Comparison of Logistic Force of Mortality Models for Predicting Life Table Probabilities of Death: A Simulation-Based Approach

by

James R. Eynon

Submitted in Partial Fulfillment of the Requirements

for the Degree of

MASTER OF SCIENCE

in the

Mathematics

Program

YOUNGSTOWN STATE UNIVERSITY

December, 2011

Comparison of Logistic Force of Mortality Models for Predicting Life Table Probabilities of Death: A Simulation-Based Approach

James R. Eynon

I hereby release this thesis to the public. I understand that this thesis will be made available from the OhioLINK ETD Center and the Maag Library Circulation Desk for public access. I also authorize the University or other individuals to make copies of this thesis as needed for scholarly research.

Signature:

James R. Eynon, Student Date

Approvals:

Dr. Thomas Wakefield, Thesis Advisor Date

Dr. G. Andy Chang, Committee Member Date

Dr. Richard Burden, Committee Member Date

Peter J. Kasvinsky, Dean of School of Graduate Studies and Research Date

ABSTRACT

A program was written in the statistical software package R for conducting Monte Carlo studies based on simulated life tables, and then used in a study to compare two different models for predicting life table probabilities of death. A parametric probability model was used by the program to generate the cohort distribution of deaths, based on supplied life table data. For the present study a cohort life table was constructed using mortality data from the Social Security Death Index, Master File. The models evaluated in the present Monte Carlo study are alternative three-parameter versions of the logistic force of mortality model. The models were fit to simulated life table data for ages 80 to 99, and then used to make probability of death predictions for ages 80 to 105. The Monte Carlo simulations were used to obtain the average values and standard deviations of the probability of death predictions generated by the two models, which were then compared to one another and to the actual probabilities of death based on the probability model that generated the simulated life table data. Results of the simulations showed that the mean probabilities of death predicted by the two models were very similar over the range of ages considered, but usually deviated somewhat from the actual probabilities of death. In the age range of 80-99, the average percentage deviation was less than 2% for each model, while in the age range of 100-105, the average percentage deviation for the models was around 5-6%. In the age range of 100-105, both models always underestimated the true probability of death.

First I would like to thank my thesis advisor, Dr. Wakefield, for his time, his great encouragement and true kindness to me, and patience through the changes in direction of my research and the revisions of my paper, and for his consistently gracious and timely responses in email communication, which were a tremendous blessing as this thesis project neared final completion. What I have learned from his example is worth far more than what I have learned from his instruction. I also want to thank Dr. Chang and Dr. Burden very much for being on my thesis committee, for their time in reviewing and evaluating my work, as well as for their graciousness in it. I also thank Dr. Jamal Tartir, Graduate Coordinator, and the other professors in the Department of Mathematics and Statistics at Youngstown State University from whom I have received instruction and help, or have worked under as a graduate assistant or teaching assistant, and thank the chair of the department, Dr. Nathan Ritchey, for his leadership, as well as our secretaries Sandi Petiya and Nancy O'Hara for their work in the department and help to make it a better place through the pleasant way they interact with us. I thank the School of Graduate Studies and Research for the opportunity to study here as a graduate assistant, and for all those who have helped me through my time here. I also want to thank the professors and teachers who through the many years have had a positive influence on my life, and especially two from my youth that have continued to impact my life, Mr. Stan DeVaney and Mrs. Mary Alley, who I had as teachers at Kansas City Christian Academy. I especially thank my parents, Dr. James and Rhonda Eynon, for their love, support, faithfulness, and help to me to get to this place in my life, and for how good they have been to me through the years, and I thank others in my family who have made an impact on my life including my sister Heidi, my brother Marco, my Grandpa and Grandma Eynon, and my Grandma Reid. I also thank my brothers and sisters in Christ of Salem Bible church who have prayed for me through this project and been there for me including my faithful pastor and his wife, Rev. Douglas and Valerie DeMar, and my friend and mentor Mr. Don Rhinemiller. I am absolutely dependent upon God in everything, and owe a great deal to others for their many prayers through the years. Finally, and above all, I thank the Lord Jesus Christ, who is always with me and never fails in his love for me, and who died on the cross for my sins, bearing the full wrath of God that I deserve, but then rising again victoriously on the third day, having completed God's work of salvation for all who believe. Through faith in him I have received by grace alone the forgiveness of my sins and God's free gift of eternal life, and will never have to face the second death.

Contents

1 Introduction

In this research, a program was written in the statistical software package R for conducting Monte Carlo studies based on simulated life tables, and then used in a study to compare two different models for predicting life table probabilities of death. Simulation of life tables has previously been implemented in other studies. For example, Scherbov and Ediev (2011) used a simulation-based approach in studying life expectancy estimation, with a special emphasis on small populations. As noted by Scherbov and Ediev, preceding their work were other simulation-based studies related to life expectancy, including those by Silcocks et al (2001), Toson et al (2003), Eayres and Williams (2004), and Williams (2005). Howard (2011) used a simulation-based approach in a study concerning the problem of completing life tables for non extinguished cohorts. The present study uses a Monte Carlo simulation-based approach in comparing predictions of life table probabilities of death generated by different force of mortality models. The models evaluated in the Monte Carlo study are alternative three-parameter logistic force of mortality models.

A logistic force of mortality model was first empirically proposed by Wilfred Perks in 1932 (Thatcher, 1998). He proposed a model that included a term for childhood mortality (kc^{-x}), as given by the following equation (Tabeau, 2001)

$$\mu_x = \frac{A + Bc^x}{kc^{-x} + 1 + Dc^x} \ .$$

R.E. Beard, who was a colleague of Perks, later presented a logistic model that he referred to as a Perks model, which did not include a term for childhood mortality (Horiuchi and Coale, 1990). Beard was also the first to derive a logistic force of mortality model under theoretical assumptions, given in a 1959 paper as follows (Beard, 1959; Beard, 1971)

$$\mu_k = \alpha + \frac{(p+1)\beta e^{\lambda k}}{(\gamma\lambda - \beta) + \beta e^{\lambda k}} \ . \tag{1}$$

In Equation (1), p and γ relate to the Gamma distribution that Beard used in his derivation of the logistic model. Beard assumed a heterogeneous population with members belonging to groups governed by a Makeham form of the force of mortality, $\mu_k^s = \alpha + \beta s e^{\lambda k}$, where s is a "longevity" factor which is drawn from a Gamma distribution $\phi(s) = \kappa s^p e^{-\gamma s}$ for $0 \leq s < \infty$ (Beard 1959; Beard, 1971). It is thus easy to understand why the logistic model that Beard derived is also known as the gamma-Makeham model.

The logistic model used in this thesis is of the form given in Equation (2) (Thatcher, 1999)

$$\mu_x = \frac{\kappa\alpha e^{\beta x}}{1 + \alpha e^{\beta x}} + \gamma \ . \tag{2}$$

In Equation (2), the parameter γ is not the same as the γ in Equation (1), but instead is just α with a different name, and the parameter β is just λ with a different name. The parameter α is also not the same as the α in Equation (1).

The model can be broken down into two components which combine to give the total force of mortality at a given age. The first quantity on the right side of Equation (2) is called the senescent force of mortality, which depends upon the age of the individual. The second component, which can be referred to as the background force of mortality, is given by γ in Equation (2). This force of mortality does not vary with age, but makes a greater relative contribution to the total force of mortality at young adult ages, when the quantity $\alpha e^{\beta x}$ is small (See Bongaarts, 2005; Thatcher, 1999). All of the parameters in the model should theoretically be positive values (compare Horiuchi and Coale, 1990, and Thatcher, 1999) .

In a study published in 1999, Thatcher proceeded under the assumption that $\kappa = 1$ and fit historical life table data using a simpler three-parameter logistic force of mortality model as given by

$$\mu_x = \frac{\alpha e^{\beta x}}{1 + \alpha e^{\beta x}} + \gamma \ . \tag{3}$$

The force of mortality model given in Equation (3) will be referred to as the three-parameter logistic model with gamma (G-model) in this paper. Bongaarts fit the G-model to mortality data from 14 different countries in a study published in 2005, and some of his results will be presented later when discussing the results of the Monte Carlo study of the present work.

If no assumption about the parameter κ is made, but instead one assumes that $\gamma = 0$, an alternative three-parameter logistic model is obtained, as given in Equation (4)

$$\mu_x = \frac{\kappa\alpha e^{\beta x}}{1 + \alpha e^{\beta x}} \ . \tag{4}$$

This type of logistic model is attributed to Beard (Beard, 1959; Beard, 1964; Beard, 1971; Horiuchi and Wilmoth, 1998; Tabeau, 2001; Doray, 2008). It may also be referred to as a gamma-Gompertz model, but in this paper it will be referred to as

the three-parameter logistic model with kappa (K-model).

The G-model and the K-model are the focus of the Monte Carlo simulation study conducted as part of this thesis. The life table probabilities of death predicted by the models are compared using a Monte Carlo simulation which estimates their expected values by the average values calculated over many simulated life tables. The standard deviations of the predicted probabilities of death are also obtained. The Monte Carlo simulation program developed in this thesis uses a parametric probability model to generate the cohort distribution of deaths, providing for simulation of the life tables in the Monte Carlo study. Actual life table data supplied to the Monte Carlo simulation program determines the probability of death parameters for the probability model that generates the simulated life table data. A cohort life table was constructed for the present study using mortality data from the Social Security Death Index, Master File (DMF) of the Social Security Administration of the United States of America.

Supporting functions employed by the Monte Carlo simulation program provide for automated nonlinear least squares (NLS) estimation of the parameters in each of the three-parameter logistic force of mortality models when fitting the models to the simulated life table data for a specified range of ages. The NLS parameter estimates are then used to determine probability of death predictions based on each of the models, for a specified range of ages, which may differ from the range of ages used in NLS estimation. The Monte Carlo simulation allows for calculation of the average values and standard deviations of the probability of death predictions generated by the two models, at all specified ages, and these average values can then be compared to the probability of death parameters of the stochastic model used to generate the simulated life tables.

2 Background

2.1 Probability and Statistics Review

The parametric probability model used in the Monte Carlo simulation program to generate simulated life table data is a joint distribution of discrete binomial random variables, which are the number of deaths at each age considered. We therefore first review some probability theory of discrete random variables and the binomial distribution, as follows or is adapted primarily from Casella and Berger (1990).

Define the sample space of a discrete random variable X as the set of all possible

real-number values that X can take.

The probability mass function (pmf), denoted by $f_X(x)$, of the discrete random variable X gives the probability that X will take on a specific value x, for all values of x. This may be written as

$$f_X(x) = P(X = x) \quad \forall x \; .$$

The pmf of any discrete random variable X must be nonnegative for all values of x, and the sum of the values of the pmf over all values of x must be unity. In other words, the pmf must satisfy the following two properties:

1. $f_X(x) \geq 0 \quad \forall x$

2. $\sum_x f_X(x) = 1 \; .$

Define the support set χ of a discrete random variable X as the set of all values of X such that $f_X(x)$ is positive, as given below

$$\chi = \left\{ x : \; f_X(x) > 0 \right\} \; .$$

Then the expected value of a discrete random variable X, denoted by $E(X)$, is given by

$$E(X) = \sum_{x \in \chi} x f_X(x) \; .$$

Furthermore, the expected value of $g(X)$, where g is a function of the discrete random variable X, is given (if the sum exists) by

$$Eg(X) = \sum_{x \in \chi} g(x) f_X(x) \; .$$

The value $Eg(X)$ is said to not exist when $E|g(X)| = \infty$.

The variance of a discrete random variable X, denoted by $\mathrm{Var}(X)$ or σ^2, is given by

$$\mathrm{Var}(X) = E[X - E(X)]^2$$

and the standard deviation of X is then given by

$$\sigma(X) = \sqrt{\mathrm{Var}(X)} \; .$$

In the parametric probability model used in the Monte Carlo simulation program, the distribution of cohort deaths at a given age is assumed to be binomial. The pmf of a random variable having a binomial distribution is given by

$$f_X(x) = \binom{n}{x} p^x (1-p)^{n-x} \quad x = 0, 1, \ldots, n$$

where n is a positive integer, and p is a value in the interval [0,1].

The two parameters of a binomial distribution are thus n, the size parameter, and p, the probability parameter. When a random variable X has a binomial distribution with parameters n and p, we may indicate this by writing $X \sim \text{binom}(n, p)$. For a random variable $X \sim \text{binom}(n, p)$, the expected value and variance of X are given by

$$E(X) = np$$

$$\text{Var}(X) = np(1-p) \; .$$

The parametric probability model used in the Monte Carlo simulation program involves a multivariate random variable that represents the number of deaths at each different age being considered. We therefore now extend our review of probability theory to multivariate random variables and distributions, still following or adapting the treatment by Casella and Berger (1990), but with the use of partitions inspired by the treatment of Dunn and Shultis (2012).

A multivariate discrete random variable $\mathbf{X} = (X_1, \ldots, X_n)$ has a joint pmf $f_{\boldsymbol{X}}(\boldsymbol{x})$ given by

$$f_{\boldsymbol{X}}(\boldsymbol{x}) = P(X_1 = x_1, \ldots, X_n = x_n) \quad \forall \boldsymbol{x} \in \mathbf{R}^n \; .$$

Then $f_{\boldsymbol{X}}(\boldsymbol{x})$ satisfies the following:

$$P(\boldsymbol{X} \in A) = \sum_{\boldsymbol{x} \in A} f(\boldsymbol{x}) \quad \forall A \subset \mathbf{R}^n \; .$$

The expected value of $g(\boldsymbol{X})$, where $g(\boldsymbol{x})$ is a function that is real-valued, and that is defined on the subset of $\mathbf{R}^n$ that is the sample space of $\boldsymbol{X}$, is given by

$$Eg(\boldsymbol{X}) = \sum_{\boldsymbol{x} \in \mathbf{R}^n} g(\boldsymbol{x}) f(\boldsymbol{x})$$

and the variance of $g(\boldsymbol{X})$ is given by

$$\text{Var}[g(\boldsymbol{X})] = E[g(\boldsymbol{X}) - Eg(\boldsymbol{X})]^2 \, .$$

Let $\boldsymbol{C} = (C_1, \ldots, C_k) \subset \boldsymbol{X}$ and $\boldsymbol{D} = (D_1, \ldots, D_{n-k}) \subset \boldsymbol{X}$ such that $\{\boldsymbol{C}, \boldsymbol{D}\}$ forms a partition of $\boldsymbol{X}$. Then the marginal distribution of $\boldsymbol{C}$ is given by the pmf

$$f(\boldsymbol{c}) = \sum_{\boldsymbol{d} \in \mathbf{R}^{n-k}} f(\boldsymbol{x}) \quad \forall \boldsymbol{c} \in \mathbf{R}^k \, .$$

Given $\boldsymbol{C}$ and $\boldsymbol{D}$ as defined above, the conditional pmf of $\boldsymbol{C}$ given $\boldsymbol{d}$ is given by

$$f(\boldsymbol{c}|\boldsymbol{d}) = \frac{f(\boldsymbol{x})}{f(\boldsymbol{d})} \, .$$

For example, suppose $C = X_1$ and $\boldsymbol{D} = (X_2, \ldots, X_n)$. Then the marginal distribution of X_1 is given by

$$f(x_1) = \sum_{(x_2, \ldots, x_n) \in \mathbf{R}^{n-1}} f(x_1, \ldots, x_n)$$

and the conditional pmf of X_1 given $\boldsymbol{d} = (x_2, \ldots, x_n)$ is given by

$$f(x_1|x_2, \ldots, x_n) = \frac{f(x_1, \ldots, x_n)}{f(x_2, \ldots, x_n)}$$

or equivalently

$$P(X_1 = x_1 | X_2 = x_2, \ldots, X_n = x_n) = \frac{P(X_1 = x_1, \ldots, X_n = x_n)}{P(X_2 = x_2, \ldots, X_n = x_n)} \, .$$

Finally, two limit theorems that are significant in the theoretical basis of Monte Carlo simulation are the Law of Large Numbers and the Central Limit Theorem (Dunn and Shultis, 2012). We therefore present each of these below, as follows from Rizzo (2008).

Theorem 1. Strong Law of Large Numbers. *Let $\varepsilon > 0$ and $X_1, \ldots, X_n$ be random variables that are identically distributed such that $E|X_1| < \infty$ and $E(X_1) = \mu$, and such that the random variables are pairwise independent. Define $\overline{X}_n = \frac{1}{n} \sum_{i=1}^n X_i$ for $n = 1, 2, \ldots$. Then*

$$P(\lim_{n \to \infty} |\overline{X}_n - \mu| < \varepsilon) = 1 \, .$$

We say that as n approaches infinity, $\overline{X}_n$ converges to μ almost surely. Thus the sample mean approaches the population mean as the size of an appropriately chosen sample becomes large. In the case of a Monte Carlo simulation, the sample mean is the Monte Carlo estimate of some expected value, and based on the Strong Law of Large Numbers this estimate approaches the expected value as the number of histories in the simulation gets large (Dunn and Shultis, 2012).

Theorem 2. Central Limit Theorem. *Let $X_1, \ldots, X_n$ be random variables that are independent and identically distributed such that $E(X_i)$ equals μ and such that $0 < Var(X_i) < \infty$, and define*

$$Z_n = \frac{\overline{X} - \mu}{\sigma/\sqrt{n}} \, .$$

Then Z_n is distributed asymptotically as a normal distribution with mean zero and variance one ($Z_n \sim N(0,1)$).

This means that for any probability distribution that satisfies the variance condition stated above, the mean of an appropriately chosen sample is asymptotically distributed as $N(\mu, \sigma^2/n)$ (Dunn and Shultis, 2012; Casella and Berger, 1990). Thus as the sample size n gets large, the sample mean $\overline{X}$ approaches the distribution mean μ, and the variance of $\overline{X}$ approaches σ^2/n , which clearly gets smaller and smaller as the sample size n gets larger and larger (Dunn and Shultis, 2012).

2.2 Monte Carlo Method

In her book, *Statistical Computing with R*, Maria Rizzo states,

> "Monte Carlo methods encompass a vast set of computational tools in modern applied statistics...[and] may refer to any method in statistical inference or numerical analysis where simulation is used" (Rizzo, 2008).

Monte Carlo methods rely on repeated sampling, not by obtaining new samples again and again from the actual phenomena being studied, but by obtaining simulated samples. These samples may be generated by resampling from a real sample, or by modeling the way that real samples are generated, and using random numbers to obtain simulated random samples based on the model (Rizzo, 2008; Dunn and Shultis, 2012). The latter approach is the method employed in the Monte Carlo study of this thesis.

Sampling is at the heart of Monte Carlo methods (Gentle, 2003). Dunn and Shultis state,

> "The analysis technique called Monte Carlo is, in essence, a methodology to use sample means to estimate population means" (Dunn and Shultis, 2012).

There is uncertainty in these Monte Carlo estimates of expected values, and the variance of the sample mean can be used to estimate this uncertainty (Dunn and Shultis, 2012).

When predicting life table probabilities of death using the three-parameter logistic models evaluated in the Monte Carlo study of the present work, and samples of life table data generated from the specified probability distribution, the predicted probabilities of death obtained from a given model will vary depending on the sample chosen, since the parameter estimates are based on the given sample, and the samples are subject to random variation. The predicted probability of death at a given age and for a given model is a random variable itself with some probability distribution. If many samples are randomly drawn from the probability distribution that generates the simulated life table data, and probability of death predictions are generated for each sample using the estimated parameter vectors found when fitting the models to each sample, random variates from the distributions of the probability of death predictions at each age may be obtained (adapted from Rizzo, 2008, by extending to a function of an estimator). Estimates of the expected values and standard deviations of the predicted probabilities of death may then be found (Dunn and Shultis, 2012; see also Davidian, 2005). This is the Monte Carlo approach used in the simulation program and study presented in this thesis. The following treatment includes elements adapted/extended or that follow from Dunn and Shultis (2012), Rizzo (2008), and Davidian (2005).

More formally, we may observe that in a given history of the Monte Carlo simulation, the probability of death prediction for age x, $\hat{q}_x$, is a function of the estimator, $\hat{\boldsymbol{\theta}}$, which estimates the parameters of either the K-model or the G-model, and we can show this explicitly as $\hat{q}_x(\hat{\boldsymbol{\theta}})$. In each history of the Monte Carlo simulation we randomly draw a sample $\boldsymbol{d} = (d_1, \ldots d_n)$ that is generated from the simulated distribution of the multivariate random variable $\boldsymbol{D}$, which represents the number of deaths at each age being considered when fitting the models. Now $\hat{\boldsymbol{\theta}}$ is a function of $\boldsymbol{D} = (D_1, \ldots, D_n)$, and so we can write $\hat{\boldsymbol{\theta}}(\boldsymbol{D})$ (Rizzo, 2008). Therefore $\hat{q}_x$ is a function of $\boldsymbol{D}$ through $\hat{\boldsymbol{\theta}}$ and we can show this explicitly by writing $\hat{q}_x(\boldsymbol{D})$. Since

$\boldsymbol{D}$ is a random variable, the function $\hat{q}_x(\boldsymbol{D})$ is also a random variable (Casella and Berger, 1990), and therefore its behavior is governed by some pmf. Our goal is to estimate the expected value and variance of the random variable $\hat{q}_x(\boldsymbol{D})$.

We may not know the pmf of $\hat{q}_x(\boldsymbol{D})$, but we can use the Monte Carlo method to estimate the expected value of $\hat{q}_x(\boldsymbol{D})$, $E(\hat{q}_x(\boldsymbol{D}))$, by the sample mean, $\bar{\hat{q}}_x$, using the following formula

$$\bar{\hat{q}}_x = \frac{1}{m}\sum_{i=1}^{m} \hat{q}_x(\boldsymbol{d}^i)$$

where m is the number of histories of the Monte Carlo simulation and $\hat{q}_x(\boldsymbol{d}^i)$ is the predicted probability of death based on the sample $\boldsymbol{d}^i$ randomly drawn from the simulated distribution of $\boldsymbol{D}$ in the i^{th} history of the Monte Carlo simulation (Dunn and Shultis, 2012; see also Rizzo, 2008, and Davidian, 2005).

Due to uncertainty in the Monte Carlo estimate of $E(\hat{q}_x(\boldsymbol{D}))$, the variance of $\bar{\hat{q}}_x$ should also be calculated (Dunn and Shultis, 2012; Gentle, 2003). Since the estimate $E(\hat{q}_x(\boldsymbol{D}))$ is really a sample mean, being $\bar{\hat{q}}_x$, and since the samples $\hat{q}_x(\boldsymbol{d}^i)$, $i = 1, \ldots, m$ are generated independently and identically (i.e. randomly) from the distribution of $\hat{q}_x(\boldsymbol{D})$, the variance of $\bar{\hat{q}}_x$ is given by

$$\text{Var}(\bar{\hat{q}}_x) = \frac{\sigma^2}{m}$$

where σ^2 is the variance of $\hat{q}_x(\boldsymbol{D})$ and m is the number of samples drawn from that distribution in the simulation (Dunn and Shultis, 2012). The Monte Carlo sample variance s^2 can be used to estimate σ^2, as given by the following formula

$$s^2 = \frac{1}{m-1}\sum_{i=1}^{m}(\hat{q}_x(\boldsymbol{d}^i) - \bar{\hat{q}}_x)^2$$

(Dunn and Shultis, 2012; see also Davidian, 2005). Then the variance of $\bar{\hat{q}}_x$ can be approximated as

$$\text{Var}(\bar{\hat{q}}_x) \simeq \frac{s^2}{m} = \frac{1}{m}\left[\frac{1}{m-1}\sum_{i=1}^{m}(\hat{q}_x(\boldsymbol{d}^i) - \bar{\hat{q}}_x)^2\right]$$

(Dunn and Shultis, 2012; see also Davidian, 2005). The standard deviation of $\bar{\hat{q}}_x$ is then just $\sqrt{\text{Var}(\bar{\hat{q}}_x)}$.

