 failure rate (under a 4% floor) from the above-noted 6.42% to around one-in-twenty. Reducing the payout by a full 100 basis points would reduce the relevant failure rate to around one-in-twenty five, but it can't be driven under the 1-in-100 threshold with any competitive and acceptable haircuts. In conclusion, these haircuts may not solve the *failure* problem entirely, but could move the needle in the right direction.

In fact, there are many *ad hoc* and arbitrary strategies that one could implement on an ongoing basis—and one might be especially tempted to hold back some dividends in (very) good years, but the risk there is the loss of transparency and predictability. This chapter is about managing a business, as opposed to theory or more simulation tricks, so I'll repeat something I have said a number of times before in this book. Whatever the sponsor plans to do—in very good times or very bad times—should be disclosed and explained in advance, so that everyone enters the pool with full information. The tontine algorithm is really not the kind of bridge you wait to cross until you get there.

8.2 Mixing Cohorts, Genders and Initial Sums

As we approach the tail-end of this book, one of the issues I haven't properly addressed is the possible mixing of age and gender cohorts into one larger group of *modern tontine* participants. For most of the prior chapters, the given **R**-script has assumed an initial time-zero pool size of `GL0`, with each member contributing and investing exactly `f0` dollars, and each participant assumed to be precisely of age x. These assumptions have been extremely convenient for the modelling and simulation projections, but in practice are highly unrealistic. First, the sign-up and onboarding period is likely to take some time. Moreover, even if `GL0` investors do end-up joining the *modern tontine* fund, it could conceivably take place over the course of weeks and months. What happens to those who die before others have joined? Is it fair to give the newcomers those dividends? Second, and a much bigger issue is that investors are likely to (want to) invest different sums of money, not all `f0` and they certainly will **not** share a common birthday. In fact, they might not even share a common birth year or even birth decade. Gender will also affect mortality rates. In our language, even if we assume the same (m, b) Gompertz parameters, the x is most certainly different.

Now, in theory one can get back to work and modify the **R**-script, the most recent one being version 3.3, to account for these modifications. Perhaps a version 3.5 of the script would allow for `GL0` different investments from each of the investors. In that case, instead of a scalar `f0`, we would have to define and keep track of an entirely new matrix with `GL0` rows and `TH` columns. The first column of this new matrix will contain the initial investment made, and the subsequent column would contain the tontine dividends that each one of these `GL0` investors are entitled to each year, assuming they are alive. To be clear, life & death would have to be simulated based on each individual's unique age, call it x_k with k ranging from $k = 1$ to `GL0`, versus one single age x for the entire group.

Moreover, this new matrix would have to be generated per simulation scenario, so each run would create `N` copies of this new matrix. The collection of individualized payout matrices would replace the one `TONDV` matrix, which recall didn't properly identify the specific participant. In other words, the computer memory requirements for this sort of exercise would be enormous, at least if you want to keep track of all scenarios at the end of an experiment run.

Another and more subtle issue is how to allocate and partition tontine dividends at the end of each year (or quarter, or month) when the fund starts-off with many different investors at different ages. It's obviously not fair (nor equitable) to distribute the same amount per initial dollar invested to everyone who happens to be alive at the end of the period. We certainly can't multiply a universal κ value by the `DETFV` matrix—as we did in the core of the simulation code—and then split that pot of available money among a homogenous group of `GLIVE` survivors. After all, some survivors might be quite young in age (or gender) and might have invested very little, and they effectively would be subsidized by old investors who might have invested quite a bit more.

Please stop and think about that fact carefully, which is something that *historical tontines* struggled with in their early years during the seventeenth and eighteenth century. Elon Musk (who is currently 50 years old) invests $50 billion in a tontine. You are 30 years old and invest $30. Does that sound fair? To be clear, this isn't just a computer size or memory issue. Rather, the fix here must involve assigning a new κ payout rate that is unique to each of the `GL0` investors who are still alive, based on their age (gender) and possibly the amount they initially invested. In some cases, we might have to impose economic restrictions and the maximum and minimum investment to ensure we don't have a lopsided gamble.

So, if I have convinced you that this mixing cohorts is economically complicated, requires a complete re-write of the **R**-script and must be done with utmost caution that was exactly my intention. More to the point, what I have just described goes beyond the scope of this (free) book. But the computational labour involved isn't insurmountable, and the theory itself has been formulated by many of the researchers I highlighted in the literature review of Chap. 1. It can be done, but not here and not now.

Of course, while theory suggests it's possible to do this *fairly* or at least *equitably*—ensuring that on average the *actuarial present value* all investors receive relative to their initial investment is identical—there is yet another business problem lurking in the background; securities regulation. The next step involves more than talented coders and developers. One requires clever lawyers as well. You see, if the *modern tontine* fund operates as a traditional collective investment trust, the process of allocating and paying *dividends that depend on investor's age or gender* will contravene securities law. It's that simple. Something might be perfectly legal, normal and commonplace under an insurance umbrella, but it can land you in jail in the securities world. I will return to this matter again, in the final Chap. 10.

What all of this implies is that for now we are left with the simple version of the *modern tontine* in which investors should be pooled with others who are *similar in age* so that they can all receive the same dividend payouts without (too) much redistribution from those who are older (and might invest more) to those who are younger (and might invest less). Of course, it remains to be seen what happens when the sponsor of a *modern tontine* announces to the world that anyone between the age of $x_{\texttt{low}}$ and $x_{\texttt{high}}$ are welcome to join the pool, regardless of gender. Only time will tell.

8.3 Test Yourself

1. Show that adding caps do not have any material impact on the ruin probability, assuming a 4% floor.
2. Show that skimming 100 basis points from the natural tontine dividend helps reduce the ruin probability.
3. Show that skimming 100 basis points from the natural tontine dividend only during the first 10 years helps reduce the ruin probability.
4. Show that skimming 100 basis points from the natural tontine dividend during years when the market return is below 0 helps reduce the ruin probability.

9 Solutions and Advanced Hints

This chapter contains high-level solutions to the end-of-chapter *Test Yourself* questions together with some advanced hints on how to get your R-scripts to run faster, smoother and with greater efficiency. This chapter also discusses various other risk metrics, sensitivities to parameter assumption and other matters that were discussed only briefly in earlier chapters. (Note: The primary architect of the solutions provided in this chapter is Joe Bisk.)

9.1 Chapter 3: Brief Solutions to *Test Yourself* Qs.

1. **Question:** Investigate the impact of changing the Gompertz (m, b) assumptions, in the baseline 4% case. In other words, assume the $m = 95$ (in all scenarios) or that $m = 85$ (in all scenarios) and discuss the qualitative difference in the tontine dividends `TONDV` and the decumulation tontine fund `DETFV` values.
   Answer: We examine the median `TONDV` and `DETFV` at year 25, which is when the initial population that started at $x = 65$ will reach age 90. This allows us to see what happens around the modal life expectancy. Here are numerical results (Table 9.1).

   As one might suspect intuitively, *ceteris paribus* (that is all else being equal) higher assumed modal values of m will result in more survivors at any given age, lower tontine dividends at those ages, higher tontine fund values and an overall slower decumulation of the tontine fund. The slightly less intuitive result has to do with dispersion values of b, which recall from the Gompertz model is also the inverse of the mortality growth rate. Notice that when b is reduced from the assumed value of $b = 10$, to $b = 5$ years, the tontine dividends (at year 25) are lower. The intuition here is that as b is reduced to a (rather unrealistic) 5 years, deaths are more concentrated around the modal value which (indirectly) reduces the benefits of longevity risk pooling and the so-called mortality credits.

M. A. Milevsky, *How to Build a Modern Tontine*, Future of Business and Finance,
https://doi.org/10.1007/978-3-031-00928-0_9

Table 9.1 Median `TONDV[,25]` and `DETFV[,25]` for different m and b values

Median `TONDV[,25]`	$m = 85$	$m = 90$	$m = 95$
$b = 10$	\$8.7	\$7.7	\$7.0
$b = 5$	\$8.5	\$7.2	\$6.4
Median `DETFV[,25]`	$m = 85$	$m = 90$	$m = 95$
$b = 10$	\$5.0M	\$9.7M	\$14.5M
$b = 5$	\$0.5M	\$5.6M	\$14.7M

Table 9.2 Confidence intervals (50%) of dividend distributions for different ν and σ values

$(r = \nu, \sigma)$	`TONDV[,10]`	`TONDV[,20]`	`TONDV[,30]`
(3%, 1%)	(\$6.7 , \$7.1)	(\$6.6 , \$7.2)	(\$6.4 , \$7.4)
(5%, 4%)	(\$7.7 , \$9.2)	(\$7.4 , \$9.6)	(\$6.9 , \$10.1)

2. **Question:** Instead of the 4% assumption for both r and `EXR`, please generate simulation results assuming a more conservative 3% return, with a 1% standard deviation, and a more aggressive 5% return, with a 4% standard deviation. In particular, focus on the `TONDV` matrix and create a summary table of the range of tontine dividend payouts in years 10, 20 and 30, with a 50% confidence interval. In other words, compute the first and third quartile at those dates. Explain and comment on the results, and remember that `EXR=r`.
   Answer: Here are some high-level summary results from running the R-scripts using the revised values of $(r = \nu, \sigma)$. As in other chapters, we use the Greek letter ν to denote the expected continuously compounded return, which might (occasionally) be distinct from the assumed return r (Table 9.2).

