7-9-2018

Quantile Optimization in Stochastic Financial Planning Model

JIATIAN XU
University of Connecticut - Storrs, jiatian.xu@uconn.edu

Quantile Optimization in Stochastic Financial Planning Model

Jiatian Xu, PhD

University of Connecticut, [2018]

Abstract

The traditional algorithmic approach of financial planning is based on Monte Carlo simulations of yield curves and mortality. Financial advisors determine the withdrawal amount under an acceptable level of failure (target ruin) in the simulated future cash flows by using trial and error methods. The number of iterations with full credibility has allowed financial advisers and software systems to be precise about calculating a withdrawal level that achieves a target ruin. However, it is always extremely time consuming (several hours or days).

Rather than try and determine the optimal withdrawal level, this research creates a formulaic method to obtain a closed-form solution of the maximum withdrawal for each simulated scenario. And then we choose the quantile (target ruin) of all the maximum withdrawals as the final optimal solution. The runtime is dramatically decreased (within seconds). Based on this methodology, this research has also built a non-linear optimization model using an adjusted Lagrange method to solve the optimal allocation of each asset in the portfolio which can result in the optimal withdrawal level.

Keywords

Stochastic financial planning, Markov Chain Monte Carlo (MCMC) simulations, formulating method, quantile optimization, non-linear optimization model, Lagrange method

Quantile Optimization in Stochastic Financial Planning Model

Jiatian Xu

B.A., Southwestern University of Finance and Economics, [2006]

M.S., University of Connecticut, [2016]

A Dissertation

Submitted in Partial Fulfillment of the

Requirements for the Degree of

Doctor of Philosophy

at the

University of Connecticut

[2018]

APPROVAL PAGE

Doctor of Philosophy Dissertation

Quantile Optimization in Stochastic Financial Planning Model

Presented by

Jiatian Xu, B.A., M.S.

Major Advisor

_____

Jeyaraj Vadiveloo

Associate Advisor

_____

Emiliano Valdez

Associate Advisor

_____

Guojun Gan

University of Connecticut

[2018]

Table of Contents

Chapter 1 Introduction

One of the most robust areas of research in the field of financial planning is the ongoing research on methods to help clients determine how much they can withdraw from their retirement accounts to optimize their lifestyle while minimizing the chance of ruin due to portfolio volatility, mortality or longevity.

The commonly used traditional financial planning model is based on Monte Carlo simulations of the yield curves with a fixed mortality period. Usually, it will be the life/longevity expectancy. The inclusion of Monte Carlo simulations has allowed financial advisors to be more precise about the optimal withdrawal amount which can achieve an acceptable level of failure or success in the simulated future portfolio returns. A starting withdrawal level will be input into the model to see whether the number of failures in the Monte Carlo simulations can satisfy a client's risk tolerance. If not, we need to try another withdrawal number and redo this process until the output can achieve the desired success rate.

However, under a pre-determined allocation of assets in a retirement portfolio, this trial and error method usually takes eight to ten hours to get the final optimal withdrawal amount (if the number of iterations is set to be 1000 and the death is assumed to happen at age 95). Moreover, if we still use the same method to explore the optimal allocation of each asset which can maximize the final optimal withdrawal amount satisfying a client's target ruin, it might take several days, even weeks, to get the solution. It is definitely unacceptable for business applications.

This research creates a new method based on formulating the process of withdrawing and accumulating from a retirement portfolio. A closed form solution of the optimal withdrawal amount for each iteration is derived and then the appropriate quantile (target ruin) is chosen to be

the final optimal withdrawal amount. By using this new methodology and the application of R programming techniques, the runtime can be decreased to within seconds.

Moreover, based on this formulaic methodology, the fixed-period simulation model can be improved to a random-period simulation model. It means that the health condition status of the individual each year can be also incorporated into the Monte Carlo simulation and extended into a Markov Chain Monte Carlo (MCMC) process. In other words, in the MCMC model, the mortality period will be a random number in each simulation. Additionally, according to the different individual health statuses (healthy versus disabled) each year, the retirement living needs can be adjusted appropriately.

Furthermore, the above modeling or the optimal withdrawal level (quantile) is based on a pre-determined allocation of assets in a retirement portfolio. How to determine the ideal asset allocation is another very important question of the financial planning strategy. There are so many existing theories in this field. However, we cannot use those methods directly to solve this quantile optimization problem in the stochastic retirement financial planning process since our goal is not to maximize the returns from a portfolio or to minimize the volatility of a portfolio. For the retirement financial planning strategy, the final target is to maximize an individual's annual withdrawal level while achieving the desired likelihood of success.

This paper demonstrates a nonlinear optimization model to obtain the optimal allocation of assets of a retirement portfolio that maximizes an individual's annual withdrawal level while achieving a minimum level of ruin. The optimal asset allocation in a retirement portfolio can be solved precisely and instantaneously by using this model.

Chapter 2 Literature Review

The topic of optimal withdrawal strategies in retirement is one of the most active areas of research and discussion in the retirement planning literature.