By the Central Limit Theorem, the Monte Carlo sample mean, $\bar{\hat{q}}_x$, is asymptotically distributed $N(\mu, \sigma^2/m)$, where $\mu = E(\hat{q}_x(\mathbf{D}))$ (Dunn and Shultis, 2012). Thus

the Monte Carlo estimate of the variance provides a measure of the uncertainty in how well the Monte Carlo estimate of the expected value estimates the true expected value, and confidence intervals can be constructed about the estimate (Dunn and Shultis, 2012).

There are other statistics that Dunn and Shultis present as also being important in assessing the results of a Monte Carlo simulation. One such statistic is the relative error, R, which is the ratio of the standard deviation of the mean to the mean (Dunn and Shultis, 2012). In the present context R is given by

$$R \equiv \frac{\sqrt{\mathrm{Var}(\bar{\hat{q}}_x)}}{\bar{\hat{q}}_x} \, .$$

Dunn and Shultis (2012) cite a work produced by Los Alamos National Laboratories (MCNP, 2003) when stating that the value of R from a good simulation should usually be no more than 0.05. Other measures for assessing the results of a Monte Carlo simulation include the figure of merit, variance of the variance, and assessment of the underlying probability distribution (see Dunn and Shultis, 2012, for details).

The Monte Carlo method can also be used to estimate the expected values and standard deviations of the life table estimates of the probabilities of death, $\hat{q}_x = \frac{D_x}{N_x}$ (which are actually the maximum likelihood estimates). The Monte Carlo estimates of the predicted probabilities of death determined from each of the three-parameter logistic models may be compared with one another and with the actual probability of death parameters used to generate the simulated life table data. The standard deviations of the probabilities of death predicted by the models may also be compared to one another and to the standard deviations of the life table (maximum likelihood) estimates of the probabilities of death.

2.3 Random Number Generation

The "random" numbers used in Monte Carlo simulations often are not truly random but rather pseudorandom, generated from some algorithm that produces a sequence of numbers that appears to be random. References to random number generation in this paper will simply use the word random, though the fact that the numbers are only pseudorandom should still be kept in mind. The quality and properties of the specific random number generator (RNG) used in a Monte Carlo simulation study should also be considered. The following discussion of random number generation and the generator used in the Monte Carlo simulation program developed as part of this thesis follows from the work of Gentle (2003).

Among the different types of RNGs that have been developed is the *simple linear congruential generator*. This generator can be written in the form

$$x_i \equiv (ax_{i-1} + c) \bmod m \quad 0 \leq x_i < m \; .$$

It can be seen that each pseudorandom number generated is simply the modular reduction of a linear combination of the preceding number generated in the random sequence. We may recall that the modular reduction of a number n by a modulus m is the remainder of the quotient $\frac{n}{m}$. For example, $4 \equiv 9 \bmod 5$. Because the simple linear congruential generator produces a number based on the preceding value in the sequence, in order to begin a sequence, a seed must be supplied to the generator, which is the initial value of x_{i-1}. An additional fact to note is that by dividing each x_i by the modulus m, a sequence of random numbers is generated in the interval (0, 1), and for appropriate values of the multiplier a and modulus m, an apparently random uniform (0, 1) distribution of numbers can be generated (Gentle, 2003).

From the above equation it can be seen that if $c = 0$, then each random number is simply the modular reduction of a fixed multiple of the preceding number, where now $0 < x_i \leq m$, and the RNG is a *multiplicative congruential generator*. It is also possible to have what may be referred to as a *multiple recursive multiplicative congruential generator*, in which each random number generated is a function of the k preceding numbers in the random sequence, with $k > 1$. There are then k multipliers $(a_1, \ldots, a_k)$ and the generator has order k.

One of the properties of a RNG that should be considered is the *period* or *cycle length* of the generator. The sequence of numbers produced by a RNG will eventually begin repeating itself, and the period of a RNG is the amount of numbers that are produced before this occurs. Gentle states,

> "For a random number generator to be useful in most practical simple applications, the period must be of the order of at least 10^9 or so..." (Gentle, 2003).

There are other properties of a RNG that are also important, and various tests have been devised to check the quality of a generator (see Gentle, 2003, for details).

According to Gentle,

> "... although the simple linear congruential generator forms the basis for many good random number generators, by itself it is generally not adequate for serious applications" (Gentle, 2003).

One of the reportedly good generators that is based upon the simple linear congruential generator is the Wichmann-Hill Generator, which is the RNG used in the Monte Carlo simulation program developed in this thesis. The Wichmann-Hill RNG is a combined generator that uses three linear congruential generators as *source generators*. Each of these source generators produces a sequence of numbers as given by the equations

$$\begin{aligned} x_i &\equiv 171 x_{i-1} \bmod 30269 \\ y_i &\equiv 172 x_{i-1} \bmod 30307 \\ z_i &\equiv 170 x_{i-1} \bmod 30323 \ . \end{aligned}$$

Each triplet of numbers (x_i, y_i, z_i) generated by the source generators is then linearly combined and reduced modular 1 to yield a random number in the interval (0, 1), as given by

$$u_i \equiv \left(\frac{x_i}{30269} + \frac{y_i}{30307} + \frac{z_i}{30323} \right) \bmod 1 \ .$$

Gentle states that the Wichmann-Hill generator "...has good randomness properties...[and] is useful...for common applications" (Gentle, 2003). He also states that the generator has a period on the order of 10^{12}. Dunn and Shultis (2012) and Gentle (2003) both cite a study by DeMatteis and Pagnutti (1993) as supporting that the Wichmann-Hill generator has favorable higher-order autocorrelation properties. However, it should also be noted that Gentle does not think it likely that any one RNG is the choice for all uses, and even recommends the idea of using multiple RNGs with multiple seeds when part of an application (Gentle, 2003). Gentle describes a number of tests that can be utilized to test a RNG, and suggests that

> "...before using a random number generator, it is wise to apply ad hoc goodness-of-fit tests that may uncover problems in that particular application, [though] it is desirable that the quality of the output of the generator not be dependent on specific transformations or on a specific seed" (Gentle, 2003).

As alluded to in the preceding quote, random samples generated from various probability distributions other than uniform(0,1) may be desired. Such is the case when life table data is simulated in the Monte Carlo study of this thesis. The parametric probability model for generating cohort deaths requires the generation of random samples

from the binomial distribution. The random numbers generated by a RNG can be used in algorithms that generate random samples from various probability distributions, including the binomial. There are different approaches that can be taken when generating such random samples (See Gentle, 2003, for details), and when considering the amount of random numbers needed in a particular Monte Carlo simulation, it should be understood that these transformations may require many more random numbers than the size of sample being generated.

In the Monte Carlo simulation program developed as part of this thesis, random samples are generated from the binomial distribution using the standard *rbinom* function in R. The function *rbinom* uses what Gentle describes as an efficient method to generate random numbers from the binomial distribution, given by Kachitvichyanukul and Schmeiser (1988) (R Documentation; Gentle, 2003). The Monte Carlo simulation study conducted in the present work simulated life tables that contain survival figures for ages 80 to 106+. This required the generation of 26 random numbers from the appropriate binomial distributions modeling the number of deaths at age 80 to 105.

By determining the expected number of uniform random variates required to generate each of the binomial random variates under the method of Kachitvichyanukul and Schmeiser (1988), over all histories of the Monte Carlo simulation, we are able to determine the total expected number of uniform(0,1) random variates generated by the RNG in the Monte Carlo simulations conducted in the present study. Functions were written in R to carry out these calculations, and this is described in Appendix 7.2. Using the functions supplied with data generated by the simulations, the expected number of random numbers generated by the RNG in the simulations was found to be approximately 8.95E04. This is well below the period of the Wichmann-Hill generator, which is on the order of 10^{12}.

2.4 Nonlinear Least Squares Estimation

The following brief presentation of nonlinear regression theory follows from the treatment of Bates and Watts (2007). A nonlinear regression model includes a dependent random variable whose behavior is modeled by a deterministic component, the expectation function, and a stochastic component, which models the randomness in the response of the dependent variable. The deterministic component is a function of independent variables (or regressors) and parameters, and models the expected response of the dependent variable, based on the state of the independent variables and

the parameter values. The stochastic component models the uncertainty of the responses, as they are not exactly determined by the state of the independent variables but experience random variation, or disturbances, about the expected responses.

A nonlinear regression model may be expressed in the following form

$$\boldsymbol{Y} = \boldsymbol{\eta}(\boldsymbol{\theta}) + \boldsymbol{Z}$$

where $\boldsymbol{Y}$ is a vector of n dependent (or response) variables, $\boldsymbol{Z}$ is a vector of n random variables (or disturbances) that define the stochastic behavior of $\boldsymbol{Y}$, and where $\boldsymbol{\eta}(\boldsymbol{\theta})$ consists of n vectors that define the deterministic behavior of $\boldsymbol{Y}$, given by the functions

$$\eta_i(\boldsymbol{\theta}) = f(\boldsymbol{X}_i, \boldsymbol{\theta}) \quad i = 1, 2, \ldots, n$$

with $\boldsymbol{X}_i$ being a vector of independent variables and $\boldsymbol{\theta}$ being the parameter vector.

The difference between the nonlinear and linear regression models is that in the linear model the response variables are linear functions of the parameters, while in the nonlinear model the response variables are nonlinear functions of the parameter vector with respect to one or more parameter. Thus, the linearity versus nonlinearity of the regression model is with respect to the parameters, not the independent variables.

Nonlinear regression allows for estimation of the parameters of the model based on an observed sample, which provides the data vector $\boldsymbol{y}$. The criterion for determining the best parameter vector based on the observed data is to choose the parameter vector that minimizes the sum of squares of the residuals, $S(\boldsymbol{\theta})$, given by

$$S(\boldsymbol{\theta}) = ||\boldsymbol{y} - \boldsymbol{\eta}(\boldsymbol{\theta})||^2 \, .$$

Geometrically, the nonlinear least squares estimate, $\hat{\boldsymbol{\theta}}$, is chosen such that the point $\boldsymbol{\eta}(\hat{\boldsymbol{\theta}})$ on the curved expectation surface, $\boldsymbol{\eta}(\boldsymbol{\theta})$, is the least distance from the point corresponding to the vector of observed data, $\boldsymbol{y}$ (Bates and Watts, 2007).

According to Bates and Watts, determination of $\hat{\boldsymbol{\theta}}$ is a very challenging problem mathematically, and so iterative methods are used to find it. One such method, the Gauss-Newton method, is the default method used by the *nls* function in R (R documentation). The Gauss-Newton method is based on a sequence of Taylor series approximations of the expectation function, which approximate the curved expectation surface as a linear surface (plane) near estimates of $\boldsymbol{\theta}$. The goal is for estimates of $\boldsymbol{\theta}$ to vary with each iteration such that the estimates progressively get closer and closer to the true least squares estimate $\hat{\boldsymbol{\theta}}$ until convergence is finally reached.

The nonlinear regression model is based on a number of assumptions related to the expectation function, and the disturbances that define the stochastic behavior of the response variables. For more details about nonlinear regression, including its assumptions, and the Gauss-Newton method, see Bates and Watts (2007).

2.5 Life Table Methodology

In his book, *Statistical Models and Methods for Lifetime Data*, Jerry Lawless states,

> "The life table is one of the oldest and most widely used methods of portraying lifetime data...lifetimes and censoring times are grouped into intervals" (Lawless, 2003).

A common way of grouping individuals by age in life tables that express the mortality experience of a general human population is to use one-year intervals. In this way, all individuals dying within a one-year period, for example during their 80th year of life, are grouped together, and the probability of dying within that interval is estimated based on given data.

In general, the number of surviving individuals (or simply, survivors) who are part of a given cohort may decrease due to deaths that occur within an interval, or due to withdrawals (also known as censoring). The latter would be the case if individuals in the cohort migrated from one population being studied to another. However, the cohort obtained for the Monte Carlo study performed as part of the present work consists entirely of individuals whose deaths are recorded in the Social Security Death Index, Master File, and thus the life table constructed for this cohort does not include any withdrawals. Without the issue of withdrawals, analysis of the life table is simplified. The following description of basic life table methodology, follows in part from the treatment of Lawless (2003).

Let N_x be the number of individuals in the cohort surviving to the beginning of their x^{th} year of life (the number surviving to age x), and let D_x be the number of individuals in the cohort who die during their x^{th} year of life (the number who die at age x). Then the probability of dying at age x given that an individual has attained age x, is the conditional probability denoted by q_x, and determined as follows

$$q_x = Pr(\text{death at age } x | \text{survival to age } x) = \frac{D_x}{N_x} .$$

If there are no withdrawals, the number of individuals in the cohort surviving to age $x+1$, N_{x+1}, can be determined as follows

$$N_{x+1} = N_x - D_x \ .$$

The probability of surviving beyond age x given that an individuals has attained age x, the conditional probability denoted by p_x, is simply $1 - q_x$, and can also be determined as follows

$$p_x = \frac{N_{x+1}}{N_x} \ .$$

While q_x gives the conditional probability of dying within the one year interval beginning at age x, given survival to age x, the force of mortality, which may be denoted by μ_t, gives the instantaneous probability of death at time t given survival to time t. Given a model for the force of mortality, the values of q_x and p_x can be calculated, as will be shown at the close of this section.

The conditional pmf of the random variable D_x that represents the number of deaths at age x given survival to age x, is assumed to be a binomial distribution with size parameter N_x and probability parameter q_x (Lawless, 2003), as given by

$$Pr(D_x = d_x | \text{survival to age } x) \sim \text{Binomial}(N_x,\, q_x) \ .$$

The conditional probability of death at age x given survival to age x may be calculated accordingly as follows

$$Pr(D_x = d_x | \text{survival to age } x) = \binom{N_x}{D_x} q_x^{D_x} p_x^{N_x - D_x} \ .$$

The probability model for the deaths of individuals over all ages in the life table can be written as the following joint distribution

$$\begin{aligned} &Pr(D_x = d_x, D_{x+1} = d_{x+1} | \text{survival to age } x+1), \ldots, D_{x+k} = d_{x+k} | \text{survival to age } x+k)) \\ &\quad = Pr(D_x = d_x) Pr(D_{x+1} = d_{x+1} | D_x) \cdots Pr(D_{x+k} = d_{x+k} | D_{x+k-1}, \ldots, D_x) \end{aligned}$$

where x is the first age and $x+k$ the last age considered in the life table (compare with Lawless, 2003). Since all members of the cohort are alive at age x (the first age of the life table), no condition on survival to age x is needed (Lawless, 2003).

We may thus generate a random sample from the joint distribution of cohort deaths by first generating the number of deaths at age x based on a binomial distri-

bution with size parameter N_x equal to the initial cohort size, and the appropriate value for the probability of death parameter q_x, and then generate the number of deaths at each successive age conditional on the number of deaths generated at all preceding ages. The size parameter N_j is determined at each successive age using the formula $N_j = N_{j-1} - D_{j-1}$, as was previously given (see also addendum on page 51).

The probability of surviving beyond age x given survival to age x, p_x, can be determined from the force of mortality, based on the following equation (Thatcher, 1998)

$$p_x = e^{-\int_x^{x+1} \mu_y dy} .$$

For the force of mortality model used in this thesis, we may thus derive an equation for p_x as follows:

$$\begin{aligned}
p_x &= \exp\Big\{-\int_x^{x+1} \mu_y dy\Big\} \\
&= \exp\Big\{-\int_x^{x+1} \Big[\frac{\kappa\alpha e^{\beta y}}{1+\alpha e^{\beta y}} + \gamma\Big] dy\Big\} \\
&= \exp\Big\{-\int_x^{x+1} \frac{\kappa\alpha e^{\beta y}}{1+\alpha e^{\beta y}} dy - \int_x^{x+1} \gamma dy\Big\} \\
&= \exp\Big\{-\int_x^{x+1} \frac{\kappa\alpha e^{\beta y}}{1+\alpha e^{\beta y}} dy - \gamma y\Big|_x^{x+1}\Big\} \\
&= \exp\Big\{-\int_x^{x+1} \frac{\kappa\alpha e^{\beta y}}{1+\alpha e^{\beta y}} dy - \gamma(x+1-x)\Big\} \\
&= \exp\Big\{-\int_x^{x+1} \frac{\kappa\alpha e^{\beta y}}{1+\alpha e^{\beta y}} dy - \gamma\Big\} .
\end{aligned}$$

Letting $u = 1 + \alpha e^{\beta y}$ so that $du = \beta\alpha e^{\beta y} dy$, and temporarily ignoring the limits of integration we have

$$p_x = \exp\Big\{-\int \frac{\kappa}{\beta}\frac{du}{u} - \gamma\Big\} .$$

Integrating and then substituting back for the quantity u yields

$$\begin{aligned} p_x &= \exp\Big\{-\frac{\kappa}{\beta} ln\big|1+\alpha e^{\beta y}\big|\Big|_x^{x+1} - \gamma\Big\} \\ &= \exp\Big\{-\frac{\kappa}{\beta}\Big(ln\big|1+\alpha e^{\beta(x+1)}\big| - ln\big|1+\alpha e^{\beta x}\big|\Big) - \gamma\Big\}\,. \end{aligned}$$

Assuming $\alpha > 0$ and rearranging slightly yields the formula for p_x as derived from the four-parameter logistic force of mortality model used in this thesis

$$p_x = \exp\Big\{\frac{\kappa}{\beta}\Big[ln\big(1+\alpha e^{\beta x}\big) - ln\big(1+\alpha e^{\beta(x+1)}\big)\Big] - \gamma\Big\}\,.$$

For the G-model, which assumes $\kappa = 1$, the formula for p_x is then given by

$$p_x = \exp\Big\{\frac{1}{\beta}\Big[\ln\big(1+\alpha e^{\beta x}\big) - \ln\big(1+\alpha e^{\beta(x+1)}\big)\Big] - \gamma\Big\}$$

and for the K-model, which assumes $\gamma = 0$, the formula for p_x is given by

$$p_x = \exp\Big\{\frac{\kappa}{\beta}\Big[\ln\big(1+\alpha e^{\beta x}\big) - \ln\big(1+\alpha e^{\beta(x+1)}\big)\Big]\Big\}\,.$$

The formulas to calculate q_x are then found easily, being given by $1 - p_x$. For the four-parameter logistic model used in this thesis, the equation for q_x is given by

$$q_x = 1 - \exp\Big\{\frac{\kappa}{\beta}\Big[\ln\big(1+\alpha e^{\beta x}\big) - \ln\big(1+\alpha e^{\beta(x+1)}\big)\Big] - \gamma\Big\}\,.$$

Letting $\kappa = 1$ or $\gamma = 0$, the appropriate formulas for q_x for the G-model and K-model, respectively, are obtained.

3 Monte Carlo Simulation Program

A program was written for running Monte Carlo simulations to compare probability of death predictions generated by two different three-parameter logistic force of mortality models. Programming was done entirely in the R programming language, using the standard R for Mac OS X Cocoa GUI, R version 2.10.1, GUI 1.31 Leopard build 64-bit (5537). Along with a number of standard R functions, new functions were created and incorporated into the coding of the simulation program. The main function of the program calls a number of supporting functions written to carry out various tasks related to the simulation. The main and supporting functions are described in the

paragraphs that follow, with each supporting function described in the logical order in which it is called by the main function. All program coding that is provided in Appendix 7.3 is for representative purposes only. Any use of the coding for other simulation studies should not be done without sufficient testing and adaptation with respect to data sources, required precision and accuracy, and other considerations. A diagram showing the overall flow of the program is given in Figure 1.

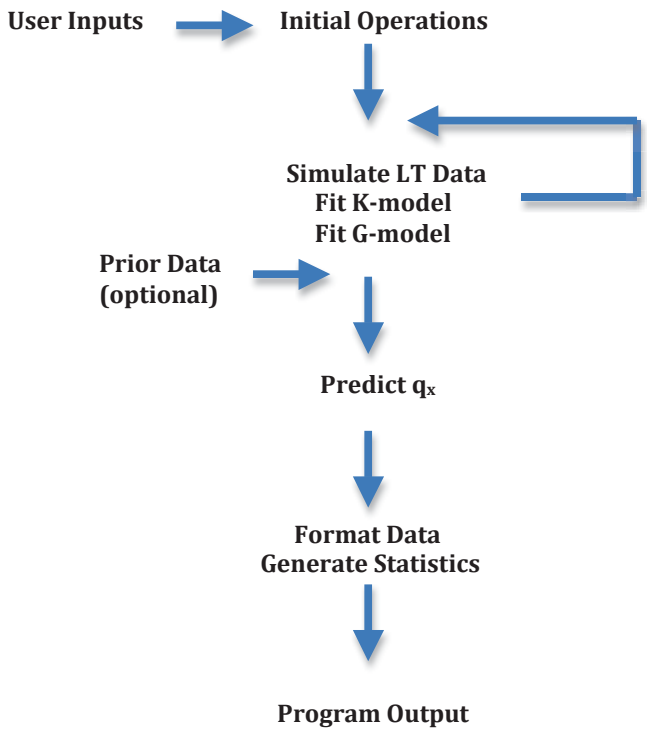

Figure 1. Overall Flow of Monte Carlo Simulation Program

3.1 User Inputs

The main function, *perksMCsim* (Perks Monte Carlo Simulation), allows for a number of user inputs when the simulation program is initially called in R. The user indicates the desired number of histories of the Monte Carlo simulation, and supplies information on the life table being used to determine the probability parameters in the parametric model that generates the simulated life table data. This involves supplying vectors of the life table ages and the number of survivors at each age. The user also chooses the age range of life table data that will be used in estimation of the parameters of the force of mortality models, as well as the age range for which

probability of death predictions will be made. For the simulation study of the present work, life table data for ages 80 to 99 was used to fit the force of mortality models, and probability of death predictions were obtained for ages 80 through 105. The user also can indicate the radix, which is the desired cohort size at the initial age of the simulated life tables. The default value for the radix is the cohort size of the supplied life table at the initial age specified for probability of death predictions. By varying the radix, comparisons of model predictions can be made at different size levels. As part of controlling the random number generator (RNG) involved in the Monte Carlo simulation, the user also enters an integer value to specify the seed for the RNG.