   Once again, it should be rather intuitive that an investment (or an underlying asset allocation) that is expected to result in higher returns (ν), albeit with greater variability (σ), will also lead to higher and more volatile tontine dividends. However, in the later years (e.g. year #30) the dispersion of tontine dividends is less affected by market performance. Rather, it is the variability in the (relatively smaller) number of survivors within the group that generates the cash-flow volatility.
3. **Question:** Going back to the $r = 4\%$ case, focus on the `TCPAY` matrix and then plot and investigate the distribution of the *time* or the *age* at which investors get their entire money back. Remember, there are 10,000 scenarios embedded within `TCPAY`, and the objective of this question is to get a sense of the times (and ages) at which investors are "made whole" assuming they are still alive.
   Answer: After running the original `v1.0` R-script within the chapter, we created the following script (and data vector) and then plotted the results using the `hist()` command. Figure 9.1 shows the output.

```
MADEWHOLE<-c()
for (i in 1:N){
  MADEWHOLE[i]<-sum(TCPAY[i,]<f0)+1}
hist(MADEWHOLE)
```

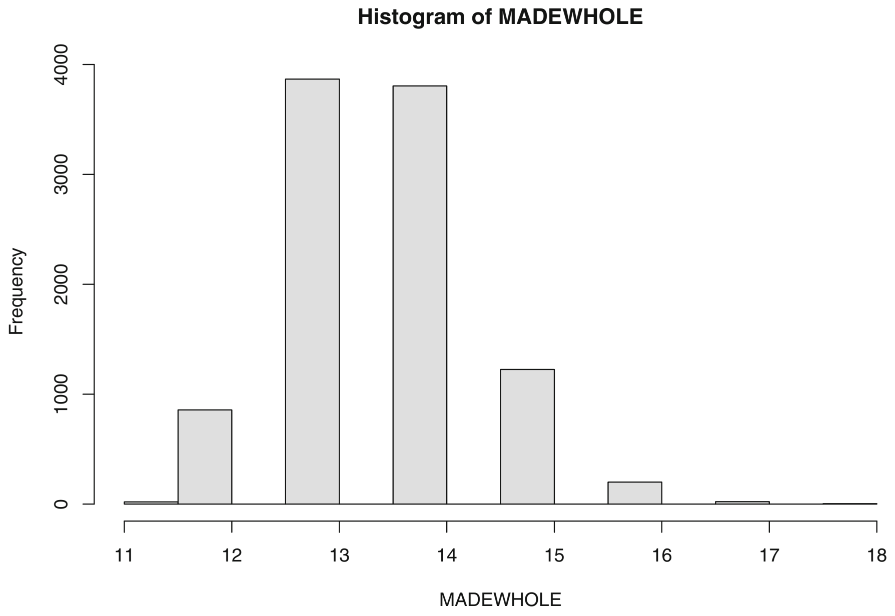


Fig. 9.1 Histogram of the time it takes for investors to be made whole

Carefully notice how the `MADEWHOLE` data vector is created from the `TCPAY` matrix, by adding-up the number of entries in the row that are less than the original `f0`, and also make sure you understand why the number one is added. The key takeaway is that *live* investors are made whole (in our simulation run) somewhere between a minimum of 12 years and maximum of 16 years. More specifically, the command `(sum(MADEWHOLE==13)+sum(MADEWHOLE==14))/10000` results in close to 77% of scenarios being made whole during years 13 and 14, and another 12% during the 15th year. In year 12, we see 9% are made whole. Around one quarter of one percent of scenarios had investors waiting for more than 16 years to get their money back. By then, the vast majority (who were still alive) had received at least $100,000 in dividends. Anyone who died after year 17, which would be age 82, had received their entire money back (albeit slowly over time).

4. **Question:** Focus on the `GLIVE` matrix and the `TCPAY` matrix and please use those two datasets to compute the number of original investors `GL0`, who when they died did not get their full money back. In other words, the sum of the tontine dividends they received until their death didn't exceed their original investment. Remember that people who die in year *i* aren't entitled to any tontine dividends in that year. Once you have figured this out, compute the number of investors who got less than 80% back, less than 60% back and less than 40%, at the time they died.

Answer: After running the `v1.0` script from the chapter, we created the following.

```
SHORT<-matrix(nrow=N,ncol=5)
for (i in 1:N){
  SHORT[i,1]<-GL0-GLIVE[i,sum(TCPAY[i,]<100)+1]
  SHORT[i,2]<-GL0-GLIVE[i,sum(TCPAY[i,]<80)+1]
  SHORT[i,3]<-GL0-GLIVE[i,sum(TCPAY[i,]<60)+1]
  SHORT[i,4]<-GL0-GLIVE[i,sum(TCPAY[i,]<40)+1]
  SHORT[i,5]<-GL0-GLIVE[i,sum(TCPAY[i,]<150)+1]}
summary(SHORT)
```

Once again, the results will be simulation specific. For example, in our simulation run, somewhere between 138 investors (the minimum value in the summary) and 351 investors (the maximum value in the summary), from a total of `GL0=1000` investors, died early and without getting their money back. This is between 14% and 35% of investors who at the time of death have not recovered their original investment. The first quartile was 194 investors and the 3rd quartile was 228 investors, which gives a better sense of the range (between 19% and 23%) of investors who never recover their original investment in the basic (version 1.0) tontine without any refunds or guarantees. Here is the rest of the output of the script.

```
> summary(SHORT)
       V1               V2               V3
 Min.   :138.0   Min.   : 94.0   Min.   : 58.0
 1st Qu.:194.0   1st Qu.:139.0   1st Qu.: 93.0
 Median :211.0   Median :151.0   Median :100.0
 Mean   :212.2   Mean   :151.3   Mean   :101.2
 3rd Qu.:228.0   3rd Qu.:162.0   3rd Qu.:109.0
 Max.   :351.0   Max.   :239.0   Max.   :163.0
       V4               V5
 Min.   : 31.0   Min.   :244.0
 1st Qu.: 58.0   1st Qu.:374.0
 Median : 64.0   Median :411.0
 Mean   : 63.7   Mean   :417.4
 3rd Qu.: 70.0   3rd Qu.:454.0
 Max.   :106.0   Max.   :749.0
```

5. **Question:** Modify the basic simulation so dividends are paid quarterly, and generate results assuming the same Gompertz: $m = 90$, $b = 10$, and $r = 4\%$ case. Be very careful when you modify the code (and increase the size of all the matrices) that your returns and payouts are properly adjusted. For example, the 3-month survival probability and investment return is obviously quite different

from the 12-month values. Once you are done, confirm the TCPAY values after 10, 20 and 30 years are virtually the same.
Answer: The script below gives the values of a modern tontine making k payments a year, where $k = 4$ represents the case of quarterly dividends.

```
k<-4; TH<-30; N<-10000; m<-90; b<-10; x<-65;
GL0<-1000; EXR<-0.04; r<-0.04; SDR<-0.03; f0<-100;
GLIVE<-matrix(nrow=N,ncol=TH*k)
GDEAD<-matrix(nrow=N,ncol=TH*k)
for (i in 1:N){GDEAD[i,1]<-rbinom(1,GL0,1-TPXG(x,1/k,m,b))
GLIVE[i,1]<-GL0-GDEAD[i,1]
for (j in 2:(TH*k)){x1<-1-TPXG(x+(j-1)/k,1/k,m,b)
GDEAD[i,j]<-rbinom(1,GLIVE[i,j-1],x1)
GLIVE[i,j]<-GLIVE[i,j-1]-GDEAD[i,j]}}
PORET<-matrix(nrow=N,ncol=TH*k)
STPRV<-matrix(nrow=N,ncol=TH*k)
for (i in 1:N){
  PORET[i,]<-exp(rnorm(TH*k,EXR/k,SDR/sqrt(k)))-1}
```

In the above script, for the most part we can divide parameters by the value of k, to convert them into the required frequency, other than the standard deviation of investment returns (σ) which must be divided by the square root of k. Recall that variance scales by k, but volatility is the square root of variance. Now, with the life & death matrices, as well as the investment returns in place, we can modify the TLIA function to account for k annual payments and finally calculate tontine fund values and dividends.

```
TLIA<-function(x,y,r,m,b){
  APV<-function(t){p2<-exp(exp((x-m)/b)*(1-exp((t/k)/b)))
  exp(-r*t/k)*p2}; sum(APV(1:((y-x)*k)))}
DETFV<-matrix(nrow=N,ncol=TH*k)
TONDV<-matrix(nrow=N,ncol=TH*k); kappa<-c()
for (i in 1:(TH*k)){kappa[i]<-1/TLIA(x+(i-1)/k,x+TH,r,m,b)}
for (i in 1:N){TONDV[i,1]<-kappa[1]*f0
DETFV[i,1]<-f0*GL0*(1+PORET[i,1])-TONDV[i,1]*GLIVE[i,1]
for (j in 2:(TH*k)){
  TONDV[i,j]<-kappa[j]*DETFV[i,j-1]/GLIVE[i,j-1]
  d3<-TONDV[i,j]*GLIVE[i,j]
  DETFV[i,j]<-max(DETFV[i,j-1]*(1+PORET[i,j])-d3,0)}}
```

With our basic matrices computed and in place, the final step is to create the (also much larger) TCPAY matrix and summarize its values.

```
TCPAY<-matrix(nrow=N,ncol=TH*k)
for (i in 1:N){TCPAY[i,]<-cumsum(TONDV[i,])}
summary(TCPAY[,10*k])
summary(TCPAY[,20*k])
summary(TCPAY[,30*k])
```

If you (the reader or user) have done this part correctly—and this really is the "test yourself" portion—then your `TCPAY` results with quarterly dividends should very closely match the original results in chapter (i.e. annual dividends) at the end of year 10, 20, 30. The small difference (if any) can be blamed on the fact that it's a completely new simulation, but the differences should really be quite small. Here is the output we got.

```
> summary(TCPAY[,10*k])
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  59.95   71.99   74.73   74.89   77.56   92.82
> summary(TCPAY[,20*k])
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  111.1   141.7   149.7   150.2   157.8   203.1
> summary(TCPAY[,30*k])
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  160.7   210.6   224.7   226.0   239.9   327.6
```

9.2 Chapter 4: Brief Solutions to *Test Yourself* Qs.

1. **Question:** Assuming that in fact nobody dies in the first decade of the tontine fund, and that mortality only kicks-in after the age of 75, please compute the number of *extra survivors* that this creates at the end of the `TH=30` year horizon. Does this have a material impact on the number of people who survive from age $x = 65$ to age $x = 95$? Explain this intuitively.
   Answer: We ran the following script to see what happens to `GLIVE[,TH]` if nobody dies in the first ten years.

```
#no deaths first 10 years
for (i in 1:N){
  GDEAD[i,1:10]<-0
  GLIVE[i,1:10]<-GL0
  for (j in 11:TH){
    GDEAD[i,j]<-rbinom(1,GLIVE[i,j-1],1-TPXG(x+j-1,1,m,b))
    GLIVE[i,j]<-GLIVE[i,j-1]-GDEAD[i,j]}}
```

```
> summary(GLIVE[,TH])
   Min.  1st Qu.  Median    Mean  3rd Qu.    Max.
  194.0   231.0   240.0   240.4   249.0   294.0
```

We now compare these numbers to the baseline case in the script below.

```
#regular Gompertz
for (i in 1:N){
  GDEAD[i,1]<-rbinom(1,GL0,1-TPXG(x,1,m,b))
  GLIVE[i,1]<-GL0-GDEAD[i,1]
  for (j in 2:TH){
    GDEAD[i,j]<-rbinom(1,GLIVE[i,j-1],1-TPXG(x+j-1,1,m,b))
    GLIVE[i,j]<-GLIVE[i,j-1]-GDEAD[i,j]}}
```

```
> summary(GLIVE[,TH])
   Min.  1st Qu.  Median    Mean  3rd Qu.    Max.
  160.0   200.0   209.0   208.7   217.0   257.0
```

If there are no deaths in the first decade there is median value of 31 additional survivors to age 95. That is only 3% of the initial pool of 10,000 which is not very significant. The intuition here is that if we start with a large pool of 65 year old's, and miraculously "don't kill" any of them for a decade, the 13% of them, i.e. `1-TPXG(65,10,90,10)` who were supposed to die in the first decade end-up perishing in the subsequent quarter century.

2. **Question:** Along the same lines, please compute the long-run tontine dividend value (i.e. the intercept in the regression) if the *modern tontine* fund was initially set-up assuming the modal value of the Gompertz parameter was $m = 90$, but in fact realized mortality was (much lower, and) consistent with $m = 93$. What is the cost in basis points (i.e. initial yield versus eventual yield)? How many more survivors will the extra 3 modal years lead to at age 95? Explain intuitively.
Answer: To answer this question, we modified only two lines in the entire R-script, changing the realization of `GDEAD` to reflect the lower mortality. Here are those two lines.

```
GDEAD[i,1]<-rbinom(1,GL0,1-TPXG(x,1,93,b))
GDEAD[i,j]<-rbinom(1,GLIVE[i,j-1],1-TPXG(x+j-1,1,93,b))
```

In our simulation run (using `set.seed(123)`), the median tontine dividend drops from $7,671 in the first year to $4,597 in year 30, this is decline of over 300 basis points, from the initial yield to the final yield. The number of survivors in year 30 goes up to 314, from 209 in our baseline case. The intercept in the canonical regression increases to 8.2 (thousand) from 7.66 (thousand). The intuition here is that when realized mortality rates are *lower* than anticipated when the pool was set-up, dividends in the beginning are too high and must be adjusted downwards overtime as participants (to put it crudely) refuse to die as planned. The slope coefficient in the canonical regression shows a decline on average of \$88 per year in tontine dividends. The volatility of tontine dividend (in this misestimated m case) increases from 13% to 17% as well. These numbers don't really do justice to the problem, and we refer readers to a more complete discussion of model risk in Chap. 7.

3. **Question:** Going back to the canonical (standard) simulation results, with ($x = 65, m = 90, b = 10$), carefully examine the TONDV matrix and compute the number of scenarios in which the tontine dividend payout falls below 80% of the **original** payout κ_1, at some point over the 30 year horizon. In other words, what is the probability of a 20% (or more) reduction in the cash-flow provided by the annuity, over the retirement horizon? What is the probability of (only a) a 10% or more reduction?
Answer: We use the standard v1.0 simulation, but then add the following R-script to compute the number of scenarios in which the above-noted *events* take place.

```
c80<-0;c90<-0
for (i in 1:N){
  c80<-c80+(min(TONDV[i,])<0.8*TONDV[i,1])
  c90<-c90+(min(TONDV[i,])<0.9*TONDV[i,1])}
c80/N;c90/N
```

The essence of the script is a counter that adds the number 1 every time the loop *finds* a simulation scenario in which the *worst* tontine dividend min(TONDV[i,]) was less than 80% (and separately 90%) of the initial tontine dividend TONDV[i,1]. After finding all those (bad) scenarios, it then scales the total by N and reports the probability, or better described as frequency. We should note that these sorts of estimates and numbers are related to *extreme value* statistics, a very important branch of statistics—and one that requires additional attention. The results were a 25% probability of having at least one tontine dividend below 80% of the initial payout rate, and a 56% probability of having at least one dividend below 90% of the initial payout rate.

4. **Question:** Similar to the prior question, but subtly different, what is the probability that at any point during the life of the fund the tontine dividend is reduced by 20%? Notice that this "event" is a larger subset of cases, because it also includes the situation in which tontine dividends are increased (in year 5, for example) and then reduced (in year 10, for example) so that from peak to trough the reduction was 20% or more.
Answer: Echoing the *trick* used earlier, after running the standard simulation we used the following script:

```
c80<-0
for (i in 1:N){
  a1<-TONDV[i,1]
  for (j in 1:TH){
    a1<-max(a1,TONDV[i,j])
    if (TONDV[i,j]<0.8*a1){c80<-c80+1; break}}}
c80/N
```

The resulting probability ranged from 41% to 44%. We then generated the canonical simulation using $N = 100,000$ instead of $N = 10,000$, and the results were consistently around 43%.

5. **Question:** Imagine that every single year the sponsor or manager extracts or removes $100,000 from the fund (per $100 million of initial fund value) to cover operating expenses. Clearly, the theoretical κ_1 payout rate is no longer sustainable and the tontine dividends will experience a negative drift over time, as evidenced by the regression slope coefficient. Using a *numerical* process of trial and error, and again assuming the canonical parameter values, please locate the revised value of $\hat{\kappa}_1$ that will support a stable and non-declining tontine dividend over time. How many basis points of initial yield $\kappa_1 - \hat{\kappa}_1$ does this annual $100,000 fixed withdrawal cost the shareholders in the fund? Are there any other issues or problems that are encountered when $100,000 is extracted every year?
Answer: There are many ways to address this problem, each with its own embedded set of economic assumptions. One (very easy) way to model this is by computing the present value of (the fixed) $100,000 per year for 30 years, and then removing that sum from the initial value of the fund. Under an $r = 4\%$ interest rate that present value would be computed via `RGOA(0,0.04,30)*100000`, which is equal to $1,729,203, or approximately 1.73% of the initially contributed $100 million. Another way to think about this present value is that every one of the `GL0=1000` participants must give up or contribute $1,729 from their initial investment to fund this fixed annual cost. Then, the remaining $98,271 goes into a *fee-free* tontine fund that yields the usual $\kappa_1 = 7.67\%$, or 0.0767×98271 for a dividend of $7,538. That, for the record, is 13 basis points less in κ yield. The fund then promises the (revised) tontine dividend, removes the PV of the fixed fees up-front (perhaps in a separate bank account) and the payouts will be stable, albeit from a slightly smaller fund. Now, whether extracting $1.73 million from the decumulation tontine fund at time zero will be viewed as equivalent to removing $100 thousand per year by the *regulators* and *shareholders* is a completely separate matter.