There are two additional inputs for use if the simulation program is being called as a continuation of one or more previous runs as part of the same Monte Carlo simulation. The user can provide an object in which the previous state of the RNG being used has been stored, so that the RNG can pick up where it left off at the end of the last run. After the simulation program has completed a run, the state of the RNG (stored in an object named "oldstate") should be saved into another object that may later be recalled if desired. The user also supplies a matrix of the raw simulation data accumulated in the previous simulation runs, which will be combined by the program with the raw simulation data from the new run, and formatted and used together to determine Monte Carlo estimates of the expected probabilities of death and approximate standard deviations of the estimates. This again requires that after the simulation program has completed a run, the matrix of raw data (stored in an object named "rawtemp") should be saved into another object that may later be recalled if desired. The feature of the Monte Carlo simulation program that has just been described allows the user to carry out a lengthy Monte Carlo study in a series of shorter runs; or, to expand upon a simulation that was initially intended to involve fewer histories. If these optional inputs are supplied, then a seed is not required for input.

3.2 Initial Operations

The program begins by setting the RNG kind to the Wichmann-Hill generator, and setting and storing the initial state of the RNG, either by the seed supplied by the user or by the prior state of the RNG supplied by the user if the simulation is a continuation of a previous run. A number of variables are set, most based on user-supplied values, and often involving some form of calculation or other operation. For example, the number of deaths, probability of death, and probability of survival by

age are calculated for the life table data supplied by the user.

One operation that occurs prior to entering the main loop of the program is the determination of parameter values to be supplied to the supporting functions that carry out NLS estimation of the K-model and G-model parameters when called within the main loop. The parameter values are simply the NLS estimates for the parameters based on the life table data supplied by the user and the range of ages specified to be used in estimation. These NLS estimates are found using the supporting functions *startfinderG* (Start Value Finder G-Model) and *startfinderK* (Start Value Finder K-Model). The operation of both functions is the same except for the difference in models, and will now be described.

The function *startfinderK* uses the R function *nls* to perform NLS estimation on the K-model. All of the default values for the *nls* controls are maintained, including a Gauss-Newton algorithm and a tolerance level of 1e-5 when determining convergence based on the relative offset convergence criterion (R documentation). The control warnOnly is at the default value of FALSE, so that an NLS model is returned by *nls* only if the convergence criterion has been satisfied (R documentation). The maximum number of iterations is the default value of 50, and the minimum step factor is the default value of 1/1024 (R documentation).

The regression equation supplied to *nls* is the equation for determining p_x based on the K-model, as was previously derived. The starting values for the parameters as supplied to *nls* are determined using three *while* loops that vary each parameter through a range of values. Different sets of starting values are thus supplied to *nls* until convergence is obtained or the ranges of values have been exhausted. Therefore the *nls* function may be called repeatedly within *startfinderK* while searching for a convergent NLS model.

When the *nls* function is called by *startfinder*K, it is done so using the *try* function in R, which allows for *startfinderK* to continue when *nls* fails to find an NLS model for a particular set of starting values. When an NLS model is not found, *try* returns an object of length one. Otherwise, the object returned is the *nls* object which has a length greater than one. This difference in object length provides the basis of the test *startfinderK* employs to determine when *nls* has found an NLS model. Once an NLS model is returned by *nls*, the sum of squares of the residuals (SSR) is calculated, and the parameter estimates are extracted and stored along with the SSR. The loops are ended and *startfinderK* returns a vector of parameter estimates and the value of the SSR, which are stored globally in the object "Kstart". Kstart is not returned by the simulation program but can be retrieved when the program finishes, before the

program is executed again, if desired.

The operation of *startfinderG* is the same as for startfinderK, and after both *startfinderG* and *startfinderK* have returned objects, which have been subsequently stored, the program moves on the main loop. It should be noted, however, that even if *startfinderG* and/or *startfinderK* do not return an NLS model(s), the program still moves to the main loop, in which NLS estimation will fail and the program will end, returning an error message, when it tries to call the function(s) that perform the NLS estimation.

3.3 Generation of Simulation Data

3.3.1 Simulated Life Table Data

The first function to be called within the primary loop is the function that generates the simulated life table data, *LTPsim* (Life Table Parametric Simulation). When the function is called, it generates data for a single simulated life table, and thus must be called during each cycle of the primary loop. The life table data generated covers the same age range as was specified by the user for obtaining probability of death predictions . *LTPsim* uses the standard function in R for random variate generation from the binomial distribution, *rbinom*, which is based on the method given by Kachitvichyanukul and Schmeiser (1988) (R Documentation). The *rbinom* function must be called repeatedly in order to generate the number of deaths at all ages being considered. After *LTPsim* finishes, returning the simulated life table data, the program saves the state of the RNG in the object "oldstate" so that it can be restored later if desired, as was previously described. The state of the RNG just prior to the execution of *LTPsim* is also saved, in the object "oldoldstate". Neither of these objects is returned by the program upon completion of the simulation, and should be saved to other objects if their preservation is desired.

LTPsim is central to the Monte Carlo simulation. It randomly generates a sample from the joint distribution used in the simulation to model cohort deaths at the specified ages, as was previously described. When drawing a simulated sample from the joint distribution of deaths at the specified ages, *LTPsim* first generates the number of deaths at age x, D_x, based on the predetermined values of N_x and q_x, and then calculates the value of N_{x+1} using D_x and N_x. The value of N_{x+1} thus found is then used along with the value of q_{x+1} from the probability model, in order to determine D_{x+1}. This same process is repeated as many times as needed to determine the number of deaths at all successive ages (please see addendum on page 51).

Thus *LTPsim* first generates a random variate from the distribution of deaths at age x, which is binomial with size parameter equal to either the cohort size at age x of the user-supplied life table, or the size specified by the user when the Monte Carlo simulation program was initially called, and with probability parameter equal to the probability of death at age x as determined from the life table supplied by the user. Once the number of deaths at age x is generated by the *rbinom* function and stored as the first element in a vector containing the simulated numbers of deaths, the cohort size at age $x+1$ can be easily calculated and stored as the second element of a vector containing the simulated numbers of survivors. A *for* loop repeats until the number of deaths and number of survivors at all necessary ages have been determined and stored in the corresponding vectors. The probability parameter of the binomial model at each age is always the same as the corresponding probability of death determined from the user-supplied life table, and the size parameter is just the number of survivors as calculated in the previous cycle of the loop.

When finished, *LTPsim* returns the vectors of the number of survivors at each age and the number of deaths at each age. Using the simulated life table data returned by *LTPsim*, the probabilities of death and probabilities of survival are calculated in the main loop, and all the vectors of simulated life table data, except the probabilities of survival, are then stored in the appropriate row of the matrix of raw data. That matrix is subsequently stored globally in a backup matrix that can be retrieved after completion of the program, for later use in a subsequent simulation if desired. To store the matrix globally for backup, the global assignment operator in R (<<-) must be used, because the ordinary assignment operator (<-) used within the main function of the program, or any supporting functions, only stores to temporary objects used when the given function is being executed. The backup matrix is named "rawtemp" and should be stored in another object after the simulation program ends, if preservation is desired. If it is not saved to another object, the next time the simulation program is run, the old backup matrix will be lost and replaced with a new one.

3.3.2 Nonlinear Least Squares Estimates

With the life table data generated for the current history of the Monte Carlo simulation, the program moves on within the main loop to NLS estimation of the parameters in the K-model and G-model. The program first performs NLS regression on the K-model, calling the supporting function *nlsfinderK* (Nonlinear Least Squares Finder K-model). The function is called within the R function *try*, just as *startfinderK* and *startfinderG* were. Thus, if the function fails in execution, the overall Monte Carlo

simulation program will still continue to execute. The inputs to this program are the vector of simulated life table probabilities of survival through each age (p_x) and the vector of corresponding life table ages, the vector of parameters values as found by *startfinderK* earlier in the program, and a vector of step values for determining the increments of change when varying the parameter start values for NLS estimation within *nlsfinder*K.

The heart of *nlsfinderK* is the same as for *startfinder*K, which was previously described. NLS regression is carried out using the R function *nls*, with all the controls at default values. Three *while* loops are used to vary the values of the three parameters in order to provide different start values for the parameters when the *nls* function is repeatedly called. If an NLS model is found, the parameter estimates are stored, and the SSR is calculated and stored. Then the while loop terminates and the parameter estimates and SSR are returned by *nlsfinder*K. The difference between *nlsfinderK* and *startfinderK* is that *nlsfinderK* uses a vector of parameter estimates supplied as input to determine the region of parameter values that will be searched when trying different start values for the parameters, whereas *startfinderK* uses vectors of minimum and maximum values of the parameters as inputs to determine the ranges of parameter values that are searched.

If *nlsfinder*K finds an NLS model, returning the parameter estimates and SSR, the program identifies that an object of length greater than one has been returned and proceeds to store the data in the matrix of raw data and backup matrix, moving on to NLS regression of the G-model. If an NLS model is not found by *nlsfinder*K, the program identifies that the object returned by *try* is length one, and the current cycle of the main loop ends without going on to try NLS regression on the G-model. The program keeps a running tabulation of the total number of NLS convergence failures for each model, and also stores the cycle numbers in which those failures occur. The function called when performing NLS regression on the G-model is *nlsfinderG* (Nonlinear Least Squares Finder G-model). The operation of this function is the same as that of *nlsfinder*K, except for the differences in the model and parameters being used. If either *nlsfinderK* or *nlsfinderG* fails to find an NLS model, the current cycle of the main loop ends and the cycle begins again, with new simulated life table data being generated. Any of the data generated in the aborted cycle is not kept in the matrix of raw data, but is stored globally in a list named "rawfails" that holds the data from failed cycles. This list is not returned with other output from the program, and should be stored to another object when the program finishes in order to preserve it, if desired.

When both the K-model and the G-model have been successfully fit to simulated life table data using NLS estimation, the loop index is incremented and one cycle of the main loop has been completed. As long as there are still histories of the simulation to complete, the next cycle of the main loop begins.

3.3.3 Probability of Death Predictions

Upon completion of all histories specified for the current run of the Monte Carlo simulation program, the program exits the main loop and checks to see if a matrix of raw data from a previous run of the program was supplied by the user when the program was initially called. If so, the *rbind* function in R is used to join the new matrix of data to the old, and the total number of histories is increased accordingly.

The probability of death predictions are found by calling the supporting function *qest* (q_x Estimator). This function computes probability of death predictions for a single vector of parameter estimates, and thus must be called separately for each three-parameter NLS model. The program calls *qest* twice (once for each model) within a *for* loop for as many repetitions as there are histories of simulation data in the matrix of raw data. The only inputs to *qest* are a vector of ages that corresponds to the range of ages for which probability of death predictions will be computed (as initially specified by the user), a vector of estimates of the model parameters, and a value of either "K" or "G" to indicate which three-parameter model is being used to make predictions. The equation used to compute the probability of death at a given age is based on that derived previously for the four-parameter logistic model, with the value of γ set to 0 when generating probability of death predictions from the K-model, and the value of κ set to 1 when generating probability of death predictions for the G-model. The formula for q_x is given again below

$$q_x = 1 - \exp\left\{ \frac{\kappa}{\beta} \left[\ln\left(1 + \alpha e^{\beta x}\right) - \ln\left(1 + \alpha e^{\beta(x+1)}\right) \right] - \gamma \right\} .$$

Upon computing probability of death predictions for the specified ages, a vector of the predictions is returned and stored in the appropriate row of one of two matrices that solely contain probability of death predictions (either the matrix for the K-model or the matrix for the G-model). Upon generating probability of death predictions for all histories of the simulation, the program moves on to process the simulation data.

3.4 Processing of Simulation Data

When all of the probability of death predictions have been computed, the program moves on to call the remaining functions that format the data, compute statistics for the probability of death predictions and standard life table probability of death estimates, and prepare the results for output. The first function called upon leaving the main loop is *MCdata* (Monte Carlo Data). This function receives the matrix of raw data and formats it, yielding an output that is in the the R *list* form. The components of the list are numbered and named as follows:

[[1]] *lsim* (Simulated life table data for survivors at each age)

[[2]] *dsim* (Simulated life table data for the number of deaths at each age)

[[3]] *qsim* (Simulated life table data for the probability of death at each age)

[[4]] *"NLSE-K"* (NLS estimate for parameters of the K-model)

[[5]] *"NLSE-G"* (NLS estimate for parameters of the G-model)

[[6]] *qK* (Probability of death predictions generated by the K-model)

[[7]] *qG* (Probability of death predictions generated by the G-model).

Each component in the list is a matrix, with each row being a vector of data generated in the numbered history of the simulation. The column names of each matrix are the ages that correspond to the data. The format of the data is the same when listed upon completion of the program, but is present as a list within a list, listed with the other information that the program generates and provides when a simulation run is completed. The list of simulation data as outlined above is labeled "simdata" in the overall list output.

With the raw data formatted as a list, the next function to be called is *MCstats* (Monte Carlo Statistics). This function computes statistics for the probability of death predictions and estimates, and is called twice–once for each model. The inputs to *MCstats* are a matrix of probability of death predictions, the life table estimates of the probabilities of death, and the vector of probabilities used in the parametric model that generated the simulated life table data. The function *MCstats* uses a *while* loop to compute statistics for the probability of death prediction and estimate at each age. The mean and standard deviation of the mean are calculated for the probability of death prediction and estimate at each age, according to the formulas

previously given, as well as the difference between the mean predicted or estimated probability of death and true probability of death at each age. When finished, the function lists vectors of the computed statistics.

The last function to be called is *MCcompare* (Monte Carlo Comparison). This function computes summary statistics that show the average magnitude of difference between the observed and predicted or estimated probabilities of death across all the ages considered, as well as the average standard deviation of the predictions or estimates. The function also formats all the statistical results for the competing probability of death predictions and the estimates so that they can be viewed in a side-by-side comparison. All columns and rows are labeled appropriately. The results are given by the function as a list of matrices (mean, stdv, difference, and summary).

When the Monte Carlo simulation program has finished processing the simulation data, it ends by returning a list of data and results that are labeled in a list as follows:

[[1]] *data* (Life table data supplied by the user)

[[2]] *simdata* (A list of 7 components of simulation data returned by *MCdata*)

[[3]] *stats* (Mean, standard deviation, difference, and summary statistics returned by *MCcompare*)

[[4]] *nlsKfailures* (The number of cycles in which no NLS model was found for the K-model)

[[5]] *which.nlsKfailed* (The cycle numbers in which NLS estimation for the K-model failed)

[[6]] *nlsGfailures* (The number of cycles in which no NLS model was found for the G-model)

[[7]] *which.nlsGfailed* (The cycle numbers in which NLS estimation for the G-model failed).

4 Simulation Study

A simulation study was conducted using the Monte Carlo simulation program described in the preceding section. Two models for predicting life table probabilities of death were compared, the K-model and G-model, each being a three-parameter logistic force of mortality model. The performance of the models was evaluated for two

different cohort size levels, using a simulated distribution of deaths with probability parameters based on a life table constructed from mortality data obtained from the U.S. Social Security Death Index, Master File (DMF) for a single birth year cohort. The life table data used will be discussed first, followed by details of the simulation, results, and discussion.

4.1 Source of Life Table Data

The parametric model employed by the Monte Carlo simulation program to generate simulated life table data uses life table data supplied by the user when the program is called in R. The supplied life table data determines the probability parameters of the binomial distributions that generate the simulated number of cohort deaths at the different ages. For the Monte Carlo study conducted as part of this research, the life table data supplied to the program was a combined male and female cohort life table of individuals born in the year 1893, constructed based on data from the DMF, which includes records of persons who were enrolled in the Social Security program and whose deaths were reported to the Social Security Administration. The information included in these records that was used in the present study was month and year of birth, and month and year of death.

In this research, the DMF records were accessed as provided in the *Social Security Death Index* (SSDI) available through Ancestry Library Edition (Source: ancestry.com, 2011). Due to the limitations of how this database could be queried, individuals who died in the same month of the year as their birth month were systematically excluded from the cohort used in this study. This was done in order that all individuals in the cohort who died between the ages of 80 and 109 could easily be classified as either dying before or after attaining their birthday in their given year of death, allowing their exact age upon death to be determined using only simple queries from the SSDI database. All individuals dying at age 110 and above were grouped into the single age category 110+, and so determination of exact age upon death for those individuals was not required. With some processing of the results in a spreadsheet, the distribution of cohort deaths by age could then be calculated. This was done for ages 80 to 110+, for deaths in the calendar years 1973 through 2011. The methodology of querying the SSDI database and processing the data in a spreadsheet is described in detail in Appendix 7.1.

Individuals who were born in 1893 and who died in 2011 would have been age 117 or 118, depending on whether they died before or after attaining their birthday. In

the 1893 cohort constructed, there were 3 deaths of such individuals in 2011. Queries of the SSDI were performed in October 2011, and therefore the number of deaths obtained for 2011 may be incomplete. However, individuals from the 1893 cohort surviving to 2011 would have been at least 117 years of age, so the number of deaths missed would likely be very small. It is also possible that some of the records of individuals with reported deaths at such old ages, as well as other ages, are the result of incorrect reporting of information related to age upon death. Finally, it is possible that there were errors in the present study when querying the SSDI and entering the data. However, the primary purpose of the cohort life table constructed in this research is to provide a realistic basis for determining the probability parameters in the parametric model used to generate simulated life table data in the Monte Carlo simulation program, in order to compare alternative models for predicting life table probabilities of death. To the extent that the cohort life table constructed in this study does not realistically represent the potential mortality experience of some population of individuals, the results of the simulation may fail to provide an accurate comparison of the performance of the estimators in a real-life scenario.

It is reasonable to assume that almost all of the individuals born in 1893 who will at some point be recorded in the DMF had already died and been recorded in the DMF by October 2011 . The number still alive would likely be a very small percentage of the number surviving to age 99 (at least 20155 individuals), and that is the oldest age of life table data to which the logistic models were fit in the Monte Carlo study of this thesis. There would, however, be a greater effect on the probability of death at older ages when the number of survivors had become significantly smaller. Thus the decision was made to limit the oldest age considered in the Monte Carlo study to 105 when comparing probability of death predictions.

There were at least 1418 surviving individuals in the cohort at age 105, and assuming that there were less than 15 surviving individuals born in 1893 whose deaths would eventually be recorded in the DMF, the effect on the estimated probability of death at age 105 would be less than 1%. It is therefore reasonable to construct a cohort life table using a method that assumes an extinct cohort (see Vincent, 1951, who is credited with the method of extinct generations, and Doray, 2002). Knowing the numbers of deaths at ages 80 through 110+, the number of individuals surviving to a given age can then be determined by summing the number of deaths at the given age and all succeeding ages. For example, to determine the population surviving to age 90, one can sum the numbers of deaths of individuals attaining 90, 91, ..., 110+ years of age. For an extinct cohort, individuals surviving to age 90 would all have

died at age 90 or above, and thus the sum just described would yield the surviving population at age 90. In a similar manner, a cohort life table based on the SSDI death data was constructed covering the ages 80 to 110+, and is given in Table 4.1.

Table 4.1 1893 SSDI Cohort Life Table

Age	Living	Deaths	q_x	Age	Living	Deaths	q_x
80	601059	46939	0.0781	96	50189	12516	0.2494
81	554120	45989	0.0830	97	37673	9619	0.2553
82	508131	44658	0.0879	98	28054	7899	0.2816
83	463473	43316	0.0935	99	20155	5996	0.2975
84	420157	43079	0.1025	100	14159	4655	0.3288
85	377078	40340	0.1070	101	9504	3254	0.3424
86	336738	40311	0.1197	102	6250	2294	0.3670
87	296427	38294	0.1292	103	3956	1587	0.4012
88	258133	34873	0.1351	104	2369	951	0.4014
89	223260	33042	0.1480	105	1418	615	0.4337
90	190218	30653	0.1611	106	803	361	0.4496
91	159565	28286	0.1773	107	442	190	0.4299
92	131279	25209	0.1920	108	252	108	0.4286
93	106070	21731	0.2049	109	144	58	0.4028
94	84339	18956	0.2248	110+	86	86	1
95	65383	15194	0.2324				

As long as one is content to consider the cohort of individuals actually obtained from the SSDI to be the population of interest, no assumptions about immigration and migration must be made. Nevertheless, immigration and migration would likely be minimal among individuals age 80 and above. In applying the method of extinct generations to construct life tables using Canadian mortality data, Doray (2002) made the assumption that half of the individuals reported to die at a given age in a calendar year were part of the cohort from one birth year and half were part of the cohort from the previous birth year. However, as already explained, a similar assumption was not required in the present study due to the method in which the cohort was obtained from the SSDI. Although individuals who died in the same month of the year as the month in which they were born were excluded from the cohort, the size of the cohort, being numbered at over 601,000 individuals alive at age 80, is still very large, and instead of employing assumptions about the true distribution of deaths of the individuals in the cohort, the actual number of individuals dying at a given age could

be determined. On the other hand, if the mortality experience of individuals who die in the same month as in which they were born differs from that of individuals who die in non-birth months, the life table constructed in this study may systematically fail to accurately represent the potential mortality experience of a population.

4.2 Simulation Details

The Monte Carlo method was employed to estimate the expected values of predicted life table probabilities of death based on two alternative models, the K-model and G-model, each being a three-parameter logistic force of mortality model. Standard deviations of the probability of death predictions were also obtained. The equations for predicting p_x were previously derived and are given again. The equation for p_x based on the G-model is given by

$$p_x = \exp\Big\{\frac{1}{\beta}\Big[\ln\big(1+\alpha e^{\beta x}\big) - \ln\big(1+\alpha e^{\beta(x+1)}\big)\Big] - \gamma\Big\}$$

while for the K-model it is given by

$$p_x = \exp\Big\{\frac{\kappa}{\beta}\Big[\ln\big(1+\alpha e^{\beta x}\big) - \ln\big(1+\alpha e^{\beta(x+1)}\big)\Big]\Big\}\,.$$

The equations for q_x are then easily found as $1 - p_x$.

By randomly generating samples of life table data from a simulated distribution of deaths by age, and obtaining life table probability of death predictions for each sample over a specified range of ages, average values of the predicted probabilities of death could be computed for each model, as well as their standard deviations and standard deviations of the means. A Monte Carlo simulation consisting of 1500 histories was used, and with this number of histories the mean probabilities of death were computed to a reasonable level of precision.

The Monte Carlo study involved two separate simulations, with the radix of the simulated life tables being varied from a larger to a smaller level so that the effect on the probability of death predictions from the models could be explored. The actual radix of the life table constructed from the DMF mortality data (601,059) was used as the radix in one simulation. This radix was roughly halved to arrive at a radix of 300,000 for the second simulation. These simulations will be referred to as the 601k and 300k simulations.

It was observed that with decreasing radix, there arose issues of NLS estimation failures and negative values for parameter estimates for some of the samples of simu-

lated life table data. For the 601k simulation, there were no NLS estimation failures or negative parameter estimates. However, for the 300k simulation there were 4 NLS estimation failures, for a failure rate of 0.3%($\frac{4}{1504} = 0.003$). When the sample size was reduced to 150k, a Monte Carlo simulation of 1500 histories included 25 NLS estimation failures, for a failure rate of 1%($\frac{15}{1525} = 0.01$). There were no negative parameter estimates in the 300k simulation, but in the 150k simulation, the K-model was twice fit with negative values for both κ and α, and the G-model was fit 10 times with a negative value for γ. For a sample size of 100k and a Monte Carlo simulation of only 100 histories, there were already 5 NLS estimation failures, for roughly a 5% failure rate. Though few in number and small in percentage, it should be considered that the 4 NLS estimation failures in the 300k simulation might have some effect on the results. This will be brought up again in a later section.