9.3 Chapter 5: Brief Solutions to *Test Yourself* Qs.

1. **Question:** Please generate a tontine dividend dashboard in which the initial seed is changed from (the year) `1693` to `3961`. Discuss how (all) the results change, and why.

Table 9.3 Tontine dashboard: dividend distribution; replicate with `set.seed(3961)`

Modern Tontine Fund: Simulation Dashboard Lifetime Income with Refundable Death Benefit					
Statistical	**DIVIDENDS: End of Year Number...**				
Outcome	$T = 1$	$T = 5$	$T = 10$	$T = 20$	$T = 30$
1 pct. (worst)	\$7,074	\$6,063	\$5,549	\$4,704	\$4,037
25 pct.	\$7,074	\$6,779	\$6,594	\$6,302	\$5,996
Median	\$7,074	\$7,087	\$7,086	\$7,073	\$7,032
75 pct.	\$7,074	\$7,412	\$7,601	\$7,892	\$8,231
99 pct. (best)	\$7,074	\$8,284	\$8,985	\$10,173	\$11,905
St.Dev.	\$0	\$474	\$746	\$1,202	\$1,691
Assumptions:	Financial: $r = 4.0\%$, $\nu = 4.0\%$, $\sigma = 3.0\%$				
$N = 10000$	Gompertz: $x = 65$, $m = 90$ and $b = 10$.				
`TH=30`	Investors: $GL_0 = 1000$ with $f_0 = \$100{,}000$.				

Answer: We generated results using the `v2.0` script after running `set.seed(3961)`, then continued with the following scripts and finally copied the results to the table (Table 9.3).

```
temp<-matrix(nrow=6,ncol=5)
n1<-0
for (j in c(.01,.25,.5,.75,.99)){
  n1<-n1+1
  n2<-0
  for (i in c(1,5,10,20,30)){
    n2<-n2+1
    temp[n1,n2]<-quantile(TONDV[,i],j)*1000
    temp[6,n2]<-sd(TONDV[,i])*1000
  }}
temp
```

Here are the results:

The results will change whenever `R` generates new and different numbers randomly. This affects the number of survivors each year as well as the investment returns, which in turn impact the dividend payout, as one might expect.

2. **Question:** Please create, report and display a tontine dashboard (similar to Table 5.1), but for the aggregate amount of death benefits paid to those who die in the years $T = 1, 5, 10, 15, 20$. That is the `AGDEB[i,j]` matrix. Explain qualitatively what happens in the later (year's) columns and discuss the statistical distribution of those death benefit payouts. Do they seem normally distributed around the mean value? Discuss.
Answer: We ran the `v2.0` script using the 1693 seed, then the following script and finally copied the results to the table (Table 9.4).

Table 9.4 Tontine dashboard: aggregate death benefits; replicate with `set.seed(1693)`

Modern Tontine Fund: Simulation Dashboard					
Lifetime Income with Refundable Death Benefit					
Statistical	**AGDEB: End of Year Number...**				
Outcome	$T = 1$	$T = 5$	$T = 10$	$T = 15$	$T = 20$
1 pct. (worst)	$300,000	$354,776	$318,953	$0	$0
25 pct.	$600,000	$713,313	$555,844	$0	$0
Median	$800,000	$864,682	$664,670	$25,337	$0
75 pct.	$1,000,000	$1,054,044	$784,096	$147,632	$0
99 pct. (best)	$1,600,000	$1,506,436	$1,094,909	$455,954	$0
St.Dev.	$289,4190	$249,057	$168,266	$114,668	$3,755
Assumptions:	Financial: $r = 4.0\%$, $\nu = 4.0\%$, $\sigma = 3.0\%$				
$N = 10000$	Gompertz: $x = 65$, $m = 90$ and $b = 10$.				
`TH=30`	Investors: $GL_0 = 1000$ with $f_0 = \$100{,}000$.				

```
temp<-matrix(nrow=6,ncol=5)
n1<-0
for (j in c(.01,.25,.5,.75,.99)){
  n1<-n1+1
  n2<-0
  for (i in c(1,5,10,15,20)){
    n2<-n2+1
    temp[n1,n2]<-quantile(AGDEB[,i],j)*1000
    temp[6,n2]<-sd(AGDEB[,i])*1000
  }}
temp
```

Here are our results:

In year 15 more than a quarter of the simulations have already paid out $100,000 in dividends, so their aggregate death benefit is zero that year. In year 20, virtually all of the simulated cases have exceeded $100,000 in total dividends paid to the living investors, so no one who dies that year would be entitled to a death benefit. In years 10 and 15 the distribution seems to have a long right tail. The 75th percentile is much farther from the median than the 25th percentile. This is clearly not normally distributed. Readers may also have noticed that (strangely) the standard deviation in year 20 is not zero. This is because there are 5 scenarios out of the 10,000 in which there are still death benefits paid out *even* in the 20th year. You can test this yourself by running `sum(AGDEB[,20]>0)`.

3. **Question:** Assume that a (nefarious or misguided) tontine sponsor *claims* they will offer a death reimbursement covenant, but instead decides to pay the beneficiaries of the deceased a tontine dividend until the entire $f_0 := \$100{,}000$ is returned to the heirs. The death benefit is stretched out over time, instead of

being paid out at once. They say: *Don't worry, we will make your family whole again, but slowly...* Please modify the core simulation v2.0 to account for this feature, discuss whether it makes any difference at all on the tontine dividend payouts, and carefully explain your results.

Answer: We modify the TLIA_RDB function in order to reflect the fact that the death benefit is not RDB but rather an annual payment whose sum equals RDB. This is one possible way of solving this problem. Another would be to break the valuation (and κ function) in two parts, one a term certain annuity, and the other a deferred annuity. Either way the answer will look something like this.

```
TLIA_RDB<-function(x,y,r,m,b,RDB){
  periods<-(y-x)
  value<-0
  for(i in 1:periods){
    PV<-exp(-r*i)
    # The next 3 lines should be combined as one
    value<-value+TPXG(x,i,m,b)
    *PV+(sum((1+r)^-(0:(max(RDB-(i-1),0))))-1)
    *TQXG(x,(i-1),i,m,b)*PV}
  value}
```

We also adjust the AGDEB matrix to reflect the dividends paid to those who are dead instead of the lump sum. Again, we emphasize this isn't the only way to solve this particular problem.

```
for (i in 1:N){
  TONDV[i,1]<-kappa[1]*f0
  TCPAY[i,1]<-TONDV[i,1]
  # Define aggregate death benefits paid (for fund)
  AGDEB[i,1]<-TONDV[i,1]*GDEAD[i,1]
  outflow<-TONDV[i,1]*GLIVE[i,1]+AGDEB[i,1]
  DETFV[i,1]<-f0*GL0*(1+PORET[i,1])-outflow
  for (j in 2:TH){
    if (GLIVE[i,j]>0){
      TONDV[i,j]<-kappa[j]*DETFV[i,j-1]/GLIVE[i,j-1]}
    else {TONDV[i,j]<-0}
    TCPAY[i,j]<-TCPAY[i,j-1]+TONDV[i,j]
    # Define aggregate death benefits paid (for fund)
    if (TCPAY[i,j-1]<f0){
      AGDEB[i,j]<-TONDV[i,j]*(GL0-GLIVE[i,j])}
    else {AGDEB[i,j]<-0}
    outflow<-TONDV[i,j]*GLIVE[i,j]+AGDEB[i,j]
    DETFV[i,j]<-DETFV[i,j-1]*(1+PORET[i,j])-outflow}}
```

The initial tontine dividend payout is approximately $7,244 per year, which is somewhere between the $7,074 number with a normal death benefit, and the $7,671 with no death benefit. This is intuitive, because the present value of the death benefit is lower than the canonical `v2.0`, but still greater than zero, which is the `v1.0` scenario. These results were obtained using the 1693 seed. The resulting dividends appear to have a small (around $37 per year) negative slope over time, which is worth thinking about more carefully.