The Monte Carlo simulation program allows for specification of the age range of life table data to be used when fitting the logistic models, as well as the age range for which the models will be used to make probability of death predictions. The simulations carried out in this study used simulated life table data for ages 80 to 99 when estimating the parameters of the logistic models using the method of nonlinear least squares, and generated life table probability of death predictions for ages 80 to 105. Although the Monte Carlo simulation program provides the capability of varying the age range used when fitting the models, it was found that problems can arise when trying to fit the models to certain ranges of age. For example, when the age range of 80-90 years was used for fitting the models in a brief simulation with only 25 histories, the NLS estimates for κ and α in the K-model were negative in all histories. Difficulties also can arise in obtaining convergence in NLS estimation. For the present study, the age range used in estimation did not result in any negative parameter estimates for the 601k or 300k simulations, but as was previously mentioned, when the radix was reduced to 150k, some of the histories of the simulation yielded negative values for some of the parameters. The potential for the models to be fit with negative estimates for certain parameters is an issue that would seem to merit further investigation, though this issue is deferred from the present study.

A question might be raised concerning the reliability of the NLS estimates obtained in automated fashion by the Monte Carlo simulation program for the simulations carried out in the present study. An investigation was made using slightly altered versions of *startfinderG* and *startfinderK*. The testing versions of these functions, *startfinderGtest* and *startfinderKtest*, do not stop when the first NLS model is obtained, but rather continue to range through the parameter space, trying NLS

estimation with every different set of parameter start values generated by the three *while* loops. The result of every attempt is stored in a list, either NA if the particular set of parameter start values does not yield an NLS model when the function *nls* is called, or else the NLS parameter estimates and SSR if an NLS model is obtained. It is then possible to analyze the multiple NLS models (often very many) obtained by either *startfinderGtest* or *startfinderKtest* and examine the consistency of the results. Another function was written, *nlstest*, that provides for processing of the data produced by *startfinderGtest* or *startfinderKtest*. This function counts the number of NLS estimation failures, as well as the number of NLS models obtained, and compares the value of the SSR for each NLS model. The function *nlstest* also generates statistics for the NLS estimate of each parameter, including the minimum, maximum, range, mean, and standard deviation. Coding for *startfinderGtest*, *startfinderKtest*, and *nlstest* is provided in Appendix 7.5.

The testing functions just described were used to test the NLS estimates based on the supplied life table, and selected simulated life tables generated in the 601k and 300k simulations. Results of tests using both the K-model and G-model with the following simulated life tables are presented in Appendix 7.4: the simulated life table for the first history of the simulations, the simulated life table for which the minimum NLS estimate for the parameter γ was generated in each simulation, the simulated life table for which the maximum NLS estimate for γ was generated in each simulation, the simulated life table for which the minimum NLS estimate of κ was generated in each simulation, and the simulated life table for which the maximum NLS estimate of κ was generated in each simulation (other life tables may also have been tested but using a different version of *nlstest* than given in Appendix 7.4, or whose results were not maintained). Based on the results of the tests given in Appendix 7.4, it was found that when there was convergence in NLS estimation for a particular life table, the NLS models always had the same SSR, to a precision of at least 7 significant figures (10 decimal places). Furthermore, the NLS estimates of the parameters were the same across the NLS models, with allowance for some spread in the distributions of the estimates. These results provide a level of assurance in the validity of the NLS estimates obtained in automated fashion during the simulations. It indeed seems to be one benefit of using NLS estimation in the application of the present study that either NLS estimation failed to produce an NLS model or else it apparently found the correct one.

The 601k and 300k Monte Carlo simulations were carried out with the integer 3 supplied to the RNG as the seed in both simulations. The Monte Carlo simulation

program was able to perform the simulations in a short period of time, requiring only about 11 minutes for one simulation and about 12 minutes for the other. Selected data generated by the simulation program for the 601k and 300k simulations is provided in Appendix 7.4. The statistics generated for the probability of death predictions and parameter estimates in both simulations are included, as well as information on NLS estimation failures, and a sample of simulation data for the first 5 of the 1500 histories of the 601k simulation.

4.3 Results

Results of the 601k Monte Carlo simulation are presented in the tables that follow. The mean predicted q_x were nearly identical for the 601k and 300k simulations, although the standard deviations of the mean q_x were systematically smaller for the 601k simulation, with all the values being between 68% and 74% of the magnitude of the standard deviations obtained from the 300k simulation. Table 4.3.1 presents first the standard deviations of the Monte Carlo estimates of the expected values of q_x, or in other words, the standard deviations of the means calculated from the simulation. This table provides a gauge for evaluating the uncertainty of the Monte Carlo estimates. Though not shown, the relative error, R, was also computed for the Life Table, K-model, and G-model standard deviations for each age in the 601k simulation, as given in Table 4.3.1 and found to be well under 0.05 in all cases (between 0.00004 and 0.0008).

Table 4.3.1 Standard Deviations of the Mean q_x for 601k Simulation

	80	81	82	83	84	85	86	87	88
Life Table	9.0E-06	9.7E-06	1.0E-05	1.1E-05	1.2E-05	1.3E-05	1.5E-05	1.6E-05	1.7E-05
K-Model	1.0E-05	8.6E-06	6.9E-06	5.3E-06	4.3E-06	4.3E-06	5.4E-06	7.0E-06	8.7E-06
G-Model	1.1E-05	7.8E-06	5.3E-06	4.0E-06	4.7E-06	6.2E-06	7.8E-06	9.1E-06	9.9E-06
	89	90	91	92	93	94	95	96	97
Life Table	2.0E-05	2.2E-05	2.4E-05	2.8E-05	3.3E-05	3.7E-05	4.4E-05	5.0E-05	5.7E-05
K-Model	1.0E-05	1.1E-05	1.2E-05	1.2E-05	1.2E-05	1.3E-05	1.5E-05	1.9E-05	2.7E-05
G-Model	1.0E-05	1.0E-05	9.7E-06	9.6E-06	1.0E-05	1.3E-05	1.6E-05	2.2E-05	2.8E-05
	98	99	100	101	102	103	104	105	
Life Table	7.0E-05	8.7E-05	1.0E-04	1.2E-04	1.6E-04	2.0E-04	2.6E-04	3.4E-04	
K-Model	3.8E-05	5.2E-05	7.0E-05	9.0E-05	1.1E-04	1.4E-04	1.7E-04	2.1E-04	
G-Model	3.5E-05	4.3E-05	5.1E-05	5.9E-05	6.8E-05	7.6E-05	8.4E-05	9.1E-05	

It may be observed that for the life table probabilities of death, the standard deviation of the mean consistently increases with age. For the K-model and G-model the standard deviation of the mean decreases at first, from age 80 to age 83 or 84, but then either increases or stays steady as age increases, except for the decreases with the G-model between ages 90 and 92. Trends in the standard deviations of the means in the 300k simulation were similar as those described above, except that for the life table probabilities of death, the standard deviation of the mean remained steady at 1.3E-05 between ages 80 and 81, for the K-model the standard deviation of the mean decreased from age 80 all the way to age 85, and for the G-model the standard deviation of the mean remained steady at 1.4E-05 between ages 90 and 92.

Results of the 601k simulation comparing the actual values of q_x with those predicted by the K-model and G-model are presented in Table 4.3.2. Estimates of the standard deviations of the predictions, and estimates of the differences between the true and predicted values of q_x are also given. Estimates of the standard deviations of the predictions were calculated by multiplying the standard deviations of the mean q_x by the square root of the number of histories in the simulation ($\sqrt{1500}$). The standard deviations of the mean q_x as given in Table 4.3.1 were used to determine the maximum number of decimal places to which the results of the simulations are reported in Table 4.3.2, following the method recommended by Davidian (2005). For a result to be reported to 3 decimal places, the standard deviation of the mean was required to be 5.0E-04 or less. For a result to be reported to 4 decimal places, the standard deviation of the mean was required to be 5.0E-05 or less. All results in Table 4.3.2 are reported to either 3 or 4 decimal places of precision. The goal of this method is to place a 95% confidence interval not larger than ± 1 around the digit in the last reported decimal place (Davidian, 2005). Dunn and Shultis (2012) discuss a statistic called the variance of the variance, which may be calculated from the results of a Monte Carlo simulation, and which they report as having been shown in a work produced by Los Alamos National Laboratories (MCNP, 2003) to be connected to the reliability of the confidence intervals constructed for a Monte Carlo simulation. However, the variance of variance was not analyzed in the present study.

Table 4.3.2 Comparison of Model Predictions in the 601k Monte Carlo Simulation

	80			81			82		
	Mean	SD	Diff	Mean	SD	Diff	Mean	SD	Diff
TRUE	0.0781			0.0830			0.0879		
Life Table	0.0781	0.0004	0.0000	0.0830	0.0004	0.0000	0.0879	0.0004	0.0000
K-Model	0.0735	0.0004	-0.0046	0.0800	0.0003	-0.0030	0.0869	0.0003	-0.0010
G-Model	0.0751	0.0004	-0.0029	0.0811	0.0003	-0.0019	0.0876	0.0002	-0.0003

	83			84			85		
	Mean	SD	Diff	Mean	SD	Diff	Mean	SD	Diff
TRUE	0.0935			0.1025			0.1070		
Life Table	0.0935	0.0004	0.0000	0.1025	0.0005	0.0000	0.1070	0.0005	0.0000
K-Model	0.0944	0.0002	0.0009	0.1024	0.0002	-0.0001	0.1110	0.0002	0.0040
G-Model	0.0947	0.0002	0.0012	0.1024	0.0002	-0.0001	0.1107	0.0002	0.0037
	86			87			88		
	Mean	SD	Diff	Mean	SD	Diff	Mean	SD	Diff
TRUE	0.1197			0.1292			0.1351		
Life Table	0.1197	0.0006	0.0000	0.1292	0.0006	0.0000	0.1351	0.0007	0.0000
K-Model	0.1202	0.0002	0.0005	0.1300	0.0003	0.0009	0.1405	0.0003	0.0054
G-Model	0.1197	0.0003	0.0000	0.1294	0.0004	0.0002	0.1398	0.0004	0.0047
	89			90			91		
	Mean	SD	Diff	Mean	SD	Diff	Mean	SD	Diff
TRUE	0.1480			0.1611			0.1773		
Life Table	0.1480	0.0008	0.0000	0.1611	0.0009	0.0000	0.1773	0.0009	0.0000
K-Model	0.1516	0.0004	0.0036	0.1634	0.0004	0.0022	0.1758	0.0005	-0.0015
G-Model	0.1509	0.0004	0.0029	0.1627	0.0004	0.0016	0.1753	0.0004	-0.0020
	92			93			94		
	Mean	SD	Diff	Mean	SD	Diff	Mean	SD	Diff
TRUE	0.1920			0.2049			0.2248		
Life Table	0.1920	0.0011	0.0000	0.2049	0.0013	0.0000	0.2247	0.0014	0.0000
K-Model	0.1889	0.0005	-0.0031	0.2026	0.0005	-0.0023	0.2170	0.0005	-0.0078
G-Model	0.1885	0.0004	-0.0035	0.2025	0.0004	-0.0024	0.2171	0.0005	-0.0077
	95			96			97		
	Mean	SD	Diff	Mean	SD	Diff	Mean	SD	Diff
TRUE	0.2324			0.2494			0.2553		
Life Table	0.2324	0.0017	0.0000	0.2494	0.0019	0.0000	0.255	0.002	0.000
K-Model	0.2320	0.0006	-0.0004	0.2476	0.0007	-0.0018	0.2637	0.0011	0.0084
G-Model	0.2322	0.0006	-0.0001	0.2480	0.0008	-0.0014	0.2641	0.0011	0.0088
	98			99			100		
	Mean	SD	Diff	Mean	SD	Diff	Mean	SD	Diff
TRUE	0.2816			0.297			0.329		
Life Table	0.281	0.003	0.000	0.298	0.003	0.000	0.329	0.004	0.000
K-Model	0.2803	0.0015	-0.0013	0.297	0.002	0.000	0.315	0.003	-0.014
G-Model	0.2807	0.0014	-0.0009	0.298	0.002	0.000[1]	0.315	0.002	-0.014
	101			102			103		
	Mean	SD	Diff	Mean	SD	Diff	Mean	SD	Diff
TRUE	0.342			0.367			0.401		
Life Table	0.343	0.005	0.000	0.367	0.006	0.000	0.401	0.008	0.000
K-Model	0.333	0.003	-0.010	0.351	0.004	-0.017	0.369	0.005	-0.032
G-Model	0.332	0.002	-0.011	0.349	0.003	-0.018	0.366	0.003	-0.035

	104			105		
	Mean	SD	Diff	Mean	SD	Diff
TRUE	0.401			0.434		
Life Table	0.402	0.010	0.000	0.434	0.013	0.000
K-Model	0.387	0.007	-0.015	0.405	0.008	-0.029
G-Model	0.383	0.003	-0.019	0.399	0.004	-0.035

[1]This result could be reported to 4 decimal places but only 3 are used for consistency of comparison

The simulated life table q_x demonstrate a very close correspondence to the true values of q_x, differing at most by one digit in the third decimal place. This provides a gauge of how well the Monte Carlo sample of 1500 simulated life tables represents the distribution from which they were supposed to be generated. We may also observe that the mean predicted probabilities of death are very similar overall for the K-model and the G-model. The ratio of K-model predicted q_x to G-model predicted q_x varies only between 0.98 and 1.02, and is unity to two decimal places for 18 of the 26 ages where predictions were made. In the age range of 80-99, the ratio varies between 0.98 and 1.01, and in the age range 100-105, the ratio ranges between 1.00 and 1.02.

For ages 80-99, the ratio of K-model to G-model standard deviation of predicted q_x ranges between 0.69 and 1.36, and gives evidence of increasing and decreasing trends in certain age intervals. At 9 of these ages, the K-model standard deviations are larger, with the ratio ranging between 1.09 and 1.32. At 9 of these ages, the G-model standard deviations are larger, with the ratio of G-model to K-model standard deviation ranging between 1.03 and 1.46. For two of these ages the ratio of K-model to G-model standard deviation is very close to one (0.99 and 1.00). For ages 100-105, the ratio of K-model to G-model standard deviation grows rather steadily from 1.36 at age 100 up to 2.26 at age 105, where the standard deviations of the K-model are then more than twice as large as those of the G-model. There is in fact an increasing trend in the ratio all the way from age 96 to 105, growing from 0.89 to 1.22 between ages 96 and 99, and then to 2.26 by age 105. It is also interesting to observe that the standard deviations for the life table q_x are larger than for either of the models, except at age 80.

The similarities in predicted q_x for the K-model and G-model are reflected in their generally similar deviations from the true q_x, though the magnitude of the ratio of K-model deviation to G-model deviation varies much more from unity than did the ratio of their predictions, ranging between 0.74 and 1.6 for most of the ages, but also reaching magnitudes of 3, 4, and 4 at ages 95, 82, and 87 respectively, where the very close fit of the G-model stands out. At age 86, the G-model predicted q_x

did not deviate from the true q_x to the reported precision of 4 decimal places, and therefore the ratio is not included above. If that ratio were to be calculated using 7 significant digits for each of the deviations, it would be very high (1087). This points to the extremely close fit of the G-model at that age. The ratio at age 99 is also not included above, where the K-model predicted q_x did not deviate from the true q_x to the reported precision of 3 decimal places, though the G-model deviation could have been reported to 4 decimal places, being 0.0001. If the magnitude of the ratio were to be calculated using 7 significant digits for each of the deviations, it would be 2.2, again in favor of the G-model. Alternatively, we may compare the ratio of the larger deviation to the smaller deviation at each age. These ratios vary between 1.0 and 1.6 for most of the ages, except at ages 82, 86, 87, 95, and 99, which cases were discussed separately above. The close fit of the simulated life table q_x to the true q_x is readily observed, as all deviations are zero to the reported number of decimal places of the deviations.

4.4 Discussion

Overall, the K-model and G-model generated probabilities of death that were very similar, but usually deviated somewhat from the true q_x. When both models deviated from the true q_x at a given age (to the reported level of precision), it was in the same direction. At age 86 the G-model did not deviate from the true q_x to a precision greater than 4 significant figures, while the K-model slightly overestimated q_x. Also, at age 99, where K-model's predicted q_x is expressed only to 3 decimal places due to the magnitude of the standard deviation of the mean (5.2E-05), the K-model did not deviate from the true q_x to 3 decimal places while the G-model very slightly overestimated q_x in the fourth decimal place. However, if the K-model deviation were to be reported to 4 decimal places, it would be an underestimation of q_x. Excluding those two ages, the models tended to either both underestimate or both overestimate q_x.

As a way of further analyzing the results of the 601k simulation, average values of the magnitude of difference, percentage difference, and standard deviation of the predicted q_x were calculated for the G-model and K-model for two different age ranges: 80-99 and 100-105. The calculations were carried out using more digits than reported in Table 4.3.2. The results are given in Table 4.4.1.

Table 4.4.1 Comparison of Average Results of Models for Two Age Ranges

	80-99			100-105		
Average	| Diff |	% | Diff |[1]	SD	| Diff |	% | Diff |	SD
K-model	0.0026	1.8	0.00055	0.019	5.0	0.0051
G-model	0.0023	1.5	0.00053	0.022	5.6	0.0028

[1] % | Diff | = 100[| Diff | ÷ (true q_x)]

The average magnitude of difference and percentage difference are similar for the K-model and G-model in both age ranges, as is the average standard deviation in the age range of 80-99. However, in the age range of 100-105, the higher average standard deviation of the K-model stands out, though the statistical significance of the difference has not been tested. In the age range of 80-99, the average percentage deviation is less than 2% for each model, and from the age by age results previously given in Table 4.3.2, it can be seen that the deviations most often show up in the third decimal place of the differences, though sometimes not until the fourth decimal place, or not even there. In the age range of 100-105, the average percentage deviation is 5% and 5.6% for the K-model and G-model respectively, and the deviations always show up in the second decimal place of the differences (see Table 4.3.2). This may suggest a difference in model accuracy between the 80-99 age range and 100-105 age range, though the statistical significance has not been tested. We may note that the age range of 100-105 is the oldest age range as well as the age range in which the models were extrapolated, and the extent to which these two distinct factors may have contributed to a possibly significant increase in model inaccuracy cannot be ascertained from the results of the present simulations alone.

By possible way of comparison with the results of the present study, we may observe the results of Doray (2008) when fitting a simpler two-parameter logistic model (known as the Kannisto model), expressed in a linearized form and using a slight approximation, to Canadian mortality data. He fit the model separately to male and female cohort life tables for the combined birth years of 1888-1892, within the age range of 80 to 99. Results from Doray's study and the present study are given for comparison in Table 4.4.2, and average results for the age ranges 80-94 and 95-99 are given in Table 4.4.3.

Table 4.4.2 Comparison of Logistic Model Fits from Two Studies–I

	Doray Study						Present Study		
	Male			Female				G-model	K-model
Age	q_x	Predicted	Diff[1]	q_x	Predicted	Diff	q_x	Diff	Diff
80	0.0959	0.0958	0.0001	0.0643	0.0641	0.0002	0.0781	-0.0029	-0.0046
81	0.1017	0.1033	-0.0016	0.0690	0.0701	-0.0011	0.0830	-0.0019	-0.0030
82	0.1125	0.1113	0.0012	0.0779	0.0767	0.0012	0.0879	-0.0003	-0.0010
83	0.1212	0.1198	0.0014	0.0859	0.0838	0.0021	0.0935	0.0012	0.0009
84	0.1308	0.1287	0.0021	0.0935	0.0914	0.0021	0.1025	-0.0001	-0.0001
85	0.1384	0.1382	0.0002	0.1009	0.0996	0.0013	0.1070	0.0037	0.0040
86	0.1483	0.1481	0.0002	0.1094	0.1084	0.0010	0.1197	0.0000	0.0005
87	0.1579	0.1586	-0.0007	0.1156	0.1178	-0.0022	0.1292	0.0002	0.0009
88	0.1648	0.1695	-0.0047	0.1240	0.1279	-0.0039	0.1351	0.0047	0.0054
89	0.1716	0.1810	-0.0094	0.1335	0.1385	-0.0050	0.1480	0.0029	0.0036
90	0.1900	0.1928	-0.0028	0.1464	0.1498	-0.0034	0.1611	0.0016	0.0022
91	0.2003	0.2051	-0.0048	0.1566	0.1618	-0.0052	0.1773	-0.0020	-0.0015
92	0.2149	0.2178	-0.0029	0.1722	0.1743	-0.0021	0.1920	-0.0035	-0.0031
93	0.2320	0.2309	0.0011	0.1851	0.1875	-0.0024	0.2049	-0.0024	-0.0023
94	0.2505	0.2443	0.0062	0.2021	0.2012	0.0009	0.2248	-0.0077	-0.0078
95	0.2705	0.2580	0.0125	0.2214	0.2154	0.0060	0.2324	-0.0001	-0.0004
96	0.2787	0.2720	0.0067	0.2371	0.2301	0.0070	0.2494	-0.0014	-0.0018
97	0.2995	0.2861	0.0134	0.2519	0.2453	0.0066	0.2553	0.0088	0.0084
98	0.3277	0.3003	0.0274	0.2729	0.2608	0.0121	0.2816	-0.0009	-0.0013
99	0.3232	0.3147	0.0085	0.3004	0.2766	0.0238	0.2975	0.0001	-0.0001[2]

Source: Doray (2008) and present study [1]Differences were not given in Doray (2008) but were calculated to use in this table [2]Four decimal places are used here for consistency with Doray study

Table 4.4.3 Average Results from Two Studies

		Doray Study		Present Study	
	Averages	Male	Female	G-model	K-model
80-94	| Diff |	0.0026	0.0023	0.0023	0.0027
	% | Diff |[1]	1.5	1.8	1.7	2.1
95-99	| Diff |	0.014	0.011	0.0023	0.0024
	% | Diff |	4.5	4.1	0.88	0.94

Source: Averages computed based on results from Doray (2008) and present study [1]% | Diff | = 100[| Diff | ÷ (true q_x)]

We may first note that the q_x for the combined male and female cohort of the present study are between the values of q_x for the female and male cohorts of Doray's study, except at age 99. Comparing the fits of the model in Doray's study with the fit of the K-model and G-model in the present study, we might conclude that in the age range 80-94 the three models appear to have an overall comparable goodness of fit to the data. For ages 80-94, the average magnitude of difference for the two-parameter model of Doray's study is 0.0026 and 0.0023 respectively when fit to the Canadian male and female life tables, while for the G-model and K-model fit to the simulated combined life tables of the present Monte Carlo study it is 0.0027 and 0.0023, respectively. The average percentage difference for the G-model (1.7) is between that of the two-parameter model fit to the male cohort data (1.5) and female cohort data (1.8), though for the K-model it is somewhat higher (2.1). On the other hand, in the older age range of 95-99, the fits of the K-model and G-model to the simulated combined life tables are noticeably better overall than that of Doray's linear two-parameter model fit to either the male or female life tables in his study. In that age range, the average magnitude of difference for the two-parameter model fit to the male and female life tables is 0.014 and 0.011, respectively, while for the G-model and K-model it is only 0.0023 and 0.0024, respectively. The average percentage differences are similarly much lower for the G-model (0.88) and K-model (0.94) as compared to the two-parameter model fit to either the male life table (4.5) or female life table (4.1). These calculations were carried out using four decimal places for all differences.