4. **Question:** Please generate simulation results and create a table similar to the canonical dashboard, in which the reimbursement covenant is weakened in the following manner. If at the end of the year the aggregate value of the decumulation tontine fund is *less than* the original investment minus dividends received, those who died during that year receive the lower of the two values. So, for example, consider the following scenario. Tontine dividends of exactly $7, 000 are received for 3 years, and in the 4th year the investor dies. Under the normal covenant, the beneficiaries of the deceased should receive a death benefit of $100, 000 minus the $21, 000 already received, which is: $79, 000 at the end of year 4. But imagine that the total fund value happens to only be worth $60 million, perhaps due to a (very) bad year in the markets, and there are 800 survivors at the end of year #4. This implies that the notional value of the fund at the end of the 4th year is $75,000 per live survivor. One can think of this as the reserve per person. But, this is also $4, 000 less than the death benefit promised under a *strong* covenant. Well then, under the *weak* one proposed here, it would be reduced to $75,000. Obviously this situation is somewhat artificial, but nevertheless please generate dashboard results under this particular design *and* report the distribution of the number of people—from an original pool of 1000 investors—who die and do **not** get their entire money back. Discuss the qualitative impact of fund values and tontine dividends, when initially `RTLIA` is used to set payouts but then weakened when someone actually dies.
Answer: We changed the `v2.0` script to calculate `AGDEB` based on the so-called *weak* covenant and added a counter to measure how many times the weak covenant actually affects the death benefit. Here is the script.

```
count <- c()
for (i in 1:N){
  count[i]<-0
  TONDV[i,1]<-kappa[1]*f0
  TCPAY[i,1]<-TONDV[i,1]
  AGDEB[i,1]<-GDEAD[i,1]*min(f0,f0*(1+PORET[i,1]))
  # The next two lines should be combined.
  DETFV[i,1]<-f0*GL0*(1+PORET[i,1])-TONDV[i,1]
  *GLIVE[i,1]-AGDEB[i,1]
  for (j in 2:TH){
    TONDV[i,j]<-kappa[j]*DETFV[i,j-1]/GLIVE[i,j-1]
    TCPAY[i,j]<-TCPAY[i,j-1]+TONDV[i,j]
    # The next two lines should be combined.
    AGDEB[i,j]<-max(min(f0-TCPAY[i,j-1],DETFV[i,j-1]
                   *(1+PORET[i,j])/GLIVE[i,j]),0)*GDEAD[i,j]
    # The next two lines should be combined.
    DETFV[i,j]<-DETFV[i,j-1]*(1+PORET[i,j])-TONDV[i,j]
    *GLIVE[i,j]-AGDEB[i,j]
    # The next two lines should be combined
    if (f0-TCPAY[i,j-1]>DETFV[i,j-1]
        *(1+PORET[i,j])/GLIVE[i,j])
    {count[i]<-count[i]+GDEAD[i,j]}
  }}
```

The weak covenant does not affect the dividend payout, which sits around $7,084 as the intercept in the regression model. The slope of the regression is roughly 0, and this was when the 1693 seed was used (Table 9.5).

It appears there is no meaningful change in the tontine dividend payout if the covenant is *weakened*. This is also evident from the distribution of the number of investors who die without getting their money back which is quite rare in our scenarios. We created a frequency table of our `count` vector, which measures how many people died in each simulation without getting their money back. It appears that in 98% of simulations, no investors died without getting their money back. Qualitatively, the *weak* covenant makes no difference to the evolution of the fund value compared to the *strong* covenant, but the latter certainly sounds better than the former.

```
> table(count)
count
   0   1   2   3   4   5   6   7   8   9  10  11
9809   1   1   3   6   9  14  20  14  20  17  14
  12  13  14  15  16  17  18  19  20  21  22  23
  11   5   7   1   2   3   3   5   6   5   4   1
  24  25  26  27  29  30  32  36  41  45  46  50
   3   1   2   3   2   1   1   1   1   1   1   1
```

Table 9.5 Tontine dashboard: dividend distribution under Weakened covenant; replicate with `set.seed(1693)`

Modern Tontine Fund: Simulation Dashboard
Weakened Covenant Refundable Death Benefit

Statistical	**DIVIDENDS: End of Year Number...**				
Outcome	$T=1$	$T=5$	$T=10$	$T=20$	$T=30$
1 pct. (worst)	$7,074	$6,072	$5,538	$4,664	$4,060
25 pct.	$7,074	$6,769	$6,582	$6,305	$5,980
Median	$7,074	$7,074	$7,065	$7,059	$7,000
75 pct.	$7,074	$7,383	$7,589	$7,886	$8,203
99 pct. (best)	$7,074	$8,315	$8,979	$10,165	$11,929
St.Dev.	$0	$469	$743	$1,188	$1,683
Assumptions:	Financial: $r=4.0\%$, $\nu=4.0\%$, $\sigma=3.0\%$				
$N=10000$	Gompertz: $x=65$, $m=90$ and $b=10$.				
`TH=30`	Investors: $GL_0=1000$ with $f_0=\$100{,}000$.				

```
  56
   1
```

5. **Question:** Modify the `v3` script to allow new members to join the tontine in the first 15 years. You can do this by adding `LAPSE` to our `GLIVE` values (instead of subtracting `LAPSE`). When a new person joins during the first 15 years, they pay a lower `f0` into the fund, their initial investment is reduced by 30% of `TCPAY` (30% of the dividends they missed out on by joining late). After 15 years we no longer allow people to join. Also note that we do not allow people to lapse in this specific fund. Find the median tontine dividend for this fund.
Answer: Using the same 1693 seed, we modify the `v3.0` script as follows. First we set `dsc<-0.3` instead of `dsc<-0.25`. Next we modify the part of the script that computes `GLIVE` to add those that join (negative lapsers), and then we assume `eta=0.02` for the first 15 years.

```
for (i in 1:N){
  LAPSE[i,1]<-rbinom(1,GL0,eta[1])
  GDEAD[i,1]<-rbinom(1,GL0,1-TPXG(x,1,m,b))
  GLIVE[i,1]<-GL0-GDEAD[i,1]+LAPSE[i,1]
  for (j in 2:TH){
    LAPSE[i,j]<-rbinom(1,GLIVE[i,j-1],eta[j])
    GDEAD[i,j]<-rbinom(1,GLIVE[i,j-1],1-TPXG(x+j-1,1,m,b))
    GLIVE[i,j]<-GLIVE[i,j-1]-GDEAD[i,j]+LAPSE[i,j]}}
```

Note that we must change `AGLAP` to calculate the aggregate amount contributed to the fund by new joiners, and we must modify `outflow` by subtracting `AGLAP` instead of adding it.

```
for (i in 1:N){
  TONDV[i,1]<-kappa[1]*f0
  TCPAY[i,1]<-TONDV[i,1]
  AGDEB[i,1]<-f0*GDEAD[i,1]
  AGLAP[i,1]<-(f0-TCPAY[i,1]*dsc)*LAPSE[i,1]
  outflow<-TONDV[i,1]*GLIVE[i,1]+AGDEB[i,1]-AGLAP[i,1]
  DETFV[i,1]<-f0*GL0*(1+PORET[i,1])-outflow
  for (j in 2:TH){
    TONDV[i,j]<-kappa[j]*DETFV[i,j-1]/GLIVE[i,j-1]
    TCPAY[i,j]<-TCPAY[i,j-1]+TONDV[i,j]
    AGDEB[i,j]<-max(f0-TCPAY[i,j-1],0)*GDEAD[i,j]
    AGLAP[i,j]<-max(f0-TCPAY[i,j-1]*dsc,0)*LAPSE[i,j]
    outflow<-TONDV[i,j]*GLIVE[i,j]+AGDEB[i,j]-AGLAP[i,j]
    DETFV[i,j]<-DETFV[i,j-1]*(1+PORET[i,j])-outflow}}
```

The median dividend in the 20th year is $7,176, which is higher than the baseline case. The objective of this question is to get you to start thinking about how we would deal with new people joining the tontine after it already started.

9.4 Chapter 6: Brief Solutions to *Test Yourself* Qs.

1. **Question:** Please generate a PORET[i,j] matrix in which 40% of the tontine fund assets are placed in a fund resembling the historical SP500 total return index, and the remaining 60% of the fund is invested in fixed income bonds that are normally distributed with a mean return of 2%, and volatility of 4%, per annum. There is no need to generate the entire *modern tontine* simulation, but please report the summary mean, standard deviation, skewness and kurtosis of the PORET[i,j] matrix in the tenth year of the fund.
Answer: The following script illustrates how to create this mixed PORET matrix and gives the first four moments of PORET in the tenth year of the fund.

```
set.seed(1693)
for (i in 1:N){
  # The next two lines should be combined.
  PORET[i,]<-0.6*rnorm(TH,0.02,0.04)
  +0.4*ANPATH(SP500TR$RETURN,TH)}

mean(log(1+PORET[,10]))
> 0.05631358
sd(log(1+PORET[,10]))
> 0.06735162
skewness(log(1+PORET[,10]))
> -0.003604842
kurtosis(log(1+PORET[,10]))
> 3.020791
```

   This type of allocation provides moments that constitute an average of the volatile high-return SP500 and the more stable low-return bond fund. The skewness and kurtosis of this mixture is close to that of LogNormal.