Unfortunately, there is no data past age 99 in Doray's study for comparison. A word of caution regarding the comparisons is in order, however, since in Doray's study the model was fit to separate life tables for males and females, using different mortality data that involved a combined 5 year birth cohort, and in which the q_x were observed from a specific sample, whereas in the present study the models were fit to many simulated life tables, with the reference q_x being the known, true values of the probability model used to generate the mortality data. It would be interesting in a future study to compare the fit of the K-model and G-model with the fit of Doray's version of the Kannisto model, as well as the original nonlinear version, under identical conditions, as was done with the K-model and G-model in the present study.

Further comparisons may be attempted by considering a comprehensive study by Thatcher et al (1998) in which both a four-parameter logistic model and a two-parameter logistic model (along with others) were fit to mortality data combined from a number of industrialized nations for the periods of 1960-70, 1970-80, and 1980-90, as

well as for male and female cohorts from the combined birth years of 1871-80. For the present comparison, only the fit of the models to the cohort data will be considered. The 1871-1880 male and female cohorts were constructed from data for 11 nations, all in Europe except Japan, and included 3.2 million males and 4.8 million females (Thatcher et al, 1998). Thatcher et al fit a four-parameter logistic model of the form

$$\mu_x = c + \frac{ae^{bx}}{1 + \sigma^2 \frac{a}{b}(e^{bx} - 1)}$$

and a two-parameter (Kannisto) model of the form

$$\mu_x = \frac{ae^{bx}}{1 + a(e^{bx} - 1)} \ .$$

The models were fit to the male and female cohort life tables in the age range of 80-98 using maximum likelihood estimation, and model predictions for q_x were generated for ages 80-120. Thatcher et al provide a table that lists the deviations of the model predictions from the observed q_x for ages 99-109. The deviations for ages 100-105 found by Thatcher et al, along with those found in the present study, are shown in Table 4.4.4.

Table 4.4.4 Comparison of Logistic Model Fits from Two Studies—II

	Thatcher et al Study						Present Study		
	Male	4-Par	2-Par	Female	4-Par	2-Par	Combined	G-mod	K-mod
Age	q_x	Diff	Diff	q_x	Diff	Diff	q_x	Diff	Diff
100	0.4568	-0.0388	-0.0353	0.3971	-0.0148	-0.0089	0.3288	-0.014	-0.014
101	0.4622	-0.0302	-0.0261	0.407	-0.0100	-0.0031	0.3424	-0.011	-0.010
102	0.4547	-0.0092	-0.0045	0.4191	-0.0076	0.0005	0.3670	-0.018	-0.017
103	0.4941	-0.0357	-0.0304	0.4332	-0.0080	0.0013	0.4012	-0.035	-0.032
104	0.4506	0.0199	0.0259	0.4392	-0.0009	0.0097	0.4014	-0.019	-0.015
105	0.5199	-0.0379	-0.0313	0.4878	-0.0372	-0.0251	0.4337	-0.035	-0.029
Avg |Diff|[1]		0.029	0.026		0.013	0.008		0.022	0.019
Avg % |Diff|		6.0	5.4		2.9	1.8		5.6	5.0

Source: Thatcher et al (1998) and present study [1]Average | Diff | and % | Diff | calculated using differences expressed to four decimal places in both studies

It can be readily seen that the observed q_x at ages 100-105 for both the male cohort and the female cohort in the study by Thatcher et al are substantially higher than those for the combined cohort constructed in the present study. Also, the fit of the four-parameter logistic model appears to be substantially better for the female

cohort than for the male cohort, with the same holding true for the two-parameter logistic model.

Overall, the average magnitude of difference for the G-model (0.019) and K-model (0.022) is between that of the four-parameter model fit to the male cohort data (0.029) and female cohort data (0.013). The same holds true when compared to the two-parameter model fit to the male cohort data (0.026) and female cohort data (0.008). The average percentage difference for the G-model (5.6) and K-model (5.0) is similar to that of the four-parameter model (6.0) and two-parameter model (5.4) fit to the male cohort data, but is substantially higher than that of either the four-parameter model (2.9) or two-parameter model (1.8) fit to the the female cohort data. Some similarities in the fit of the four or two-parameter models in the study by Thatcher et al when compared to the fit of the G-model and K-model in the present study are shown in bold in Table 4.4.4, and described below.

The absolute deviations of the three-parameter logistic models are nearly the same at ages 100-101 as the deviations for the four-parameter model fit to the female cohort data, while at age 103 they are similar to the deviations of both the four-parameter and two-parameter models fit to the male cohort data. At age 104, the magnitudes of the deviations of the three-parameter models are similar to the deviation of the four-parameter model fit to the male cohort data, but the three-parameter models underestimated q_x while the four and two-parameter models overestimated q_x. At age 105, the deviation of the G-model is similar to the deviations of the four-parameter models fit to either male or female cohort data, while the deviation of the K-model is more similar to the deviations of the two-parameter model fit to the male or female cohort data. At age 102, however, the similarities break down, as the deviations of the three-parameter models are around 2-4 times larger than those of the four and two-parameter models fit to the male cohort data, and very much larger than those of the four and two-parameter models fit to the female cohort data.

It should be noted that the age by age comparisons are based on absolute deviations and not percentage deviations. It should also be cautioned again that the above comparisons are limited by differences between the present study and that of Thatcher et al, including the use of combined versus gender-specific life tables; differences in the mortality data used to fit the models, including that Thatcher et al used a cohort combined from 10 birth years while the present study used a cohort from a single birth year, and that Thatcher et al combined mortality data from a number of nations while the present study used mortality data only from the Social Security DMF. There are other differences as well, including that Thatcher et al fit

the models using maximum likelihood estimation to data for ages 80-98 as compared to ages 80-99 in the present study, and found estimates based on a single male or female life table as compared to the many simulated life tables of the present study.

Another aspect of the present Monte Carlo study that may be explored is to consider the values estimated for the parameters of the K-model and G-model throughout all histories of the simulations. Although the Monte Carlo simulation program does not calculate statistics for the parameter estimates, it does return matrices containing the parameter estimates for all histories of the simulation. Post-processing of the data can thus be done to calculate summary statistics for the parameter estimates. This was done in R for the simulations conducted in the present study, and the summary statistics for both the 601k and 300k simulations are presented in Table 4.4.5. The R coding used to generate the statistics is provided in Appendix 7.4.

Table 4.4.5 Statistics for Parameter Estimates Generated in Simulations

601k	K-model			G-model		
	alpha x 10^5	beta	kappa	alpha x 10^5	beta	gamma
Minimum	2.24	0.086	0.91	0.45	0.099	0.0050
Maximum	3.56	0.101	3.56	2.59	0.117	0.0280
Range	1.32	0.015	2.64	2.14	0.017	0.0230
Mean	3.19	0.093	1.47	1.08	0.108	0.0174
SD	0.15	0.002	0.31	0.28	0.002	0.0034
300k	K-model			G-model		
	alpha x 10^5	beta	kappa	alpha x 10^5	beta	gamma
Minimum	1.35	0.083	0.83	0.39	0.098	0.0008
Maximum	3.81	0.103	7.08	3.02	0.118	0.0297
Range	2.46	0.020	6.25	2.63	0.020	0.0290
Mean	3.13	0.093	1.52	1.13	0.108	0.0171
SD	0.25	0.003	0.55	0.39	0.003	0.0047

It may be noted that the mean values of the parameter estimates are very similar for the 601k and 300k simulations, but that the spread of the estimates for each of the parameters is greater in the 300k simulation. The increase in the range of α and percentage increase in the standard deviation of α are substantially greater for the K-model than for the G-model. The greater variability in the parameter estimates when fitting the models to life tables with a smaller radix may be a possible reason for the failures in NLS estimation as the radix is decreased, as was previously mentioned.

Focusing on the 601k simulation, we may note that the parameter α shows more variability in the G-model than in the K-model, with a standard deviation almost twice as large for the G-model. The mean value of α, however, is about three times greater for the K-model than for the G-model. The variability of the parameter β is similar for the K-model and G-model, with the ranges being close to the same and the ratio of K-model to G-model standard deviation being 0.96, but the distribution is shifted to larger values for the G-model, in which the mean value of β is 0.108 as compared to 0.093 for the K-model. The parameter κ shows a large degree of variability, as does the parameter γ. The standard deviation of κ is 21% of its mean, and similarly for γ the standard deviation is 20% of its mean. Although Thatcher (1999) states that a 1998 study by Thatcher et al found the value of κ to be near unity (based on fitting a four-parameter logistic model as was previously given), the simulations in the present study suggest that at least for the three-parameter K-model, κ can vary substantially depending on the sample of life table data, and the value may not always be close to one for a given cohort of individuals.

A comparison between the mean parameter estimates found for the G-model in the present study and the results of a study by Bongaarts (2005) may also be explored. In his study, Bongaarts fit the G-model to mortality data from 11 European nations, Japan, Canada, and the U.S. He did not use combined data as in the study by Thatcher et al (1998), but rather kept the data separate for each nation. The mortality data used in Bongaart's study is for the years 1950-2000, except for some missing years for some of the countries (Bongaarts, 2005; see article for reference of the original source of mortality data). The G-model was fit using NLS estimation, for the age range 25-109, separately for males and females, to mortality data for each separate year, and the average parameter estimates for each country were then calculated. Bongaarts provides a table that includes the average values of the parameter estimates for each country, and the average of all the means, for both males and females. The minimum, maximum, and average over all the nations' averages are given in Table 4.4.6, along with the values estimated for the U.S. mortality data.

Comparing the statistics for the G-model parameters in Table 4.4.5 to the values for the U.S. in Table 4.4.6, we may observe that the mean value of α in both simulations (1.08E-05 and 1.13E-05 for 601k and 300k simulations, respectively) is close to, but slightly higher than, the average value of α estimated for females of 14 industrialized nations (1.01E-05), but is only about half the average value for U.S. females (2.18E-05) and well below that for U.S. males (6.36E-05). The mean value of β for the simulations (0.108) falls between the average value of β for females (0.114)

and males (0.105) for the 14 nations, and is very close to the average value for males, though it is higher than the average value of β for both U.S. males (0.094) and females (0.101). The mean value of γ in both simulations (0.0174 and 0.0171 for 601k and 300k simulations, respectively) is much larger than for either males (0.00075) or females (0.00046) of the 14 nations, as well as for U.S. males (0.00087) and females (0.00042).

Table 4.4.6 Statistics for Parameter Estimates of G-model
Fit to Mortality Data for 14 Industrialized Nations

	alpha x 10^5		beta		gamma	
	Females	Males	Females	Males	Females	Males
Minimum	0.62	1.48	0.101	0.094	0.00027	0.00032
Maximum	2.18	6.36	0.120	0.112	0.00093	0.00104
Average	1.01	3.12	0.114	0.105	0.00046	0.00075
United States	2.18	6.36	0.101	0.094	0.00042	0.00087

Source: Results from Bongaarts, 2005

In making these comparisons, we must again remember that we are comparing parameter estimates for combined male and female mortality data used in the simulation study with gender-specific mortality data used by Bongaarts. Also, the simulation study used cohort mortality data, for a single birth year, whereas the study by Bongaarts used annual mortality data, and for a number of years. Furthermore, the G-model was fit to a much wider age range in the study by Bongaarts (25-109). Also, the present results are based on many simulated data sets whereas the results in Bongaart's study are not. The much higher values of γ estimated in the present study do raise the question as to possible causes. The mean values of α and β both fall between the average values of the parameter estimates for males and females of the 14 nations, which may provide some encouragement, even though they differ from the values specific to the U.S.

5 Conclusions

In this research, a Monte Carlo simulation program was developed and utilized in a simulation study comparing the life table probability of death predictions generated by two alternative three-parameter logistic force of mortality models. Based on results from the simulations conducted in the present study, the two models were found to generate overall very similar probability of death predictions for the range of ages

considered, including both the age range to which the models were fit (80-99) and an older age range (100-105) where the models were extrapolated to generate true predictions. The average standard deviation of the predicted probabilities of death for the two models was nearly the same in the 80-99 age range, but in the 100-105 age range the G-model appeared overall to be more precise than the K-model, though the statistical significance of the difference was not tested. It should be noted that the results of the present study are limited to a specific set of cohort mortality data, and it may be fruitful in future studies to compare the three-parameter models in multiple simulations based on several different sets of cohort data.

The model predictions of q_x generally deviated somewhat from the values of q_x used in the probability model that generated the simulated life tables. In the age range of 80-99, the average percentage deviation was less than 2% for each model, and the deviations most often showed up in the third decimal place of the differences, though sometimes not until the fourth decimal place or not even there. In the age range of 100-105, the average percentage deviation for the models was around 5-6%, and the deviations always showed up in the second decimal place of the differences. This may suggest a difference in model accuracy between the 80-99 age range and 100-105 age range, though the statistical significance has not been tested. In the age range of 100-105, both models always underestimated q_x. It would be fruitful in a future simulation study to fit the models to the full range of ages 80-105 and compare their fit at ages 100-105 with their extrapolated fit found in the present study. This would help to determine if extrapolation contributes to what may be a decreased level of accuracy of the models at those ages.

There is room for follow-up and further investigation regarding issues that may arise when using the Monte Carlo simulation program and fitting the three-parameter logistic models to a sample of cohort mortality data. Those issues include the potential for parameter estimates to be negative, and the potential for convergence failures when the simulation program performs NLS estimation. In the 601k and 300k simulations of the present study, however, there were no negative parameter estimates. There were also no NLS estimation failures in the 601k simulation, though there were four in the 300k simulation, and it should be considered how these may have affected the results. For example, four of the simulated life tables generated in the 601k simulation were not used in the 300k simulation, due to NLS estimation failures in those cycles, and four additional simulated life tables were generated in the 300k simulation, which were not used in the 601k simulation. Thus when comparing the two simulations, there is an additional factor of variation in the experimental conditions. Also, when

considering the 300k simulation by itself, the results for the mean probabilities of death, standard deviations, and differences, as well as the calculated statistics of the parameter estimates may be affected by some degree of bias.

Another area that might be investigated is the differences in the estimates of the G-model parameters found in the simulations for the 1893 cohort constructed in the present study from the Social Security DMF, when compared to the findings by Bongaarts (2005) for the U.S. mortality data he used, for years 1950-2000. Also, the mean estimates of the parameter γ found in the simulations are substantially higher than might be expected. The 1893 cohort would have been in their early twenties at the time of World War I, and perhaps this may have contributed to the higher value of the background force of mortality parameter γ.

It would be interesting to run the simulations of the present study using different sources of mortality data, including separate life tables for males and females, and compare the results. It also may be fruitful to construct life tables over a much broader range of ages and run simulations that fit the models to an extended age range (i.e. 30-99 years of age), and also to compare the predictions of the K-model and G-model at ages over 105. The simulation approach presented in this work may be extended to compare other force of mortality models, and over more extensive age ranges. Models may be compared over different cohort data sets and evaluated for their robustness in fitting the specific characteristics and peculiarities of different cohorts and populations, including small populations. Models may be altered to attempt to adjust for systematic overestimation or underestimation of q_x at certain ages, and new models may be explored.

6 References

References

[1] Ancestry.com. *Social Security Death Index* [database on-line]. Provo, UT, USA: Ancestry.com Operations Inc, 2011. Provided in Ancestry Library Edition, distributed by ProQuest LLC. Original data: Social Security Administration. *Social Security Death Index, Master File*. Social Security Administration.

[2] Bates, D.M., and Watts, D.G. 2007. *Nonlinear Regression Analysis and its Applications*. New Jersey: Wiley.

[3] Beard, R.E. 1959. "Note on Some Mathematical Mortality Models." Pp. 302-11 in *CIBA Foundation Colloquia on Ageing, The Life Span of Animals 5*, edited by G.E.W. Wolstenholme and M. O'Connor. Boston: Little, Brown and Company.

[4] Beard, R.E. 1964. "Some Observations on Stochastic Processes with Particular Reference to Mortality Studies." *International Congress of Actuaries* 3: 463-477.

[5] Beard, R.E. 1971. "Some Aspects of Theories of Mortality, Cause of Death Analysis, Forecasting and Stochastic Processes." Pp. 57-68 in *Biological Aspects of Demography*, edited by W. Brass. New York: Barnes & Noble.

[6] Bongaarts, J. 2005. "Long-Range Trends in Adult Mortality: Models and Projection Methods." *Demography* 42: 23-49.

[7] Casella, G., and Berger, R. 1990. *Statistical Inference*. Belmont, CA: Wadsworth.

[8] Davidian, M. Simulation handout for ST810A, Spring 2005. Retrieved from: http://www4.stat.ncsu.edu/~davidian/st810a/simulation_handout.pdf, 2011.

[9] De Matteis, A., and Pagnutti, S. 1993. "Long-range Correlation Analysis of the Wichman-Hill Random Number Generator." *Statistics and Computing* 3: 67-70.

[10] Doray, L.G. 2002. "Living to Age 100 in Canada in 2000." *International Symposium on Living to 100 and Beyond: Survival at Advanced Ages*, Society of Actuaries, SOA Monograph M-LI02-01, 21p.

[11] Doray, L.G. 2008. "Inference for Logistic-type Models for the Force of Mortality." *International Symposium on Living to 100 and Beyond,* Society of Actuaries*,* SOA Monograph M–LI08–1*,* 18p.

[12] Dunn, W.L., and Shultis, J.K. 2012. *Exploring Monte Carlo Methods*. San Diego: Elsevier.

[13] Eayres, D. and Williams, E.S. 2004. "Evaluation of Methodologies for Small Area Life Expectancy Estimation." *Journal of Epidemiology & Community Health* 58: 243-249.

[14] Gentle, J.E. 2003. *Random Number Generation and Monte Carlo Methods, 2nd Edition*. New York: Springer.

[15] Horiuchi, S. and Coale, A.J. 1990. "Age Patterns of Mortality for Older Women: An Analysis Using the Age-specific Rate of Mortality Change With Age." *Mathematical Population Studies* 2: 245-67.

[16] Horiuchi, S., and Wilmoth, J.R. 1998. "Deceleration in the Age Pattern of Mortality at Older Ages." *Demography* 35: 391-412.

[17] Howard, R.C.W. 2011. "Mortality Rates at Oldest Ages." *International Symposium on Living to 100 and Beyond,* Society of Actuaries*,* in 2011 Living to 100 Monograph*,* 35p.

[18] Kachitvichyanukul, V., and Schmeiser, B.W. 1988. "Binomial Random Variate Generation." *Communications of the ACM* 31: 216-223.

[19] Lawless, J.F. 2003. *Statistical Models and Methods for Lifetime Data, 2nd Edition*. New Jersey: Wiley.

[20] MCNP, 2003. *A General Monte Carlo N-Particle Transport Code, Version 5*, Vol. I *Overview and Theory*, LA-UR-03-1987, Los Alamos Nat. Lab., Los Alamos, NM.

[21] Perks, W. 1932. "On Some Experiments in the Graduation of Mortality Statistics." *Journal of the Institute of Actuaries* 63: 12-40.

[22] R Development Core Team (2009). R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. ISBN 3-900051-07-0, URL http://www.R-project.org.

[23] Rizzo, M.L. 2008. *Statistical Computing with R*. Boca Raton: Chapman & Hall/ CRC.

[24] Scherbov, S., and Ediev, D. 2011. "Significance of Life Table Estimates for Small Populations: Simulation-based Study of Standard Errors." *Demographic Research* 24: 527-550.

[25] Silcocks, P.B.S., Jenner, D.A., and Reza, R. 2001. "Life Expectancy as a Summary of Mortality in a Population: Statistical Considerations and Suitability for Use by Health Authorities." *Journal of Epidemiology & Community Health* 55: 38-43.

[26] Slud, Eric V. 2006. "More Probability Theory in Life Tables." Chapter 3 in *Actuarial Mathematics and Life-Table Statistics*. Retrieved from: http://www2.math.umd.edu/~slud//s470/BookChaps/, Feb 2012.

[27] Tabeau, E. 2001. "A Review of Demographic Forecasting Models for Mortality." Pp1-32 in *Forecasting Mortality in Developed Countries: Insights from a Statistical, Demographic and Epidemiological Perspective*, edited by Ewa Tabeau, Anneke van den Berg Jeths, and Christopher Heathcote. Boston: Kluwer.

[28] Thatcher, A.R., Kannisto, V., and Vaupel, J.W. 1998. *The Force of Mortality at Ages 80 to 120*, *Monographs on Population Aging, 5*, Odense, Denmark: Odense University Press.

[29] Thatcher, A.R. 1999. "The Long-Term Pattern of Adult Mortality and the Highest Attained Age." *Journal of the Royal Statistical Society* 162 Part 1: 5-43.

[30] Toson, B., Baker, A., and the Office of National Statistics. 2003. "Life Expectancy at Birth: Methodological Options for Small Populations." *National Statistics Methodological Series* 33: 1-27.

[31] Vincent, P. 1951. "La mortalité des vieillards." *Population* 6: 181-204.

[32] Williams, E., Dinsdale, H., Eayres, D., and Tahzib, F. 2005. *Technical Report: Calculating Life Expectancy in Small Areas.* South East Public Health Observatory.

Addendum

The function *LTPsim* was written after the author viewed material on a website, including one or two methods for simulating life table data, one of which was probably similar to or the same as the method used in *LTPsim*. The author subsequently was unable to find the website again for reference. The basic method used for simulating life tables in *LTPsim* is similar to the one presented in Chapter 3 of *Actuarial Mathematics and Life-Table Statistics* (Slud, 2006), though Slud uses conditional probability of survival as the probability parameter in the binomial distributions, as opposed to conditional probability of death, and therefore generates the number of survivors at each age more directly.

7 Appendices

7.1 Determining Cohort Distribution of Deaths by Age

The primary method used to obtain mortality data from the SSDI database aimed to reduce the number of queries necessary while still providing for exact determination of age upon death for each individual included in the cohort. The SSDI database was systematically queried for each year of death between 1973 and 2011, for individuals born in the year 1893.

For each year of death between 1973 and 2003, results from 55 separate queries were used to determine how many individuals died before attaining their birthday and how many died after attaining it. Rather than performing a separate query for every combination of birth month/death month in a given year, which would require $12 \times 12 = 144$ queries per year, only 55 queries (one of which was a check) were performed since for a given birth month, once the number of individuals who died before attaining their birth month (or after) is determined, the number who died after their birth month (or before) can be calculated by subtracting that number, and the number dying in their birth month, from the total number of individuals born in the given month and dying in the given year. One of the queries obtained the total number of deaths in the entire year, which could be compared with the sum of deaths in each month of the year. As already explained in this paper, individuals who died in the same month of year as the month in which they were born were excluded from the cohort. For the years of death between 2004 and 2011, corresponding to age upon death of at least 110 (the final interval of the life table), only 13 queries were performed, one for the total number of deaths in the year, and one each for deaths that occurred in birth months, in order to exclude those individuals from the cohort.