2. **Question:** Create a PORET[i,j] matrix in which investment returns are based entirely on the historical SP500 total return, but the annual returns are both floored and capped. The floor and cap are located at the upper and lower 15th percentile. What this means is that 15% of the *worst months* are not used or experienced by the fund, in exchange for giving up or sacrificing the 15% of *best months*. To be very clear, your bootstrap procedure should only use 100% or all of 612 months, but replace the extreme returns with their floored and capped values. Again, there is no need to simulate tontine dividends. Rather, report the summary statistics (a.k.a. moments) of this PORET[i,j] matrix in the 10th year.
Answer: The following script illustrates how to bootstrap the capped & floored PORET matrix and returns the first four moments of PORET in the tenth year of the fund.

```
set.seed(1693)
PORET<-matrix(nrow=N,ncol=TH)
```

```
for (i in 1:N){
# Sample SP500
  P1<-ANPATH(SP500TR$RETURN,TH)
# Replace bottom 15\% with floor
  P1[P1<quantile(P1,0.15)]<-quantile(P1,0.15)
# Replace top 15\% with cap
  P1[P1>quantile(P1,0.85)]<-quantile(P1,0.85)
  PORET[i,]<-P1}
mean(log(1+PORET[,10]))
> 0.1042289
sd(log(1+PORET[,10]))
> 0.1127769
skewness(log(1+PORET[,10]))
> -0.07229733
kurtosis(log(1+PORET[,10]))
> 1.991791
```

It looks like the caps & floors significantly reduce the volatility of returns from 15% to 11% without reducing the mean. The skewness here is -0.07 which is not as bad as the -0.20 of the uncapped returns, and the kurtosis is 1.99, which is also quite a bit lower than the 3.2 of the uncapped case. From this simulation it appears that caps and floors are a highly effective way to reduce the *modern tontine's* fund volatility over time.

3. **Question:** The discussion of skewness and kurtosis has been devoted exclusively to the investment returns, but the same computation and summary statistics could be applied to the tontine dividends. Please generate the numbers underlying Tables 6.1 and 6.2 and compare the skewness and kurtosis of the *modern tontine* dividends in the 10th year of the fund. Note that the 10th year is rather arbitrary, but a horizon or time period must be fixed whenever the summary statistics are computed. Averaging all of the dividends or all of the returns would be mixing too many (calendar) apples and oranges.
Answer: After running the `v2.0` tontine script using the two different `PORET` matrices, one bootstrapped from the SP500 index and the other LogNormally generated with `EXR=0.1` and `SDR=0.15`. In both cases we used the 1693 seed. When we used the LogNormal distribution the skewness of the dividends in the tenth year was 1.71 and kurtosis was 8.28. When we bootstrapped from SP500 the skewness of the dividends in the tenth year was 1.64 and kurtosis was 7.64. Overall the skewness and kurtosis of `PORET` do not have a substantial impact on the skewness and kurtosis of the tontine dividends, which are naturally very high.
4. **Question:** The previous question explored the effect that skewness and kurtosis of the `PORET` matrix have on the skewness and kurtosis of `TONDV[,10]`. This question looks at how `PORET` volatility affects the skewness and kurtosis of `TONDV[,10]`. Generate a `PORET` matrix where volatility is 0, and the expected annual return and discount rate are 10%. Use this returns matrix to simulate

Table 9.6 Natural decumulation fund dividend distribution; replicate with `set.seed(1693)`

Natural Decumulation Fund Dividends				
Statistical	$T = 5$	$T = 10$	$T = 15$	$T = 20$
1 pct. (worst)	\$4,982	\$4,619	\$4,413	\$4,199
Median	\$5,799	\$5,817	\$5,834	\$5,865
99 pct. (best)	\$6,705	\$7,290	\$7,660	\$8,170
St.Dev /Mean	6%	10%	12%	14%

Table 9.7 Modern tontine dividend distribution; replicate with `set.seed(1693)`

Modern Tontine Dividends				
Statistical	$T = 5$	$T = 10$	$T = 15$	$T = 20$
1 pct. (worst)	\$6,599	\$6,108	\$5,719	\$5,429
Median	\$7,674	\$7,666	\$7,661	\$7,672
99 pct. (best)	\$8,977	\$9,620	\$10,271	\$10,740
St.Dev /Mean	7%	10%	13%	15%
Assuming: $r = \nu = 4\%, \sigma = 3\%$ $x = 65, m = 90, b = 10$				

modern tontine dividends and measure the skewness and kurtosis of the dividends in the 10th year.
Answer: We ran the `v2.0` script using `EXR<-0.1; SDR<-0; r<-0.1;` and then measured `skewness(TONDV[,10])` and `kurtosis(TONDV [,10])`. The skewness was 0.2 and the kurtosis was 3.2. It appears that the high skewness and kurtosis of the tontine dividends are primarily caused by the volatility of the portfolio returns. We also reran this experiment with various seeds, and much higher values for `GL0, N` and `b`, the skewness and kurtosis were consistently around 0 and 3, respectively.

9.5 Chapter 7: Brief Solutions to *Test Yourself* Qs.

1. **Question:** Using the non-tontine *natural decumulation* fund, please generate a table with the median as well as the (top) 99th and (bottom) 1st percentile of the fund value in 5, 10, 15 and 20 years, assuming a 4% expected return and valuation rate, and a 3% standard deviation, for a 30 year time horizon. Compare with the results for a *modern tontine* (version 1.0) and confirm that the volatility of the payouts (a.k.a. dividends) as a percent of the expected value is indeed higher for the tontine fund. Finally, force the m parameter to be (astronomically) high for the *modern tontine* script and confirm you get the same exact numerical results as the *natural decumulation* fund (Tables 9.6 and 9.7).
   Answer: The volatility of payouts in year 20 is 14% for the natural decumulation fund and 15% for the modern tontine. We forced the m parameter to be nine million (yes, years) and generated the standard tontine `v1.0` script. Here are our results (Table 9.8):
   It appears the volatility of payouts is also 14%, just like the natural decumulation fund. Our dividend values are also statistically the same as those of the

Table 9.8 Modern tontine with m = 9 million that mirrors natural decumulation fund; replicate with `set.seed(1693)`

Modern tontine with m=9 Million				
Statistical	$T = 5$	$T = 10$	$T = 15$	$T = 20$
1 pct. (worst)	\$5,014	\$4,629	\$4,404	\$4,172
Median	\$5,837	\$5,830	\$5,823	\$5,829
99 pct. (best)	\$6,749	\$7,308	\$7,647	\$8,121
St.Dev /Mean	6%	10%	12%	14%

natural decumulation fund, but the numbers are slightly different even though we use the same 1693 seed. The small difference is caused by the fact the modern tontine algorithm generates (some, redundant) random numbers for the `GDEAD` matrix before it generates the random numbers for the `PORET` matrix. Even though all the `GDEAD` numbers are zero, due to the astronomically high m value. The random number generator is still affected by this step, so we get the result of (basically) using a different seed.

2. **Question:** Although the pattern of *dividends* or better described as payouts for the *natural decumulation* fund was presented within the chapter, please generate the relevant figures for the underlying fund value (using the same parameter values) and then discuss and explain their qualitative pattern.
   Answer: Figure 9.2 shows the dividends and Fig. 9.3 shows the fund value of the natural decumulation fund over time. For this fund, dividends are lower and less volatile than for the modern tontine. The fund's value over time appears to be more concave as opposed to a modern tontine which seems more convex. In

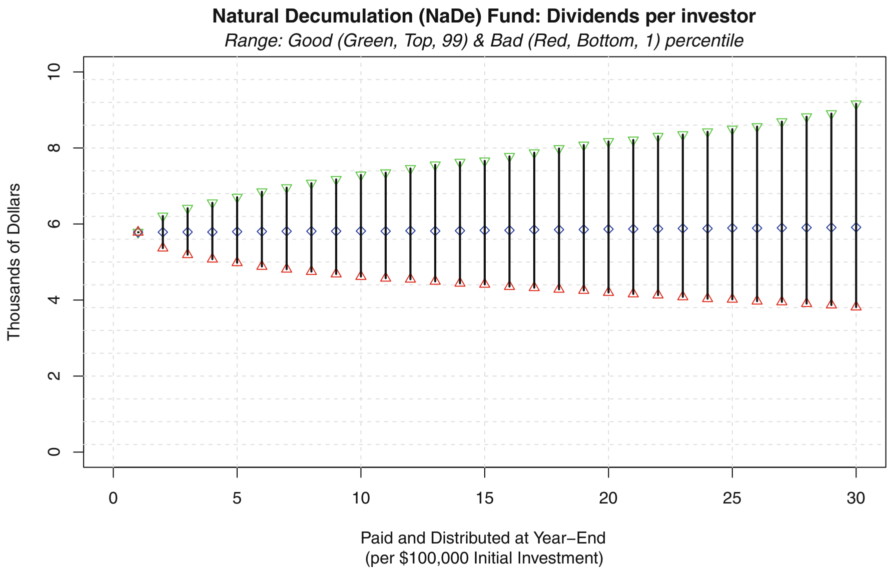


Fig. 9.2 Natural Decumulation (NaDe) fund dividends

the beginning, when death is rare, the natural decumulation fund value declines slower than a modern tontine since it pays individual investors a smaller dividend. Towards the end, when many people have died, the natural decumulation fund value declines quicker than the modern tontine since it needs to pay dividends to `GL0` investors, while the modern tontine only pays dividends to those who survive.

3. **Question:** Investigate the impact of getting the dispersion coefficient (b) wrong, like we did for the modal value coefficient (m). In particular, assume some reasonable investment returns and that ($m = 90, b = 10$) for pricing purposes, but that realized mortality is such that $b = 7$. In other words, the dispersion of lifetimes is lower, and the rate at which mortality accelerates $1/b$ is higher than 10%. What is the qualitative impact on the pattern of modern tontine dividends over time?

   Answer: If the dispersion coefficient is 7 but we pay dividends assuming $b = 10$ then the dividends will trend downwards for the first 18 years and then they will begin trending upwards quickly. Here is the intuition behind this, in the first 18 years, before investors are: $m - b$ or $90 - 7 = 83$ year old, fewer people die each year than expected so the dividends fall. After year 18, we get more deaths each year than what we expected, since the deaths are more closely clustered around m. After time m, the number of survivors drops quicker than expected, so dividends keep rising. If we assume $b = 7$ and the dispersion coefficient is indeed 7, then the dividends will be stable around 7.4 which is lower than our baseline case. Overall the higher and larger the dispersion coefficient b, the greater the

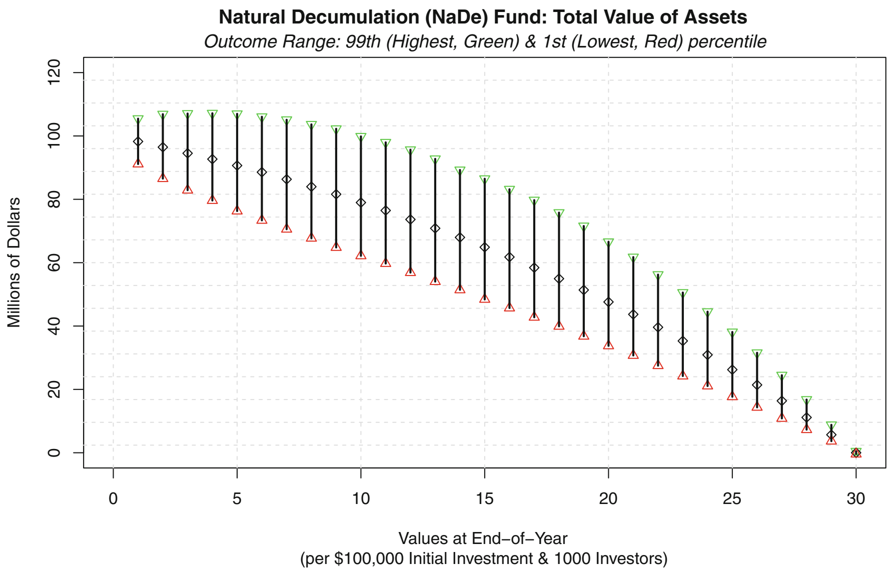


Fig. 9.3 Natural Decumulation (NaDe) fund value

tontine dividends, all else remaining equal. Why? Well, life expectancy will be lower, which then increases the mortality credits. Here are the median annual dividends in the baseline case when we assume correctly that $b = 10$:

```
 [1] 7671 7671 7666 7667 7674 7664 7665 7661 7666 7666 7668
[12] 7661 7664 7658 7661 7661 7673 7662 7666 7672 7662 7662
[23] 7664 7664 7652 7653 7633 7636 7642 7626
```

And, here they are when we (also) assume $b = 10$ but it turns out to be 7. Notice the fall and then rise.

```
 [1] 7671  7635  7594 7551  7510 7478 7428  7376 7331
[10] 7285  7235  7186 7154  7108 7071 7056  7040 7029
[19] 7043  7082  7134 7225  7365 7575 7863  8257 8845
[28] 9669 10922 13275
```

Finally, here is what happens if we assume correctly that $b = 7$.

```
 [1] 7444 7444 7441 7438 7437 7448 7442 7435 7436 7436 7431
[12] 7424 7434 7426 7421 7430 7431 7421 7421 7423 7415 7409
[23] 7400 7404 7408 7400 7422 7426 7418 7395
```

4. **Question:** Use the (magic) script to locate the best fitting (m, b) parameters for the basic CPM2014 mortality table at age 65, and then compute the *initial payout yield* using the Gompertz `TLIA(.)` function under a 4% and 2% valuation rate assumption. Compare that to the 7.4% and 6.0% yields derived and explained in the chapter.
Answer: We ran the following script after loading the (static) mortality table `CPM2014`.

```
qx<-CPM2014$qx_u[48:77]
x<-65:94; y<-log(log(1/(1-qx)))
fit<-lm(y~x)
h<-as.numeric(fit$coefficients[1])
g<-as.numeric(fit$coefficients[2])
m<- log(g)/g-h/g; b<- 1/g;
m;b;
> 89.8461
> 8.228034
x<-65; TH<-30
1/TLIA(x,x+TH,0.04,m,b)
> 0.07529409
1/TLIA(x,x+TH,0.02,m,b)
> 0.06124139
```

Our numerical results were 11 basis points higher than the previously mentioned 7.42% and 6.01% yields which were derived in the chapter. This is extremely close. Remember, it is quite expected for the numbers to slightly differ since we are fitting a curve to (discrete) real world data.

9.6 Chapter 8: Brief Solutions to *Test Yourself* Qs.

1. **Question:** Show that adding caps do not have any material impact on the ruin probability, assuming a 4% floor.
   Answer: We look at the probability of ruin (failure) by the end of year 25 assuming a 4% floor, and see how that changes if we add an 8% cap. With no cap the ruin probability is 6.42%. But, when we add the 8% cap, the ruin probability stays the exact same. If we add a 7% cap, which is quite low, then the ruin probability drops to 6.41% which is negligible. The intuition here is that in simulated scenarios where the floor induces ruin or financial failure, the tontine dividend virtually never rises high enough to hit the cap. We used the following base parameters:

```
# Base parameters are set.
x<-65; m<-90; b<-10; GL0<-1000; TH<-30; N<-10000
EXR<-0.035; SDR<-0.07; r<-0.035; f0<-100;
# Parameters that Govern lapsation and redemption.
eta0<- rep(0.02,15)
eta<-c(eta0,rep(0,TH-length(eta0))); dsc<-0.03
# Paramters that govern floors and caps
kfloor<-0.04; kcap<-0.08
```

   To account for the cap, we modify the calculation of the tontine dividends TONDV, so that it's generated in the following manner:

```
TONDV[i,j]<-min(kappa[j]*DETFV[i,j-1]/GLIVE[i,j-1],kcap*f0)
TONDV[i,j]<-max(TONDV[i,j],kfloor*f0)
```

2. **Question:** Show that skimming 100 basis points from the natural tontine dividend helps reduce the ruin probability or the failure rate.
   Answer: We create and set our parameters as in the previous question, but instead of kcap we create kskim<-0.01, and then we adjust the process for calculating TONDV as follows.

```
# Define first dividend (per individual).
TONDV[i,1]<-kappa[1]*f0-kskim*f0
```

```
# Define annual dividend (per individual).
TONDV[i,j]<-kappa[j]*DETFV[i,j-1]/GLIVE[i,j-1]
```

```
TONDV[i,j]<-TONDV[i,j]-kskim*f0
TONDV[i,j]<-max(TONDV[i,j],kfloor*f0)
```

When we skim 100 basis points of `f0` from each annual dividend, the ruin probability by year 25 drops from 6.42% to 3.65%. This is a significant decline in the risk, but is still unacceptably high. Investors pay the price of $1,000 each year (per $100k investment) which is also very high. Overall this does not seem like a good strategy or a way of generating (more) stability.

3. **Question:** Show that skimming 100 basis points from the natural tontine dividend only during the first 10 years helps reduce the ruin probability.
Answer: We modify the script that calculates `TONDV` as follows:

```
TONDV[i,1]<-kappa[1]*f0-kskim*f0
```

```
TONDV[i,j]<-kappa[j]*DETFV[i,j-1]/GLIVE[i,j-1]
if (j<11){TONDV[i,j]<-TONDV[i,j]-kskim*f0}
TONDV[i,j]<-max(TONDV[i,j],kfloor*f0)
```

This results is `ruin4[25]=0.0445` which is pretty close to the improvement or benefit generated when dividends were skimmed over the full 30 years. But now the total amount skimmed is much smaller. The message is that if we choose to skim, we only need to do that early on, and the skimming in later years is less effective.

4. **Question:** Show that *skimming* 100 basis points from the natural tontine dividend during years when the market return is below 0, helps reduce the ruin probability.
Answer: We modify the script that calculates `TONDV` as follows:

```
TONDV[i,j]<-kappa[j]*DETFV[i,j-1]/GLIVE[i,j-1]
if (PORET[i,j]<0){
TONDV[i,j]<-TONDV[i,j]-kskim*f0}
TONDV[i,j]<-max(TONDV[i,j],kfloor*f0)
```

The result is a 4.73% ruin probability by year 25, which is not as good as when we *skim* every year, or when we skim for the first ten years, but the total cost is lower for most investors. We can expect to see negative market returns roughly 10 out of 30 years (`sum(PORET<0)/300000`), which at first glance looks like the previous scenario (when we skimmed the first 10 years). The main difference is that in this scenario most investors experience fewer skimmed years, since the probability of dividends getting skimmed is spread equally over the whole time horizon, including the later years when many investors are dead. But in the previous question most investors will be alive to have their dividend skimmed for the full 10 years. And yes, the word *skim* doesn't sound very appetizing or ethical, but hopefully the mathematical point is clear.