As a representative example, Table 7.1 shows the data obtained from the SSDI for year of death 1973. As can be seen, for the birth month of January, only two queries were required. One query obtained the total number of individuals born in January 1893 who died in the entire year of 1973, and the other query obtained the number of individuals born in January 1893 who died in January 1973. Those individuals were excluded from the cohort because of the difficulty of determining which ones attained their birthday and which did not. All the individuals born in January 1893 and dying between February 1973 and December 1973 would have died after attaining their 80th birthday, and that number is easily calculated.

Table 7.1 1893 Cohorts Deaths in 1973 as Obtained from SSDI Queries

Deaths 51942 Birth Month	1973 Death Month All	Jan	Feb	Mar	Apr	May	Jun	July	Aug	Sep	Oct	Nov	Dec
Jan	4559	454											
Feb	4205	417	376										
Mar	4650	466	431	392									
Apr	4239	421	354	301	385								
May	4116	393	362	320	340	340							
Jun	3920	363	408	328	317	325	285						
July	4386							339	346	340	357	346	389
Aug	4411								359	365	367	336	388
Sep	4625									372	380	362	411
Oct	4376										374	370	392
Nov	4138											355	371
Dec	4317												401
Total	51942												

Source: Ancestry.com *Social Security Death Index* (accessed in Ancestry Library Edition, October 2011)

The birth months of June and July, being in the middle of the year, require the most number of queries (7 each). All the individuals born in June 1893 and dying between July 1973 and December 1973 would have died after attaining their 80th birthday. We find that number by subtracting the sum of deaths of individuals born in June 1893 and dying between January 1973 and June 1973 from the total number of individuals born in June 1893 and dying in the year 1973. For birth month July, we sum the number of deaths of individuals born in July 1893 and dying between July 1973 and December 1973 and subtract this from the total number of deaths of individuals born in July 1893 and dying throughout 1973 to obtain the number of individuals born in July 1893 who died before attaining their birth month. Those individuals would have died at age 79. The number of individuals born in July 1893 and dying after their birth month can then easily be calculated. Those individuals would have died at age 80. Though this method decreases the number of queries necessary, it may be noted that if any individuals in the DMF would happen to have a birth month recorded but no death month recorded (only year of death), then this could result in inaccuracies in determining the number of individuals who died before as opposed to after their birth month, and thus the number of deaths at each age.

Once all birth months in a given year have been queried, we can sum to find the number of individuals born in 1893 who died before attaining their birth month in the given year, and sum to find the number of individuals who died after their birth month in that year. This process is conducted for all years of death in order to obtain the cohort distribution of deaths at age 80 and above.

7.2 Estimation of Total Number of Random Variates Used in Simulation

The number of random variates generated by the RNG in each simulation was estimated based on information given by Kachitvichyanukul and Schmeiser (1988). For a given binomial(n, p) distribution from which a binomial random variate is to be generated using their BTPE algorithm, we can determine the expected number of uniform(0,1) random numbers that must be supplied (Kachitvichyanukul and Schmeiser, 1988). For every iteration of the BTPE algorithm, two uniform(0,1) random numbers are used (Kachitvichyanukul and Schmeiser, 1988). Knowing this, and the expected number of iterations of the BTPE algorithm when generating a binomial random variate from a binomial(n, p) distribution, one can calculate the expected number of uniform(0,1) random numbers that are needed to generate the binomial random variate (Kachitvichyanukul and Schmeiser, 1988).

Functions were coded in R that provide for post-processing of the data from a simulation in order to calculate the expected number of uniform(0,1) random variates needed to generate all binomial random variates in the simulation. The function *binvar* calculates the expected number of uniform(0,1) random variates needed to generate one binomial(n, p) random variate, as directly follows from explicit information given by Kachitvichyanukul and Schmeiser (1988). The required inputs are n, the number of survivors at age x, and p, the conditional probability of death q_x. These parameters vary for each age of the simulated life table.

The function *binvar* is called within another function, *totalran*, which uses two *for* loops to calculate the total expected number of binomial(n, p) random variates needed for the simulation. The outer *for* loop ranges through all the histories of the Monte Carlo simulation, while the inner *for* loop ranges through the simulated life table in a given history of the simulation. For the simulations carried out in the present study, there were 1500 histories, and each history generated a simulated life table that required 26 binomial random variates (deaths at ages 80-105). The function *totalran* returns the total expected number of uniform(0,1) random variates needed in

the simulation, as well as the average number of uniform(0,1) random variates needed to generate each binomial random variate in the simulation.

Using *totalran*, the expected number of uniform(0,1) random variates required from the RNG for the 601k simulation was found to be 89,480.03, with about 2.29 required on average for each binomial random variate generated. For the 300k simulation, the total expected number of uniform(0,1) random variates required was found to be 89,460.03, with about 2.29 required on average. R coding for *totalran* only is provided below.

```
totalran <- function(data){
    lx <- data$simdata$lsim
    qx <- data$stats$mean[1,]
    num <<- matrix(NA,length(lx[,1]),length(qx))
    total <- 0
    for(i in 1:length(lx[,1])){
        for(j in 1:length(qx)){
            num[i,j] <<- binvar(lx[i,j],qx[j])
            total <- total + num[i,j]
            }
        }
    avg <- total/(length(num[,1])*length(num[1,]))
    return(total=total,avg=avg)
    }
```

7.3 R Coding for Monte Carlo Simulation Program

```
Note 1: Lines that were too long to fit onto the page were split into multiple lines
in the coding shown below, and might need adjusted when running the code in R.
Note 2: Any comments present in original coding have been removed from the coding below.

perksMCsim <- function(reps,table.ages,table.pop,est.ages,pred.ages,size="default",
                rng="Wich",seed,state="fresh",olddata="empty"){
    RNGkind("Wich")
    if(state == "fresh"){
        set.seed(seed)
        oldstate <<- .Random.seed
        }
    else .Random.seed <<- state
    n <- x <- l <- i <- m <- NA
    d <- q <- p <- k <- begin <- NA
    end <- ll <- dl <- ql <- pl <- xl <- NA
    nlsGstart <- nlsKstart <- NA
    n <- reps
    x <- table.ages
    l <- table.pop
    j <- length(l)
    d <- l[1:(j-1)]-l[2:j]
    q <- d/l[1:(j-1)]
    p <- 1-q
    k <- length(est.ages)
    begin.est <- est.ages[1]-x[1]+1
    end.est <- est.ages[k]-x[1]+2
    ll <- l[begin.est:end.est]
    if(size=="default"){
        l1 <- ll[1]
        }
    else l1 <- size
    dl <- ll[1:(end.est-1)]-ll[2:end.est]
    ql <- q[begin.est:(end.est-1)]
    pl <- 1-ql
    xl <- est.ages
    m <- length(pred.ages)
    begin.pred <- pred.ages[1]-x[1]+1
    end.pred <- begin.pred + (m-1)
    qp <- q[begin.pred:end.pred]
    raw <- matrix(NA,n,(3*m+9))
    rawtemp <<- NA
    s <- 1
    t <- 0
    nlsGfail <<- 0
    nlsKfail <<- 0
    nlsGfails <<- c(NA)
    nlsKfails <<- c(NA)
    rawfails <<- vector("list")
    min <- c(1e-6,0.04,-0.2)
    max <- c(1e-3,0.14,0.1)
    step <- c(2,0.01,0.05)
    nlsGstart <- try(startfinderG(xl,pl,min,max,step))
    Gstart <<- nlsGstart
```

```
min <- c(0.5,1e-6,0.04)
max <- c(3,1e-3,0.14)
step <- c(0.1,2,0.01)
nlsKstart <- try(startfinderK(xl,pl,min,max,step))
Kstart <<- nlsKstart
while(s <= n){
    t <- t + 1
    sl <- sd <- sq <- sp <- qG <-NA
    NLSEG <- NLSEK <- qK <- NA
    simLT <- LTPsim(l1,qp)
    oldoldstate <<- oldstate
    oldstate <<- .Random.seed
    sl <- simLT$survivors
    sd <- simLT$deaths
    sq <- sd/sl[1:m]
    sp <- 1-sq
    raw[s,1:(m+1)] <- sl
    raw[s,(m+2):(2*m+1)] <- sd
    raw[s,(2*m+2):(3*m+1)] <- sq
    rawtemp <<- raw
    step <- c(0.1,0.0001,0.005)
    nlseK <- try(nlsfinderK(xl,sp[1:k],nlsKstart$par,step))
    if(length(nlseK) != 1){
        raw[s,(3*m+2):(3*m+5)] <- c(nlseK$par,nlseK$val)
        rawtemp <<- raw
        step <- c(0.0001,0.005,0.05)
        nlseG <- try(nlsfinderG(xl,sp[1:k],nlsGstart$par,step))
        if(length(nlseG) != 1){
            raw[s,(3*m+6):(3*m+9)] <- c(nlseG$par,nlseG$val)
            s <- s + 1
            rawtemp <<- raw
            }
        else {
            nlsGfail <<- nlsGfail + 1
            nlsGfails[nlsGfail] <<- t
            rawfails[[(nlsGfail+nlsKfail)]] <<- raw[s,]
            }
        }
    else {
        nlsKfail <<- nlsKfail + 1
        nlsKfails[nlsKfail] <<- t
        rawfails[[(nlsGfail+nlsKfail)]] <<- raw[s,]
        }
    }
if(olddata != "empty"){
    raw <- rbind(olddata,raw)
    rawtemp <<- raw
    n <- length(raw[,1])
    }
qK <- matrix(NA,n,m)
qG <- matrix(NA,n,m)
for(i in 1:n){
    qK[i,] <- qest(pred.ages,raw[i,(3*m+2):(3*m+5)],p="K")
    qG[i,] <- qest(pred.ages,raw[i,(3*m+6):(3*m+9)],p="G")
    }
```

```
    simdata <<- MCdata(raw,qK,qG,est.ages,pred.ages)
    nlsKstats <<- MCstats(simdata$qsim,simdata$qK,qp)
    nlsGstats <<- MCstats(simdata$qsim,simdata$qG,qp)
    results <<- MCcompare(qp,pred.ages,nlsKstats,nlsGstats)
    list(data=l,simdata=simdata,stats=results,nlsKfailures=nlsKfail,which.nlsKfailed=
        nlsKfails,nlsGfailures=nlsGfail,which.nlsGfailed=nlsGfails)
    }

Note: Above coding is given as was used to run simulations in present study, though
several items will be noted for correction:
1. rng="Wich" is not needed with inputs since "Wich" is declared in program as RNGkind
2. begin,end were initialized to NA instead of begin.est,end.est,begin.pred,end.pred
3. qp, nlsGstart, nlsKstart, Gstart, and Kstart were not initialized to NA
4. NLSEG and NLSEK were initialized to NA rather than nlseG and nlseK

LTPsim <- function(l1,q) {
    k <-length(q)+1
    ls <- rep(0,times=k)
    ds <- rep(0,times=(k-1))
    ls[1] <- l1
    for (i in 1:(k-1)) {
        ds[i] <- rbinom(1,ls[i],q[i])
        ls[i+1] <- ls[i]-ds[i]
        }
    list(survivors=ls,deaths=ds)
    }

startfinderG <- function(x,p,min,max,step){
    mo <- min[2]
    A <- max[1]
    M <- max[2]
    G <- max[3]
    astep <- step[1]
    mstep <- step[2]
    gstep <- step[3]
    par <- c(NA,NA,NA)
    val <- 1e7
    nlse <<- vector("list")
    i <- 1
    while(mo <= M){
        ao <- min[1]
        while(ao <= A){
            go <- min[3]
            while(go <= G){
                snls <- try(nls(p ~ I(exp((1/m)*(log(1+a*exp(m*x))-
                            log(1+a*exp(m*(x+1))))-g)),start=list(a=ao,m=mo,g=go)),
                                silent=TRUE)
                if(length(snls) != 1){
                    ssr <- sum(resid(snls)^2)
                    nlse[[i]] <<- c((summary(snls))$par[,1],ssr)
                    par <- nlse[[i]][1:3]
                    go <- G
                    ao <- A
                    mo <- M
                    }
```

```
                else nlse[[i]] <<- NA
                go <- go + gstep
                i <- i+1
                }
            ao <- ao * astep
            }
        mo <- mo + mstep
        }
    list(par=par,val=ssr)
    }

startfinderK <- function(x,p,min,max,step){
    mo <- min[3]
    K <- max[1]
    A <- max[2]
    M <- max[3]
    kstep <- step[1]
    astep <- step[2]
    mstep <- step[3]
    par <- c(NA,NA,NA)
    val <- 1e7
    nlse <<- vector("list")
    i <- 1
    while(mo <= M){
        ko <- min[1]
        while(ko <= K){
            ao <- min[2]
            while(ao <= A){
                snls <- try(nls(p ~ I(exp((k/m)*(log(1+a*exp(m*x))-
                            log(1+a*exp(m*(x+1))))))),start=list(k=ko,a=ao,m=mo)),
                                silent=TRUE)
                if(length(snls) != 1){
                    ssr <- sum(resid(snls)^2)
                    nlse[[i]] <<- c((summary(snls))$par[,1],ssr)
                    par <- nlse[[i]][1:3]
                    ao <- A
                    ko <- K
                    mo <- M
                    }
                else nlse[[i]] <<- NA
                ao <- ao * astep
                }
            ko <- ko + kstep
            }
        mo <- mo + mstep
        }
    list(par=par,val=ssr)
    }

nlsfinderG <- function(x,p,theta,step){
    mo <- theta[2]-0.02
    A <- theta[1]+0.0005
    M <- theta[2]+0.02
    G <- theta[3]+0.02
    astep <- step[1]
```

```
    mstep <- step[2]
    gstep <- step[3]
    par <- c(NA,NA,NA)
    val <- 1e7
    nlse <<- vector("list")
    i <- 1
    while(mo <= M){
        ao <- theta[1]-0.0005
        while(ao <= A){
            go <- theta[3]-0.02
            while(go <= G){
                snls <- try(nls(p ~ I(exp((1/m)*(log(1+a*exp(m*x))-
                            log(1+a*exp(m*(x+1))))-g)),start=list(a=ao,m=mo,g=go)),
                                silent=TRUE)
                if(length(snls) != 1){
                    ssr <- sum(resid(snls)^2)
                    nlse[[i]] <<- c((summary(snls))$par[,1],ssr)
                    par <- nlse[[i]][1:3]
                    go <- G
                    ao <- A
                    mo <- M
                    }
                else nlse[[i]] <<- NA
                go <- go + gstep
                i <- i+1
                }
            ao <- ao + astep
            }
        mo <- mo + mstep
        }
    list(par=par,val=ssr)
    }

nlsfinderK <- function(x,p,theta,step){
    ko <- theta[1]-0.5
    K <- theta[1]+0.5
    A <- theta[2]+0.0005
    M <- theta[3]+0.02
    kstep <- step[1]
    astep <- step[2]
    mstep <- step[3]
    par <- c(NA,NA,NA)
    val <- 1e7
    nlse <<- vector("list")
    i <- 1
    while(ko <= K){
        mo <- theta[3]-0.02
        while(mo <= M){
            ao <- theta[2]-0.0005
            while(ao <= A){
                snls <- try(nls(p ~ I(exp((k/m)*(log(1+a*exp(m*x))-
                            log(1+a*exp(m*(x+1))))))),start=list(k=ko,a=ao,m=mo)),
                                silent=TRUE)
                if(length(snls) != 1){
                    ssr <- sum(resid(snls)^2)
```

```
                    nlse[[i]] <<- c((summary(snls))$par[,1],ssr)
                    par <- nlse[[i]][1:3]
                    ao <- A
                    ko <- K
                    mo <- M
                    }
                else nlse[[i]] <<- NA
                ao <- ao + astep
                }
            mo <- mo + mstep
            }
        ko <- ko + kstep
        }
    list(par=par,val=ssr)
    }

qest <- function(x,theta,p){
    if(p == "G"){
        k <- 1
        a <- theta[1]
        m <- theta[2]
        g <- theta[3]
        }
    else{
        k <- theta[1]
        a <- theta[2]
        m <- theta[3]
        g <- 0
        }
    q <- rep(NA,length(x))
    q <- 1-exp((k/m)*(log(1+a*exp(m*x))-log(1+a*exp(m*(x+1))))-g)
    return(q)
    }

MCdata <- function(raw,qK,qG,est.ages,pred.ages){
    k <- length(est.ages)
    m <- length(pred.ages)
    lsim <- raw[,1:(m+1)]
    colnames(lsim) <- c(pred.ages,"+")
    dsim <- raw[,(m+2):(2*m+1)]
    colnames(dsim) <- pred.ages
    qsim <- raw[,(2*m+2):(3*m+1)]
    colnames(qsim) <- pred.ages
    nlseK <- raw[,c((3*m+3):(3*m+4),(3*m+2),(3*m+5))]
    colnames(nlseK) <- c("alpha","beta","kappa","vqlue")
    nlseG <- raw[,(3*m+6):(3*m+9)]
    colnames(nlseG) <- c("alpha","beta","gamma","value")
    colnames(qK) <- pred.ages
    colnames(qG) <- pred.ages
    list(lsim=lsim, dsim=dsim, qsim=qsim, "NLSE-K"=nlseK, "NLSE-G"=nlseG,
        qK=qK, qG=qG)
    }

MCstats <- function(sim.data,model.data,true.data){
    sim <- sim.data
```

```
    mod <- model.data
    q <- true.data
    n <- length(sim[,1])
    m <- length(sim[1,])
    MCmean.sim <- rep(NA,times=m)
    MCmean.mod <- rep(NA,times=m)
    MCstdv.sim <- rep(NA,times=m)
    MCstdv.mod <- rep(NA,times=m)
    MCdif.sim <- rep(NA,times=m)
    MCdif.mod <- rep(NA,times=m)
    i <- 1
    while(i <= m){
        MCmean.sim[i] <- sum(sim[,i])/n
        MCmean.mod[i] <- sum(mod[,i])/n
        MCstdv.sim[i] <- sqrt((1/n)*(1/(n-1))*sum((sim[,i]-MCmean.sim[i])^2))
        MCstdv.mod[i] <- sqrt((1/n)*(1/(n-1))*sum((mod[,i]-MCmean.mod[i])^2))
        MCdif.sim[i] <- MCmean.sim[i]-q[i]
        MCdif.mod[i] <- MCmean.mod[i]-q[i]
        i <- i + 1
        }
    list(mean.sim=MCmean.sim,mean.model=MCmean.mod,stdv.sim=MCstdv.sim,
        stdv.mod=MCstdv.mod,dif.sim=MCdif.sim,dif.mod=MCdif.mod)
    }

MCcompare <- function(q,pred.ages,stats1,stats2){
    mean <- rbind(q,stats1$mean.sim,stats1$mean.mod,stats2$mean.mod)
    rownames(mean) <- c("True","Simulated","K-Model","G-Model")
    colnames(mean) <- pred.ages
    stdv <- rbind(stats1$stdv.sim,stats1$stdv.mod,stats2$stdv.mod)
    rownames(stdv) <- c("Simulated(ML)","K-Model","G-Model")
    colnames(stdv) <- pred.ages
    difference <- rbind(stats1$dif.sim,stats1$dif.mod,stats2$dif.mod)
    rownames(difference) <- c("Simulated(ML)","K-Model","G-Model")
    colnames(difference) <- pred.ages
    n <- length(q)
    summary <- matrix(NA,3,2)
    summary[1,1] <- sum(abs(stats1$dif.sim))/n
    summary[2,1] <- sum(abs(stats1$dif.mod))/n
    summary[3,1] <- sum(abs(stats2$dif.mod))/n
    summary[1,2] <- sum(stats1$stdv.sim)/n
    summary[2,2] <- sum(stats1$stdv.mod)/n
    summary[3,2] <- sum(stats2$stdv.mod)/n
    colnames(summary) <- c("avg␣difference","avg␣stdv")
    rownames(summary) <- c("Simulated(ML)","K-Model","G-Model")
    list(mean=mean,stdv=stdv,difference=difference,summary=summary)
    }
```