Concluding Remarks: Tontine Thinking 10

This concluding chapter of the book *How to Build a Modern Tontine* reviews the main rationale and objectives, provides a summary of the *modern tontine* mechanics and discusses some of the constraints imposed by its unique regulatory structure.

10.1 What Is Modern About a Tontine?

Although its seventeenth century incarnation involved a rather standard design, there are in fact many ways to construct a twenty-first century version of a tontine. Arguably, the word *modern* in front of *tontine* could be used to describe any number of products, agreements & contracts in which individuals who live longer than average benefit at the expense of those who don't beat their peers in longevity. Once again please see (all) the scholars mentioned in the literature review of Chap. 1.

A simple life annuity embedded inside a Defined Benefit (DB) pension that guarantees—or at least attempts to pay—a predicable income stream for the rest of one's natural life is a type of (modern) tontine. After all, those who die *early* end up receiving less in total benefits, compared to those who die *late*, even when a spouse or beneficiary is entitled to a residual benefit. Likewise, Variable Annuities (in the US) and Segregated Funds (in Canada) involve a *long* position in an option on a tontine. The owner has the right but not the obligation to (eventually) convert the contract into an income stream that lasts for life. As soon as you hear the words *for life*, you know there is some tontine DNA embedded somewhere. In fact, one can think of simple and humble term life insurance policy—that pays a large death benefit in exchange for a small premium—as having a *short* position in a tontine. The longer you live, the less benefit you receive, relatively speaking. And, if you live too long you receive absolutely nothing in return for all those premiums. Fans of permanent insurance policies might disagree with this characterization, and this certainly isn't meant to detract from the utility benefits of protection, but there is an implicit bet on a short (versus long) life.

M. A. Milevsky, *How to Build a Modern Tontine*, Future of Business and Finance,
https://doi.org/10.1007/978-3-031-00928-0_10

What distinguishes the *modern tontine* I have attempted to describe and model in this book, from its ancient relatives and modern cousins are threefold. First, it's structured as a financial security as opposed to an insurance contract. Second, it's not meant to be used by everyone for their entire life. And third, nothing is meant to be guaranteed. Ever. The first distinction is really a regulatory matter, but one that has very important practical implications. The second is an economic matter, but with important psychological implications. The third permeates the entire structure of the *modern tontine* from start to finish, and most importantly reduces its cost. Allow me to elaborate on all three.

A financial security such as a mutual fund (MF) or exchange trade fund (ETF) is a "product" that invests in a diversified collection of stocks and bonds which is overseen by the fund manager or trustee. The owners of these fund units are entitled to benefits such as bond interest, share dividends and realized capital gains as well as the right to vote from the underlying shares. Every one of those fund owners—especially if they have invested the same amount—must be treated equally under securities regulation. All of this should be quite reasonable and exactly the type of protection investors would want.

This protection or safety assurance implies that managers can't tell one (young, healthy & female) group of shareholders in the same class that they are entitled to less dividends, less interest and less capital gains because their mortality rate is low. Likewise, the same regulators wouldn't allow another group (old, unhealthy, males) to receive the lion share of fund payouts in a given year for the same reasons. And, while a financial or insurance economist might argue: *"but, that is fair"*, the compliance and regulatory authority are unlikely to care. The legal structure of the fund prohibit such overt transfers of wealth. Yes, the fund manager could create different series of funds, or fund classes for people at different ages, genders or even health, but that becomes prohibitively expensive, administratively cumbersome and reduces the efficacy of longevity risk pooling. Remember that a pool consisting of you and a handful of your best pals isn't much of a longevity hedge. Pleading with the *security* regulators for exemption and relief are likely to be met with directions to the nearest *insurance* regulator and hearty farewell wishes.

What all this means is that a *modern tontine* fund operating under securities regulation, issued by a licensed securities entity and sold to individual investors by financial advisors and their intermediaries must abide by a long list of regulatory constraints. These constraints would simply not be present or binding if the *modern tontine* was manufactured, offered and sold under the insurance regulatory umbrella. This is why the *modern tontine* I have described in this book can best be positioned and explained as an enhanced *natural decumulation* fund that slowly depletes and pays-out an initial sum of money over a pre-determined time horizon, albeit with mortality credits from the deceased. Technically, this was structured as a temporary income versus a lifetime of income.

This brings me to the second and third point differentiating the *modern tontine* that is issued within a securities framework from products offered under an insurance regime, and that is the lack of any lifetime income guarantee. It's technically and institutionally impossible for an entity other than a regulated insurance company or pension fund to promise or guarantee something for the *natural life* of any individual investor. The manager of a mutual fund or exchange traded fund makes an assortment of promises and covenants when issuing securities, but one of those cannot be to pay an investor for as long as they live. A company like Tesla can issue and float shares that pay dividends to the shareholder for as long as they own the shares, but Elon Musk can't float securities that pay you dividends for as long as you are alive and cease upon death. Yes, he can enter into a private agreement with you that stipulates almost anything—he is Elon Musk after all—but when operating within the securities regulation even he faces limits, and he has learned this the hard way. For similar reasons, a conventional fund company can **not** issue life annuities, or a pension or a tontine that promises to pay an income for as long as the investor is alive. This is not a quirk or fact limited to securities regulation in North America. It's the nature of the institution itself, and the lawyers really don't care that the *Large of Large Numbers* operates regardless of the type of paper it's written on. Guarantees require capital to back those promises and that's precisely the role of insurance companies.

Now, what a securities-licensed entity can do is create a binding contract in which the parties can agree among themselves that *units* of the fund are cancelled if-and-when their owners happen to die. You own something while you are alive, or provide proof that you are alive within a certain time limit. If such proof isn't supplied, your investment and legal rights in these units are forfeited and ownership of the underlying securities is distributed among those who did supply such proofs—that is they are still alive. This is a type of tontine agreement—but dictates nothing about the time horizon of the agreement. This fund could be set up in advanced to disband in a mere year, or when half the initial pool of investors is dead, in 30 years or even when the *Toronto Maple Leaves* win the Stanley Cup. But, again, what it cannot do is promise to pay me as an individual investor in this fund for the rest of my life. And, one certainly can't promote, advertise or position such a fund as providing a lifetime of income, or being a substitute for a pension annuity. What it does, however, is provide (very) valuable mortality credits, a concept I will not repeat or explain here yet again.

So, what happens if the individual investor lives beyond the terminal horizon of the *modern tontine* fund? From whence will they derive their income if-and-when they reach the age of 95, or 100 or beyond? This *longevity risk* looms large in the minds of most retirees and drives them to hoard financial resources and wealth for a very advanced age, albeit one that in all likelihood may never come. If the *modern tontine* that I have described in the prior chapters of this book can't substitute for a pension annuity. Does it not defeat the purpose? Won't investors remain fearful of *running out of money* and shun such a fund?

The answer relates to something I echoed in the introduction to this book. Note that in today's western society nobody runs-out of money and starves to death. The social safety net and numerous government pension programs exclude that from ever happening.

Indeed, if this was the main fear driving all asset allocations at retirement we would observe a much greater demand for pure, simple and generic life annuities that pay an income for the rest of their natural life. And, for those with strong bequest motives they could purchase a *cashable* term certain annuity for a period of 30, 40 or 50 years which would last until well-after their time on earth, and it would continue to the next generation. Rather, the lack of demand for all these instruments likely implies other reasons for why retirees don't purchase longevity insurance. In sum, the *modern tontine* doesn't provide that sort of insurance. For consumers and retirees who would like to insure or protect themselves against living to a very old age, I urge them to purchase an advanced life delayed annuity (ALDA) which only begins to payout at age 85 or 90. As you can imagine, they are quite cheap at younger ages—given the slim odds you will ever put in a claim—and is exactly why I purchased an ALDA many years ago.

What the *modern tontine* does do, and can be seen quite clearly when placed and positioned right next to a *natural decumulation fund*—is that it offers a group of investors the ability to enhance their investment rate of return by harvesting mortality credits from those investors who predecease them. As I demonstrated, even if investors are offered a *money back* feature that allows their beneficiaries to reclaim unearned principle, the mortality credits will enhance returns by triple digit basis points. And, in today's ultra low interest rate environment those precious credits are even more valuable.

In sum, the *modern tontine* isn't a silver bullet for the silver or white-haired crowd nor is it a substitute for a proper life annuity, defined benefit pension or deferred life annuity. All of those will continue to have an important role to play in the optimal portfolio for the rational decumulator. Likewise, the *modern tontine* is not another asset class, or product class or a revolutionary breakthrough in financial engineering. As you know by now its humble origins are hundreds of years old. Rather, the *modern tontine* is a clever way to squeeze just a bit more *consumption life* from your financial assets, in exchange for sharing those assets with someone else in your cohort. That's all it is, pooling risk for a higher return. Think of it as the twenty-first century version of diversification. You spread your financial assets among many . . . others.

The next step in the evolution of "tontine thinking" is to design a modern accumulation tontine for investors who are younger and who want to harvest mortality credits earlier in the life cycle. Stay tuned for that one. . .