7.4 Simulation Data (Selected)

```
601k Simulation Data (histories 1-5 only, simulation statistics, and other information)

> l93  #Supplied Life Table Data, 1893 Cohort Survivors (Ages 80-110+)
[1] 601059 554120 508131 463473 420157 377078 336738 296427 258133 223260 190218 159565 131279 106070  84339  65383
50189  37673  28054  20155  14159 [22]   9504   6250   3956   2369   1418    803    442    252    144     86

> system.time(sim <- perksMCsim(reps=1500,table.ages=80:110,table.pop=l93,est.ages=80:99,pred.ages=80:105,seed=3))
    user  system elapsed
 643.925   5.488 646.423    #Time in seconds to complete simulation
#Note: Any warning messages (if they were present) are not shown

> sim$simdata$lsim[1:5,]
         80     81     82     83     84     85     86     87     88
[1,] 601059 554151 508348 463937 420380 377404 336945 296679 258067
[2,] 601059 554218 508364 463892 420826 377783 337181 296532 258285
[3,] 601059 553638 508093 463389 419782 376751 336313 296201 257849
[4,] 601059 554271 507938 463330 420028 376460 336387 295832 257332
[5,] 601059 554143 507952 463435 420214 377144 336677 296479 258636
         89     90     91     92     93    94    95    96    97    98
[1,] 223357 190069 159734 131313 106029 84571 65710 50479 37912 28188
[2,] 223321 190358 159840 131742 106378 84653 65533 50281 37728 28133
[3,] 223009 190117 159648 131523 106220 84282 65511 50374 37754 28016
[4,] 222605 189782 159233 131016 105767 84117 65131 49795 37465 27826
[5,] 223630 190346 159525 131059 106151 84123 65248 49921 37608 27914
         99   100  101  102  103  104  105   +
[1,] 20139 14142 9485 6132 3823 2265 1372 752
[2,] 20197 14139 9500 6286 3945 2341 1371 779
[3,] 19959 13877 9330 6131 3916 2290 1337 756
[4,] 19930 13949 9401 6207 3877 2344 1398 765
[5,] 19984 14031 9441 6210 3944 2291 1393 782

> sim$simdata$dsim[1:5,]
        80    81    82    83    84    85    86    87    88    89    90
[1,] 46908 45803 44411 43557 42976 40459 40266 38612 34710 33288 30335
[2,] 46841 45854 44472 43066 43043 40602 40649 38247 34964 32963 30518
[3,] 47421 45545 44704 43607 43031 40438 40112 38352 34840 32892 30469
[4,] 46788 46333 44608 43302 43568 40073 40555 38500 34727 32823 30549
[5,] 46916 46191 44517 43221 43070 40467 40198 37843 35006 33284 30821
        91    92    93    94    95    96   97   98   99  100  101  102
[1,] 28421 25284 21458 18861 15231 12567 9724 8049 5997 4657 3353 2309
[2,] 28098 25364 21725 19120 15252 12553 9595 7936 6058 4639 3214 2341
[3,] 28125 25303 21938 18771 15137 12620 9738 8057 6082 4547 3199 2215
[4,] 28217 25249 21650 18986 15336 12330 9639 7896 5981 4548 3194 2330
[5,] 28466 24908 22028 18875 15327 12313 9694 7930 5953 4590 3231 2266
      103 104 105
[1,] 1558 893 620
[2,] 1604 970 592
[3,] 1626 953 581
[4,] 1533 946 633
[5,] 1653 898 611

> sim$simdata$qsim[1:5,]
            80         81         82         83        84        85
[1,] 0.07804226 0.08265437 0.08736338 0.09388559 0.1022313 0.1072034
[2,] 0.07793079 0.08273640 0.08748062 0.09283626 0.1022822 0.1074744
[3,] 0.07889575 0.08226495 0.08798389 0.09410452 0.1025080 0.1073335
[4,] 0.07784261 0.08359268 0.08782174 0.09345823 0.1037264 0.1064469
[5,] 0.07805557 0.08335574 0.08764017 0.09326227 0.1024954 0.1072985
            86        87        88        89        90        91
[1,] 0.1195032 0.1301474 0.1345000 0.1490350 0.1595999 0.1779271
[2,] 0.1205554 0.1289810 0.1353698 0.1476037 0.1603190 0.1757883
[3,] 0.1192698 0.1294796 0.1351178 0.1474918 0.1602645 0.1761688
[4,] 0.1205605 0.1301414 0.1349502 0.1474495 0.1609689 0.1772057
[5,] 0.1193963 0.1276414 0.1353485 0.1488351 0.1619209 0.1784423
            92        93        94        95        96        97
[1,] 0.1925476 0.2023786 0.2230197 0.2317912 0.2489550 0.2564887
[2,] 0.1925278 0.2042246 0.2258632 0.2327377 0.2496569 0.2543204
[3,] 0.1923846 0.2065336 0.2227166 0.2310604 0.2505261 0.2579329
[4,] 0.1927169 0.2046952 0.2257094 0.2354639 0.2476152 0.2572801
[5,] 0.1900518 0.2075157 0.2243738 0.2349038 0.2466497 0.2577643
```

```
            98        99       100       101       102       103
[1,] 0.2855470 0.2977804 0.3293028 0.3535055 0.3765492 0.4075334
[2,] 0.2820887 0.2999455 0.3280996 0.3383158 0.3724149 0.4065906
[3,] 0.2875857 0.3047247 0.3276645 0.3428725 0.3612787 0.4152196
[4,] 0.2837634 0.3001004 0.3260449 0.3397511 0.3753826 0.3954088
[5,] 0.2840868 0.2978883 0.3271328 0.3422307 0.3648953 0.4191176
           104       105
[1,] 0.3942605 0.4518950
[2,] 0.4143528 0.4318016
[3,] 0.4161572 0.4345550
[4,] 0.4035836 0.4527897
[5,] 0.3919686 0.4386217

> sim$simdata$"NLSE-K"[1:5,]
            alpha       beta    kappa         vqlue
[1,] 3.198626e-05 0.09031280 1.745187 0.0002293891
[2,] 3.190883e-05 0.09156278 1.583020 0.0002885656
[3,] 2.271590e-05 0.08584234 3.476106 0.0002302841
[4,] 3.177180e-05 0.09121374 1.639365 0.0002643820
[5,] 3.134874e-05 0.09311467 1.425920 0.0002298626

> sim$simdata$"NLSE-G"[1:5,]
            alpha      beta      gamma         value
[1,] 8.189119e-06 0.1107811 0.02086594 0.0002029090
[2,] 8.987983e-06 0.1099005 0.01931545 0.0002622820
[3,] 4.504687e-06 0.1167289 0.02803281 0.0002024522
[4,] 8.337937e-06 0.1106777 0.02043647 0.0002344362
[5,] 9.693885e-06 0.1092162 0.01804526 0.0002008135

> sim$simdata$qK[1:5,]
             80         81         82         83        84        85
[1,] 0.07383620 0.08020683 0.08706986 0.09445309 0.1023841 0.1108898
[2,] 0.07356338 0.07997112 0.08687563 0.09430435 0.1022843 0.1108415
[3,] 0.07450280 0.08074880 0.08747870 0.09472299 0.1025129 0.1108800
[4,] 0.07385502 0.08027234 0.08718695 0.09462655 0.1026184 0.1111890
[5,] 0.07336972 0.07983537 0.08680409 0.09430287 0.1023580 0.1109945
             86         87        88         89         90         91
[1,] 0.1199960 0.1297270 0.1401047 0.1511485 0.1628742 0.1752933
[2,] 0.1200007 0.1297847 0.1402136 0.1513045 0.1630705 0.1755199
[3,] 0.1198563 0.1294735 0.1397627 0.1507541 0.1624764 0.1749562
[4,] 0.1203635 0.1301653 0.1406151 0.1517308 0.1635263 0.1760112
[5,] 0.1202356 0.1301022 0.1406121 0.1517795 0.1636140 0.1761200
             92         93        94         95         96         97
[1,] 0.1884127 0.2022332 0.2167498 0.2319499 0.2478137 0.2643131
[2,] 0.1886560 0.2024756 0.2169689 0.2321186 0.2478996 0.2642784
[3,] 0.1882172 0.2022799 0.2171602 0.2328690 0.2494112 0.2667851
[4,] 0.1891896 0.2030597 0.2176130 0.2328337 0.2486979 0.2651738
[5,] 0.1892960 0.2031338 0.2176177 0.2327246 0.2484228 0.2646723
             98         99       100       101       102       103
[1,] 0.2814115 0.2990638 0.3172165 0.3358076 0.3547675 0.3740199
[2,] 0.2812131 0.2986532 0.3165399 0.3348070 0.3533811 0.3721827
[3,] 0.2849808 0.3039804 0.3237563 0.3442711 0.3654772 0.3873161
[4,] 0.2822206 0.2997893 0.3178224 0.3362543 0.3550124 0.3740178
[5,] 0.2814246 0.2986230 0.3162031 0.3340933 0.3522163 0.3704898
            104       105
[1,] 0.3934826 0.4130688
[2,] 0.3911280 0.4101295
[3,] 0.4097188 0.4326059
[4,] 0.3931865 0.4124306
[5,] 0.3888284 0.4071447

> sim$simdata$qG[1:5,]
             80         81         82         83        84        85
[1,] 0.07549317 0.08131682 0.08768852 0.09464443 0.1022198 0.1104479
[2,] 0.07518479 0.08107516 0.08751263 0.09453246 0.1021688 0.1104540
[3,] 0.07645660 0.08199035 0.08808733 0.09478981 0.1021397 0.1101778
[4,] 0.07558434 0.08145123 0.08786852 0.09487224 0.1024974 0.1107771
[5,] 0.07507010 0.08102846 0.08753416 0.09462169 0.1023244 0.1106736
             86         87        88         89         90         91
[1,] 0.1193595 0.1289811 0.1393344 0.1504342 0.1622880 0.1748936
[2,] 0.1194173 0.1290841 0.1394746 0.1506026 0.1624739 0.1750855
[3,] 0.1189424 0.1284680 0.1387841 0.1499134 0.1618700 0.1746578
[4,] 0.1197416 0.1294172 0.1398247 0.1509787 0.1628857 0.1755429
[5,] 0.1196975 0.1294205 0.1398615 0.1510330 0.1629397 0.1755774
```

```
            92        93        94        95        96        97
[1,] 0.1882388 0.2022998 0.2170404 0.2324116 0.2483515 0.2647859
[2,] 0.1884241 0.2024651 0.2171718 0.2324952 0.2483738 0.2647339
[3,] 0.1882688 0.2026815 0.2178598 0.2337519 0.2502905 0.2673925
[4,] 0.1889373 0.2030441 0.2178262 0.2332340 0.2492046 0.2656630
[5,] 0.1889316 0.2029770 0.2176762 0.2329797 0.2488258 0.2651413
            98        99       100       101       102       103
[1,] 0.2816287 0.2987843 0.3161485 0.3336118 0.3510619 0.3683863
[2,] 0.2814908 0.2985502 0.3158101 0.3331632 0.3504995 0.3677090
[3,] 0.2849606 0.3028850 0.3210456 0.3393153 0.3575636 0.3756602
[4,] 0.2825229 0.2996879 0.3170538 0.3345110 0.3519473 0.3692507
[5,] 0.2818420 0.2988345 0.3160182 0.3332873 0.3505337 0.3676495
           104       105
[1,] 0.3854756 0.4022262
[2,] 0.3846851 0.4013262
[3,] 0.3934786 0.4109003
[4,] 0.3863124 0.4030293
[5,] 0.3845298 0.4010754

> sim$stats $mean
                  80         81         82         83        84
True      0.07809383 0.08299466 0.08788679 0.09345960 0.1025307
Simulated 0.07808807 0.08299489 0.08789883 0.09347356 0.1025448
K-Model   0.07352346 0.07997371 0.08692308 0.09439807 0.1024244
G-Model   0.07514969 0.08111399 0.08761998 0.09470131 0.1023904
                85         86         87         88         89         90
True      0.1069805 0.1197103 0.1291853 0.1350970 0.1479979 0.1611467
Simulated 0.1069802 0.1197138 0.1291654 0.1350773 0.1479962 0.1610996
K-Model   0.1110265 0.1202270 0.1300463 0.1405015 0.1516063 0.1633698
G-Model   0.1107177 0.1197108 0.1293930 0.1397827 0.1508920 0.1627253
                91         92         93         94         95         96
True      0.1772695 0.1920261 0.2048741 0.2247596 0.2323846 0.2493774
Simulated 0.1772587 0.1919883 0.2048504 0.2247197 0.2323766 0.2494001
K-Model   0.1757961 0.1888834 0.2026233 0.2170006 0.2319923 0.2475674
G-Model   0.1752783 0.1885369 0.2024763 0.2170600 0.2322399 0.2479559
                97         98         99        100        101        102
True      0.2553288 0.2815641 0.2974944 0.3287662 0.3423822 0.3670400
Simulated 0.2554398 0.2814804 0.2975301 0.3288422 0.3426388 0.3672333
K-Model   0.2636871 0.2803041 0.2973636 0.3148030 0.3325532 0.3505395
G-Model   0.2641365 0.2807000 0.2975553 0.3146042 0.3317435 0.3488674
               103        104        105
True      0.4011628 0.4014352 0.4337094
Simulated 0.4010161 0.4017543 0.4339996
K-Model   0.3686822 0.3868987 0.4051043
G-Model   0.3658703 0.3826494 0.3991069

$stdv
                        80           81           82           83
Simulated(ML) 9.042200e-06 9.660727e-06 1.013104e-05 1.079579e-05
K-Model       1.030349e-05 8.633915e-06 6.920447e-06 5.336612e-06
G-Model       1.090722e-05 7.794569e-06 5.257486e-06 4.028598e-06
                        84           85           86           87
Simulated(ML) 1.194735e-05 1.298040e-05 1.457043e-05 1.622078e-05
K-Model       4.269470e-06 4.284669e-06 5.385466e-06 6.990773e-06
G-Model       4.677492e-06 6.241398e-06 7.828131e-06 9.098781e-06
                        88           89           90           91
Simulated(ML) 1.694901e-05 1.985240e-05 2.207561e-05 2.408871e-05
K-Model       8.663078e-06 1.015423e-05 1.130076e-05 1.199752e-05
G-Model       9.918947e-06 1.024514e-05 1.012320e-05 9.746246e-06
                        92           93           94           95
Simulated(ML) 2.800992e-05 3.338797e-05 3.684535e-05 4.399797e-05
K-Model       1.223569e-05 1.222324e-05 1.261361e-05 1.460970e-05
G-Model       9.562835e-06 1.030054e-05 1.258124e-05 1.647755e-05
                        96           97           98           99
Simulated(ML) 5.031962e-05 5.748981e-05 7.001114e-05 8.682020e-05
K-Model       1.934310e-05 2.720156e-05 3.817134e-05 5.224940e-05
G-Model       2.171915e-05 2.801611e-05 3.513000e-05 4.285319e-05
                       100          101          102          103
Simulated(ML) 1.005955e-04 1.247377e-04 1.555834e-04 1.967618e-04
K-Model       6.950270e-05 9.002577e-05 1.139025e-04 1.411820e-04
G-Model       5.098891e-05 5.934298e-05 6.772328e-05 7.594332e-05
```

```
                        104          105
Simulated(ML) 2.575644e-04 3.363375e-04
K-Model       1.718629e-04 2.058849e-04
G-Model       8.382757e-05 9.121727e-05

$difference
                          80            81            82           83
Simulated(ML) -5.765380e-06  2.347862e-07  1.204395e-05 1.396684e-05
K-Model       -4.570367e-03 -3.020947e-03 -9.637051e-04 9.384687e-04
G-Model       -2.944145e-03 -1.880667e-03 -2.668021e-04 1.241710e-03
                          84            85           86           87
Simulated(ML)  1.403943e-05 -2.853535e-07 3.473843e-06 -0.0000198581
K-Model       -1.063415e-04  4.045952e-03 5.167220e-04  0.0008609925
G-Model       -1.403075e-04  3.737216e-03 4.755542e-07  0.0002076925
                          88            89           90           91
Simulated(ML) -1.977266e-05 -1.654243e-06 -4.712268e-05 -1.074205e-05
K-Model        5.404476e-03  3.608453e-03  2.223142e-03 -1.473341e-03
G-Model        4.685663e-03  2.894115e-03  1.578571e-03 -1.991173e-03
                          92            93           94           95
Simulated(ML) -3.780004e-05 -0.0000237638 -3.989624e-05 -8.007708e-06
K-Model       -3.142776e-03 -0.0022508236 -7.759009e-03 -3.923142e-04
G-Model       -3.489226e-03 -0.0023978640 -7.699606e-03 -1.446965e-04
                          96           97           98           99
Simulated(ML)  2.272622e-05 0.0001110130 -8.376266e-05  3.565934e-05
K-Model       -1.809941e-03 0.0083583256 -1.260001e-03 -1.308549e-04
G-Model       -1.421499e-03 0.0088077747 -8.641416e-04  6.084968e-05
                         100          101          102          103
Simulated(ML)  7.604973e-05  0.0002566015  0.0001932721 -0.0001466694
K-Model       -1.396318e-02 -0.0098289100 -0.0165005162 -0.0324805864
G-Model       -1.416198e-02 -0.0106386796 -0.0181725833 -0.0352924455
                         104          105
Simulated(ML)  0.0003191141  0.0002901565
K-Model       -0.0145365522 -0.0286051613
G-Model       -0.0187858384 -0.0346025854

$summary
              avg difference     avg stdv
Simulated(ML)   6.897891e-05 6.756834e-05
K-Model         6.490456e-03 4.135572e-05
G-Model         6.850320e-03 2.698274e-05

> sim$nlsKfailures
[1] 0
> sim$nlsGfailures
[1] 0
> sim$which.nlsKfailed
[1] NA
> sim$which.nlsGfailed
[1] NA

#Calculation of Parameter Estimate Statistics (601k Simulation)

> min(sim$simdata$"NLSE-K"[,1])
[1] 2.239933e-05
> max(sim$simdata$"NLSE-K"[,1])
[1] 3.56165e-05
> min(sim$simdata$"NLSE-K"[,2])
[1] 0.08573302
> max(sim$simdata$"NLSE-K"[,2])
[1] 0.1005873
> min(sim$simdata$"NLSE-K"[,3])
[1] 0.9121363
> max(sim$simdata$"NLSE-K"[,3])
[1] 3.556584
> (sum(sim$simdata$"NLSE-K"[,1]))/1500
[1] 3.193059e-05
> (sum(sim$simdata$"NLSE-K"[,2]))/1500
[1] 0.0927975
> (sum(sim$simdata$"NLSE-K"[,3]))/1500
[1] 1.469800
> min(sim$simdata$"NLSE-G"[,1])
[1] 4.504687e-06
> max(sim$simdata$"NLSE-G"[,1])
[1] 2.589645e-05
```

```
> min(sim$simdata$"NLSE-G"[,2])
[1] 0.09944492
> max(sim$simdata$"NLSE-G"[,2])
[1] 0.1167289
> min(sim$simdata$"NLSE-G"[,3])
[1] 0.005006144
> max(sim$simdata$"NLSE-G"[,3])
[1] 0.02803281
> (sum(sim$simdata$"NLSE-G"[,1]))/1500
[1] 1.084922e-05
> (sum(sim$simdata$"NLSE-G"[,2]))/1500
[1] 0.1083512
> (sum(sim$simdata$"NLSE-G"[,3]))/1500
[1] 0.01743935
> sqrt((1500/1499)*(((sum((sim$simdata$"NLSE-K"[,1])^2))/1500)-(((sum(sim$simdata$"NLSE-K"[,1]))/1500)^2)))
[1] 1.46533e-06
> sqrt((1500/1499)*(((sum((sim$simdata$"NLSE-K"[,2])^2))/1500)-(((sum(sim$simdata$"NLSE-K"[,2]))/1500)^2)))
[1] 0.002397061
> sqrt((1500/1499)*(((sum((sim$simdata$"NLSE-K"[,3])^2))/1500)-(((sum(sim$simdata$"NLSE-K"[,3]))/1500)^2)))
[1] 0.3102828
> sqrt((1500/1499)*(((sum((sim$simdata$"NLSE-G"[,1])^2))/1500)-(((sum(sim$simdata$"NLSE-G"[,1]))/1500)^2)))
[1] 2.794988e-06
> sqrt((1500/1499)*(((sum((sim$simdata$"NLSE-G"[,2])^2))/1500)-(((sum(sim$simdata$"NLSE-G"[,2]))/1500)^2)))
[1] 0.002486274
> sqrt((1500/1499)*(((sum((sim$simdata$"NLSE-G"[,3])^2))/1500)-(((sum(sim$simdata$"NLSE-G"[,3]))/1500)^2)))
[1] 0.003430238

300k Simulation (simulation statistics and NLS estimation failures)

> system.time(sim300 <- perksMCsim(reps=1500,table.ages=80:110,table.pop=193,est.ages=80:99,pred.ages=80:105,
    size=300000,seed=3))
   user  system elapsed
704.545   6.172 706.977 #Time in seconds to complete simulation
#(Note: Warning messages not shown)

> sim300$stats
$mean
                  80         81         82         83        84
True      0.07809383 0.08299466 0.08788679 0.09345960 0.1025307
Simulated 0.07810270 0.08295835 0.08788679 0.09346357 0.1025084
K-Model   0.07350215 0.07995712 0.08691170 0.09439227 0.1024244
G-Model   0.07511437 0.08109050 0.08760760 0.09469913 0.1023973
                85        86        87        88        89        90
True      0.1069805 0.1197103 0.1291853 0.1350970 0.1479979 0.1611467
Simulated 0.1069792 0.1197170 0.1291623 0.1351299 0.1480114 0.1611818
K-Model   0.1110324 0.1202385 0.1300630 0.1405225 0.1516305 0.1633955
G-Model   0.1107325 0.1197318 0.1294186 0.1398110 0.1509210 0.1627528
                91        92        93        94        95        96
True      0.1772695 0.1920261 0.2048741 0.2247596 0.2323846 0.2493774
Simulated 0.1772554 0.1920202 0.2049374 0.2247118 0.2323640 0.2493816
K-Model   0.1758211 0.1889052 0.2026390 0.2170067 0.2319851 0.2475430
G-Model   0.1753020 0.1885544 0.2024851 0.2170578 0.2322243 0.2479247
                97        98        99       100       101       102
True      0.2553288 0.2815641 0.2974944 0.3287662 0.3423822 0.3670400
Simulated 0.2552872 0.2814607 0.2974318 0.3288318 0.3424881 0.3671674
K-Model   0.2636414 0.2802332 0.2972637 0.3146710 0.3323865 0.3503364
G-Model   0.2640878 0.2806318 0.2974661 0.3144927 0.3316089 0.3487091
               103       104       105
True      0.4011628 0.4014352 0.4337094
Simulated 0.4008473 0.4018857 0.4330940
K-Model   0.3684420 0.3866219 0.4047927
G-Model   0.3656881 0.3824434 0.3988778

$stdv
                       80           81           82           83
Simulated(ML) 1.295237e-05 1.322055e-05 1.475680e-05 1.564757e-05
K-Model       1.467446e-05 1.233523e-05 9.916383e-06 7.643427e-06
G-Model       1.525099e-05 1.098523e-05 7.476005e-06 5.663803e-06
                       84           85           86           87
Simulated(ML) 1.740495e-05 1.878968e-05 2.000516e-05 2.243458e-05
K-Model       6.029634e-06 5.896809e-06 7.365751e-06 9.625728e-06
G-Model       6.385240e-06 8.464733e-06 1.064752e-05 1.243950e-05
```

```
                          88            89            90            91
Simulated(ML) 2.469515e-05 2.736950e-05 3.104865e-05 3.517117e-05
K-Model       1.202806e-05 1.421250e-05 1.594830e-05 1.708747e-05
G-Model       1.364971e-05 1.422062e-05 1.421924e-05 1.390575e-05
                          92            93            94            95
Simulated(ML) 4.069431e-05 4.704556e-05 5.251796e-05 6.239946e-05
K-Model       1.760857e-05 1.776324e-05 1.835458e-05 2.090794e-05
G-Model       1.385250e-05 1.495147e-05 1.798492e-05 2.311708e-05
                          96            97            98            99
Simulated(ML) 7.076429e-05 8.285561e-05 9.787397e-05 1.180788e-04
K-Model       2.701651e-05 3.739086e-05 5.209881e-05 7.114093e-05
G-Model       3.006142e-05 3.845461e-05 4.797693e-05 5.834386e-05
                         100           101           102           103
Simulated(ML) 1.416807e-04 0.0001755828 2.215190e-04 0.0002768915
K-Model       9.459768e-05 0.0001225914 1.552362e-04 0.0001926025
G-Model       6.928527e-05 0.0000805354 9.183218e-05 0.0001029218
                         104           105
Simulated(ML) 0.0003723230 0.0004850114
K-Model       0.0002346958 0.0002814423
G-Model       0.0001135657 0.0001235482

$difference
                           80            81            82            83
Simulated(ML)  8.866723e-06 -0.0000363039  1.131972e-09 3.972379e-06
K-Model       -4.591686e-03 -0.0030375369 -9.750862e-04 9.326734e-04
G-Model       -2.979461e-03 -0.0019041573 -2.791831e-04 1.239534e-03
                           84            85            86            87
Simulated(ML) -2.234246e-05 -1.270734e-06 6.680394e-06 -2.292332e-05
K-Model       -1.063192e-04  4.051833e-03 5.282625e-04  8.777024e-04
G-Model       -1.333776e-04  3.751951e-03 2.151288e-05  2.333342e-04
                          88            89            90            91
Simulated(ML) 3.286352e-05 1.353815e-05 3.507677e-05 -1.401112e-05
K-Model       5.425524e-03 3.632621e-03 2.248787e-03 -1.448311e-03
G-Model       4.714026e-03 2.923148e-03 1.606078e-03 -1.967497e-03
                           92            93            94            95
Simulated(ML) -5.925756e-06  6.327744e-05 -4.784234e-05 -2.059394e-05
K-Model       -3.120912e-03 -2.235120e-03 -7.752861e-03 -3.994481e-04
G-Model       -3.471765e-03 -2.389030e-03 -7.701798e-03 -1.602461e-04
                           96            97            98            99
Simulated(ML)  4.248692e-06 -4.153931e-05 -0.0001034016 -6.264250e-05
K-Model       -1.834308e-03  8.312685e-03 -0.0013308813 -2.306741e-04
G-Model       -1.452620e-03  8.759039e-03 -0.0009323169 -2.832379e-05
                          100           101           102           103
Simulated(ML)  6.564642e-05  0.0001059137  0.0001273670 -0.0003155113
K-Model       -1.409517e-02 -0.0099956273 -0.0167036487 -0.0327207849
G-Model       -1.427341e-02 -0.0107732738 -0.0183309011 -0.0354746667
                          104           105
Simulated(ML)  0.0004504638 -0.0006154977
K-Model       -0.0148133037 -0.0289167170
G-Model       -0.0189917637 -0.0348316445

$summary
              avg difference     avg stdv
Simulated(ML)   8.568162e-05 9.610517e-05
K-Model         6.550711e-03 5.677735e-05
G-Model         6.897079e-03 3.691306e-05

> sim300$nlsKfailures
[1] 4
> sim300$nlsGfailures
[1] 0
> sim300$which.nlsKfailed
[1]    3  503  875 1503
> sim300$which.nlsGfailed
[1] NA

#Calculation of Parameter Estimate Statistics (300k Simulation)

> min(sim300$simdata$"NLSE-K"[,1])
[1] 1.353612e-05
> max(sim300$simdata$"NLSE-K"[,1])
[1] 3.808734e-05
> min(sim300$simdata$"NLSE-K"[,2])
[1] 0.08277397
```

```
> max(sim300$simdata$"NLSE-K"[,2])
[1] 0.1031677
> min(sim300$simdata$"NLSE-K"[,3])
[1] 0.831647
> max(sim300$simdata$"NLSE-K"[,3])
[1] 7.07855
> (sum(sim300$simdata$"NLSE-K"[,1]))/1500
[1] 3.128497e-05
> (sum(sim300$simdata$"NLSE-K"[,2]))/1500
[1] 0.0929523
> (sum(sim300$simdata$"NLSE-K"[,3]))/1500
[1] 1.524542
> min(sim300$simdata$"NLSE-G"[,1])
[1] 3.921753e-06
> max(sim300$simdata$"NLSE-G"[,1])
[1] 3.021972e-05
> min(sim300$simdata$"NLSE-G"[,2])
[1] 0.0980972
> max(sim300$simdata$"NLSE-G"[,2])
[1] 0.1180489
> min(sim300$simdata$"NLSE-G"[,3])
[1] 0.0007588245
> max(sim300$simdata$"NLSE-G"[,3])
[1] 0.02974393
> (sum(sim300$simdata$"NLSE-G"[,1]))/1500
[1] 1.134860e-05
> (sum(sim300$simdata$"NLSE-G"[,2]))/1500
[1] 0.1081735
> (sum(sim300$simdata$"NLSE-G"[,3]))/1500
[1] 0.01709246
> sqrt((1500/1499)*(((sum((sim300$simdata$"NLSE-K"[,1])^2))/1500)-(((sum(sim300$simdata$"NLSE-K"[,1]))/1500)^2)))
[1] 2.518693e-06
> sqrt((1500/1499)*(((sum((sim300$simdata$"NLSE-K"[,2])^2))/1500)-(((sum(sim300$simdata$"NLSE-K"[,2]))/1500)^2)))
[1] 0.003336701
> sqrt((1500/1499)*(((sum((sim300$simdata$"NLSE-K"[,3])^2))/1500)-(((sum(sim300$simdata$"NLSE-K"[,3]))/1500)^2)))
[1] 0.553546
> sqrt((1500/1499)*(((sum((sim300$simdata$"NLSE-G"[,1])^2))/1500)-(((sum(sim300$simdata$"NLSE-G"[,1]))/1500)^2)))
[1] 3.939578e-06
> sqrt((1500/1499)*(((sum((sim300$simdata$"NLSE-G"[,2])^2))/1500)-(((sum(sim300$simdata$"NLSE-G"[,2]))/1500)^2)))
[1] 0.003369207
> sqrt((1500/1499)*(((sum((sim300$simdata$"NLSE-G"[,3])^2))/1500)-(((sum(sim300$simdata$"NLSE-G"[,3]))/1500)^2)))
[1] 0.004700127

#Testing of NLS Estimates

#Determination of the Simulation Histories which Produced Minimum or Maximum NLS Estimates for Gamma and Kappa

> which.min(sim$simdata$"NLSE-G"[,3])
[1] 1482
> which.min(sim$simdata$"NLSE-K"[,3])
[1] 1482
> which.max(sim$simdata$"NLSE-G"[,3])
[1] 3
> which.max(sim$simdata$"NLSE-K"[,3])
[1] 714
> which.min(sim300$simdata$"NLSE-G"[,3])
[1] 628
> which.min(sim300$simdata$"NLSE-K"[,3])
[1] 628
> which.max(sim300$simdata$"NLSE-G"[,3])
[1] 1233
> which.max(sim300$simdata$"NLSE-K"[,3])
[1] 153

#Example of Coding to Run NLS Test

> startfinderGtest(x=80:99,p=1-sim$stats$mean[1,1:20],min = c(1e-7,0.04,-0.2),max = c(1e-3,0.19,0.1),
    step = c(2,0.01,0.05))
$par
           a            m            g
1.059062e-05 1.082720e-01 1.740749e-02

$val [1] 0.0002221272
```

```
There were 50 or more warnings (use warnings() to see the first 50)

#Example of warning messages (first two)

Warning messages:
1: In log(1 + a * exp(m * x)) : NaNs produced
2: In log(1 + a * exp(m * (x + 1))) : NaNs produced

> g6s <- nlstest(nlse,"G")
> g6s
$na
[1] 1220
$converged
[1] 250
$ssr
0.0002221272
$same.ssr
[1] 250
$different.ssr
[1] 0
$stats
             alpha         beta        gamma
min   1.059025e-05 1.082718e-01 1.740699e-02
max   1.059094e-05 1.082724e-01 1.740805e-02
range 6.870679e-10 6.187148e-07 1.056722e-06
mean  1.059066e-05 1.082720e-01 1.740742e-02
SD    1.160886e-10 1.043506e-07 1.786258e-07

#Tests of NLS Estimates for Selected Life Tables in 601k Simulation

#Test Using Supplied Life Table and G-model
> g6s
$na
[1] 1220
$converged
[1] 250
$ssr
0.0002221272
$same.ssr
[1] 250
$different.ssr
[1] 0
$stats
             alpha         beta        gamma
min   1.059025e-05 1.082718e-01 1.740699e-02
max   1.059094e-05 1.082724e-01 1.740805e-02
range 6.870679e-10 6.187148e-07 1.056722e-06
mean  1.059066e-05 1.082720e-01 1.740742e-02
SD    1.160886e-10 1.043506e-07 1.786258e-07

#Test Using Supplied Life Table and K-model
> k6s
$na
[1] 7743
$converged
[1] 2757
$ssr
0.0002484908
$same.ssr
[1] 2757
$different.ssr
[1] 0
$stats
             alpha         beta        kappa
min   3.259700e-05 9.286452e-02 1.403434e+00
max   3.259750e-05 9.286512e-02 1.403484e+00
range 4.977609e-10 6.017056e-07 5.010287e-05
mean  3.259724e-05 9.286477e-02 1.403463e+00
SD    6.588756e-11 7.323917e-08 5.956308e-06

#Test Using Simulated Life Table History 1 and G-model
> g61
$na
[1] 1233
```

```
$converged
[1] 237
$ssr
0.0002029090
$same.ssr
[1] 237
$different.ssr
[1] 0
$stats
             alpha         beta        gamma
min   8.188760e-06 1.107807e-01 2.086537e-02
max   8.189427e-06 1.107815e-01 2.086662e-02
range 6.665852e-10 7.802038e-07 1.253209e-06
mean  8.189092e-06 1.107811e-01 2.086600e-02
SD    8.936285e-11 1.044205e-07 1.682129e-07

#Test Using Simulated Life Table History 1 and K-model
> k61
$na
[1] 7780
$converged
[1] 2720
$ssr
0.0002293891
$same.ssr
[1] 2720
$different.ssr
[1] 0
$stats
             alpha         beta        kappa
min   3.198615e-05 9.031257e-02 1.745145e+00
max   3.198647e-05 9.031305e-02 1.745218e+00
range 3.175892e-10 4.803478e-07 7.255962e-05
mean  3.198630e-05 9.031276e-02 1.745191e+00
SD    4.883453e-11 6.801819e-08 1.020385e-05

#Test Using Simulated Life Table History 1482 and G-model
#(NLS estimates for both gamma and kappa were minimum in this history)
> g61482
$na
[1] 1166
$converged
[1] 304
$ssr
0.0002608029
$same.ssr
[1] 304
$different.ssr
[1] 0
$stats
             alpha         beta        gamma
min   2.589531e-05 9.944465e-02 5.005565e-03
max   2.589721e-05 9.944534e-02 5.007006e-03
range 1.901898e-09 6.880867e-07 1.441032e-06
mean  2.589628e-05 9.944499e-02 5.006265e-03
SD    3.028044e-10 1.096416e-07 2.278051e-07

#Test Using Simulated Life Table History 1482 and K-model
#(NLS estimates for both gamma and kappa were minimum in this history)
> k61482
$na
[1] 7814
$converged
[1] 2686
$ssr
0.0002613421
$same.ssr
[1] 2686
$different.ssr
[1] 0
$stats
             alpha         beta        kappa
min   2.948215e-05 9.971361e-02 9.121275e-01
max   2.948301e-05 9.971421e-02 9.121465e-01
```

```
range 8.589166e-10 5.963737e-07 1.893942e-05
mean  2.948258e-05 9.971391e-02 9.121376e-01
SD    1.409543e-10 1.006792e-07 3.142929e-06

#Test Using Simulated Life Table History 3 and G-model
#(NLS estimate for gamma was maximum in this history)
> g63
$na
[1] 1224
$converged
[1] 246
$ssr
0.0002024522
$same.ssr
[1] 246
$different.ssr
[1] 0
$stats
             alpha         beta        gamma
min   4.504533e-06 1.167287e-01 2.803247e-02
max   4.504798e-06 1.167293e-01 2.803328e-02
range 2.649568e-10 5.655665e-07 8.151867e-07
mean  4.504647e-06 1.167290e-01 2.803293e-02
SD    4.349238e-11 9.298556e-08 1.337949e-07

#Test Using Simulated Life Table History 3 and K-model
#(NLS estimate for gamma was maximum in this history)
> k63
$na
[1] 9297
$converged
[1] 1203
$ssr
0.0002302841
$same.ssr
[1] 1203
$different.ssr
[1] 0
$stats
             alpha         beta        kappa
min   2.271475e-05 8.584189e-02 3.4745782585
max   2.272173e-05 8.584466e-02 3.4764041298
range 6.981545e-09 2.771206e-06 0.0018258713
mean  2.271763e-05 8.584303e-02 3.4756503574
SD    8.532821e-10 3.389376e-07 0.0002237596

#Test Using Simulated Life Table History 714 and K-model
#(NLS estimate for kappa was maximum in this history)
> k6714
$na
[1] 9289
$converged
[1] 1211
$ssr
0.0002747723
$same.ssr
[1] 1211
$different.ssr
[1] 0
$stats
             alpha         beta        kappa
min   2.239445e-05 8.573116e-02 3.5556195140
max   2.240292e-05 8.573442e-02 3.5578781073
range 8.473681e-09 3.258340e-06 0.0022585933
mean  2.239876e-05 8.573280e-02 3.5567348244
SD    9.527776e-10 3.675027e-07 0.0002546896

#Test Using Simulated Life Table History 714 and G-model
#(NLS estimate for kappa was maximum in this history)
> g6714
$na
[1] 1208
$converged
[1] 262
```

```
$ssr
0.0002631026
$same.ssr
[1] 262
$different.ssr
[1] 0
$stats
             alpha         beta         gamma
min   4.894183e-06 1.159403e-01 2.679214e-02
max   4.894440e-06 1.159408e-01 2.679288e-02
range 2.574420e-10 5.063781e-07 7.411405e-07
mean  4.894308e-06 1.159406e-01 2.679252e-02
SD    4.649097e-11 9.145381e-08 1.338916e-07

#Selected Tests of NLS Estimates for 300k Simulation

#Test Using Simulated Life Table History 1 and G-model
> g31
$na
[1] 1231
$converged
[1] 239
$ssr
0.0002119951
$same.ssr
[1] 239
$different.ssr
[1] 0
$stats
             alpha         beta         gamma
min   7.339825e-06 1.118514e-01 2.228472e-02
max   7.340384e-06 1.118521e-01 2.228588e-02
range 5.594923e-10 7.284534e-07 1.156002e-06
mean  7.340107e-06 1.118518e-01 2.228529e-02
SD    8.284462e-11 1.080830e-07 1.704972e-07

#Test Using Simulated Life Table History 1 and K-model
> k31
$na
[1] 7822
$converged
[1] 2678
$ssr
0.0002374287
$same.ssr
[1] 2678
$different.ssr
[1] 0
$stats
             alpha         beta         kappa
min   3.106537e-05 8.924798e-02 1.952034e+00
max   3.106589e-05 8.924852e-02 1.952143e+00
range 5.249559e-10 5.398806e-07 1.093047e-04
mean  3.106563e-05 8.924825e-02 1.952090e+00
SD    7.897073e-11 8.447395e-08 1.708820e-05

#Test Using Simulated Life Table History 628 and G-model
#(NLS estimates for both gamma and kappa were minimum in this history)
> g3628
$na
[1] 1209
$converged
[1] 261
$ssr
0.0003455593
$same.ssr
[1] 261
$different.ssr
[1] 0
$stats
             alpha         beta         gamma
min   3.021916e-05 9.809628e-02 7.567470e-04
max   3.022271e-05 9.809737e-02 7.592336e-04
range 3.545729e-09 1.087678e-06 2.486578e-06
```

```
mean  3.022093e-05 9.809683e-02 7.580093e-04
SD    5.472817e-10 1.685656e-07 3.781061e-07

#Test Using Simulated Life Table History 628 and K-model
#(NLS estimates for both gamma and kappa were minimum in this history)
> k3628
$na
[1] 7850
$converged
[1] 2650
$ssr
0.0003324193
$same.ssr
[1] 2650
$different.ssr
[1] 0
$stats
             alpha         beta        kappa
min   2.431854e-05 1.031675e-01 8.316332e-01
max   2.431957e-05 1.031683e-01 8.316517e-01
range 1.027278e-09 7.883184e-07 1.848967e-05
mean  2.431908e-05 1.031679e-01 8.316426e-01
SD    1.551351e-10 1.219374e-07 2.981000e-06

#Test Using Simulated Life Table History 153 and K-model
#(NLS estimate for kappa was maximum in this history)
> k3153
$na
[1] 10445
$converged
[1] 55
$ssr
0.0003212777
$same.ssr
[1] 55
$different.ssr
[1] 0
$stats
             alpha         beta       kappa
min   1.349490e-05 8.332777e-02 7.082472679
max   1.353009e-05 8.333582e-02 7.105471983
range 3.519600e-08 8.042900e-06 0.022999304
mean  1.351822e-05 8.333311e-02 7.090212834
SD    7.288417e-09 1.661521e-06 0.004758876

#Test Using Simulated Life Table History 153 and G-model
#(NLS estimate for kappa was maximum in this history)
> g3153
$na
[1] 1219
$converged
[1] 251
$ssr
0.0003145003
$same.ssr
[1] 251
$different.ssr
[1] 0
$stats
             alpha         beta        gamma
min   3.982493e-06 1.178829e-01 2.951353e-02
max   3.982700e-06 1.178834e-01 2.951424e-02
range 2.069064e-10 5.009775e-07 7.007243e-07
mean  3.982604e-06 1.178831e-01 2.951386e-02
SD    4.071274e-11 9.852932e-08 1.389695e-07

#Test Using Simulated Life Table History 1233 and G-model
#(NLS estimate for gamma was maximum in this history)
> g31233
$na
[1] 1217
$converged
[1] 253
$ssr
```

```
0.0002995457
$same.ssr
[1] 253
$different.ssr
[1] 0
$stats
             alpha         beta        gamma
min   3.921624e-06 1.180485e-01 2.974339e-02
max   3.921910e-06 1.180492e-01 2.974437e-02
range 2.862205e-10 7.047185e-07 9.831405e-07
mean  3.921757e-06 1.180489e-01 2.974391e-02
SD    4.664711e-11 1.147380e-07 1.612726e-07

#Test Using Simulated Life Table History 1233 and K-model
#(NLS estimate for gamma was maximum in this history)
> k31233
$na
[1] 10471
$converged
[1] 29
$ssr
0.0003081616
$same.ssr
[1] 29
$different.ssr
[1] 0
$stats
             alpha         beta       kappa
min   1.362392e-05 8.336110e-02 7.001938017
max   1.366398e-05 8.337026e-02 7.027624898
range 4.005624e-08 9.166772e-06 0.025686880
mean  1.363976e-05 8.336471e-02 7.017461731
SD    8.523240e-09 1.955909e-06 0.005467799
```

7.5 R Coding for Testing of NLS Estimates

```
Testing Functions

Note 1: Results of attempts at NLS estimation are not returned by startfinderGtest
or startfinderKtest but are stored globally in a list named nlse, which is supplied
to the function nlstest following the completion of startfinderGtest or startfinderKtest.
Note 2: The following vectors of minimum, maximum, and step values were supplied to
startfinderGtest and startfinderKtest for each test:
For startfinderGtest, min=c(1e-7,0.04,-0.2),max=c(1e-3,0.19,0.1),step=c(2,0.01,0.05)
For startfinderKtest, min=c(0.1,1e-7,0.04),max=c(5,1e-3,0.19),step=c(0.1,2,0.01)
Note 3: Lines that were too long to fit onto the page were split into multiple lines
in the coding shown below, and might need adjusted when running the code in R.

startfinderGtest <- function(x,p,min,max,step){
    mo <- min[2]
    A <- max[1]
    M <- max[2]
    G <- max[3]
    astep <- step[1]
    mstep <- step[2]
    gstep <- step[3]
    par <- c(NA,NA,NA)
    val <- 1e7
    nlse <<- vector("list")
    i <- 1
    while(mo <= M){
        ao <- min[1]
        while(ao <= A){
            go <- min[3]
            while(go <= G){
                snls <- try(nls(p ~ I(exp((1/m)*(log(1+a*exp(m*x))-
                            log(1+a*exp(m*(x+1))))-g)),start=list(a=ao,m=mo,g=go)),
                                silent=TRUE)
                if(length(snls) != 1){
                    ssr <- sum(resid(snls)^2)
                    nlse[[i]] <<- c((summary(snls))$par[,1],ssr)
                    par <- nlse[[i]][1:3]
                    }
                else nlse[[i]] <<- NA
                go <- go + gstep
                i <- i+1
                }
            ao <- ao * astep
            }
        mo <- mo + mstep
        }
    list(par=par,val=ssr)
    }

startfinderKtest <- function(x,p,min,max,step){
    mo <- min[3]
    K <- max[1]
    A <- max[2]
    M <- max[3]
```

```
    kstep <- step[1]
    astep <- step[2]
    mstep <- step[3]
    par <- c(NA,NA,NA)
    val <- 1e7
    nlse <<- vector("list")
    i <- 1
    while(mo <= M){
        ko <- min[1]
        while(ko <= K){
            ao <- min[2]
            while(ao <= A){
                snls <- try(nls(p ~ I(exp((k/m)*(log(1+a*exp(m*x))-
                            log(1+a*exp(m*(x+1))))))),start=list(k=ko,a=ao,m=mo)),
                                silent=TRUE)
                if(length(snls) != 1){
                    ssr <- sum(resid(snls)^2)
                    nlse[[i]] <<- c((summary(snls))$par[,1],ssr)
                    par <- nlse[[i]][1:3]
                    }
                else nlse[[i]] <<- NA
                ao <- ao * astep
                i <- i+1
                }
            ko <- ko + kstep
            }
        mo <- mo + mstep
        }
    list(par=par,val=ssr)
    }

nlstest <- function(nlse,model){
    j <- k <- h <- l <- s <- 0
    ssr <- NA
    t <- vector("list")
    best <<- matrix(NA,1,4)
    for(i in 1:length(nlse)){
        if(is.na(nlse[[i]][1])) {h <- h + 1}
        if(!is.na(nlse[[i]][1])){
            if(is.na(ssr)){
                ssr <- nlse[[i]][4]
                }
            best <<- rbind(best,c(nlse[[i]][1],nlse[[i]][2],nlse[[i]][3],nlse[[i]][4]))
            l <- l + 1
            }
        if(!is.na(nlse[[i]][1]) && round(nlse[[i]][4],digits=10) == round(ssr,digits=10))
            {j <- j + 1}
        if(!is.na(nlse[[i]][1]) && round(nlse[[i]][4],digits=10) != round(ssr,digits=10))
            {k <- k + 1}
        }
    n <- length(best[,1])
    best <- best[2:n,]
    n <- n-1
    stats <- matrix(NA,5,3)
    rownames(stats) <- c("min","max","range","mean","SD")
```

```
if(model=="G"){
    colnames(stats) <- c("alpha","beta","gamma")
    amin <- min(best[,1])
    amax <- max(best[,1])
    arange <- amax-amin
    amean <- sum(best[,1])/n
    astd <- sqrt((n/(n-1))*(((sum((best[,1])^2))/n)-(((sum(best[,1]))/n)^2)))
    stats[,1] <- c(amin,amax,arange,amean,astd)
    bmin <- min(best[,2])
    bmax <- max(best[,2])
    brange <- bmax-bmin
    bmean <- sum(best[,2])/n
    bstd <- sqrt((n/(n-1))*(((sum((best[,2])^2))/n)-(((sum(best[,2]))/n)^2)))
    stats[,2] <- c(bmin,bmax,brange,bmean,bstd)
    gmin <- min(best[,3])
    gmax <- max(best[,3])
    grange <- gmax-gmin
    gmean <- sum(best[,3])/n
    gstd <- sqrt((n/(n-1))*(((sum((best[,3])^2))/n)-(((sum(best[,3]))/n)^2)))
    stats[,3] <- c(gmin,gmax,grange,gmean,gstd)
    }
if(model=="K"){
    colnames(stats) <- c("alpha","beta","kappa")
    amin <- min(best[,2])
    amax <- max(best[,2])
    arange <- amax-amin
    amean <- sum(best[,2])/n
    astd <- sqrt((n/(n-1))*(((sum((best[,2])^2))/n)-(((sum(best[,2]))/n)^2)))
    stats[,1] <- c(amin,amax,arange,amean,astd)
    bmin <- min(best[,3])
    bmax <- max(best[,3])
    brange <- bmax-bmin
    bmean <- sum(best[,3])/n
    bstd <- sqrt((n/(n-1))*(((sum((best[,3])^2))/n)-(((sum(best[,3]))/n)^2)))
    stats[,2] <- c(bmin,bmax,brange,bmean,bstd)
    kmin <- min(best[,1])
    kmax <- max(best[,1])
    krange <- kmax-kmin
    kmean <- sum(best[,1])/n
    kstd <- sqrt((n/(n-1))*(((sum((best[,1])^2))/n)-(((sum(best[,1]))/n)^2)))
    stats[,3] <- c(kmin,kmax,krange,kmean,kstd)
    }
list(na=h,converged=l,ssr=ssr,same.ssr=j,different.ssr=k,stats=stats)
}
```