Stout and Mitchell (2006) developed a dynamic model of retirement withdrawal planning that allows retirees and financial planners to improve the probability of retirement portfolio success while simultaneously increasing the average withdrawal rate.

Frank, Mitchell, and Blanchett (2011) created a model based on a fixed-period simulation and captured a failure rate at some future age. The authors develop an age-based three- dimensional distribution model that illustrates a retiree transition from early retirement into later retirement, including superannuated years for the long-lived who continue to survive. The model for this concept development simultaneously:

1) Establishes an age-based distribution model.

2) Incorporates current age life expectancy directly into the model.

3) Addresses survivorship into superannuated ages.

4) Addresses sequence risk to incorporate decisions due to market changes as the retiree ages.

Rather than generic distribution periods non-correlated with retirement age, the authors have developed an age-based model that uses expected longevity to define the length of the current distribution period (DP). The current DP changes dynamically each year 1) as the retiree ages and 2) by adjusting for the percentage of those statistically expected to outlive the current age's expected longevity.

Steiner (2014) outlined a return to fundamentals with an actuarial approach to calculate the amount of withdrawal based on the “risk-free interest rate” and the greater of the retiree’s life expectancy or age 95. The method focuses on alternative systematic withdrawal approaches and concludes that "The Actuarial Approach" is a better approach than alternative systematic withdrawal approaches that are commonly used. The model shows that combining life insurance annuity products with strategic withdrawals from accumulated assets to be an efficient strategy for managing risks in retirement. An optimal systematic withdrawal strategy is one that can coordinate the various sources of retirement income to meet a retiree's financial goals in retirement.

Waring and Siegel (2015) expanded on the actuarial approach, by determining an annually recalculated virtual annuity, or ARVA (Annually Recalculated Virtual Annuity) strategy that could be derived from an annuity payout calculation, repeated each period. They suggest a personal annuity structure created by relating an asset portfolio’s value to a stream of annual spending over a term related to the remaining lifespan of the investor. The innovation in Siegel and Waring’s approach is to call for yearly recalculation of the personal annuity — they call it an “Annually Recalculated Virtual Annuity” — and to limit spending in each year to the newly calculated annuity amount, reflecting updated values of assets and real rates of returns. After a year of strong investment returns, the next year’s spending can increase; this reflects the changed annuity relationship between the newly calculated accumulated asset value and the remaining investment term. After a tough year in the markets, the following year’s spending is adjusted down similarly.

Chapter 3 Methodology

3.1 Notations

3.1.1 MCMC Simulation

$S(t)$: The price of an asset at time t

μ: The mean rate of return of an asset

σ: The volatility of an asset

$W(t)$: A Brownian motion. For all $0 = t_0 < t_1 < \cdots t_m$, the increments

1) $W(t_1) - W(t_0),\ W(t_2) - W(t_1),\ \cdots,\ W(t_m) - W(t_{m-1})$ are independent and each of these increments is normally distributed with

2) $E\left[W(t_{i+1}) - W(t_i)\right] = 0$

3) $Var\left[W(t_{i+1}) - W(t_i)\right] = t_{i+1} - t_i$

Health status:

1 – Healthy

2 – Disabled

3 – Dead

$q(x,h)$: The probability that a healthy life (x) stays healthy and dies in the coming year

$q(x,d)$: The probability that a disabled life (x) dies in the coming year

i_x: The probability that a healthy life (x) gets disabled at the start of the year

3.1.2 Formulating of the MCMC process

b_0: The total initial assets in a retirement portfolio (non-qualified/qualified account)

b_t: The portfolio value at time t

N: The mortality period (a random number)

t: The points in time, t = 0, 1, 2, …, N

b_{NQ_t}: The non-qualified account value at time t (before withdrawing)

b_{Q_t}: The qualified account value at time t (before withdrawing)

d_t: The annual withdrawal sequence from a retirement portfolio

S_t: Social security incomes at time t

P_t: Pension incomes at time t

IC_t: The infusion of capital at time t

A_t: Other retirement incomes (insurance products, annuities or deferred annuities) at time t

LN_t: The annual retirement financial living needs of year t + 1. $LN_t = 0$ if t > N

I_i: The inflation rate of year i, i = 1, 2, …, N

T_i: The adjustment factor on the annual retirement financial living needs of year i, which depends on the health status at the beginning of each year, i = 1, 2, …, N

m: The number of assets in the retirement portfolio

$\overrightarrow{w_t}$: $\overrightarrow{w_t} = \begin{bmatrix} w_{1t} & w_{2t} & \cdots & w_{mt} \end{bmatrix}$

The vector of the assets allocation in the retirement portfolio in the future year t, $t > 0$

$\overrightarrow{r_t}$: $\overrightarrow{r_t} = \begin{bmatrix} r_{1t} & r_{2t} & \cdots & r_{mt} \end{bmatrix}$

The vector of the annual returns of each asset in the future year t, $t > 0$

3.2 Assumptions

Assumption 1: Individuals' purchase and consumption behaviors in daily life often are repetitive. So, the annual retirement living needs are predictable and usually increasing by an inflation rate.

Assumption 2: The annual retirement living needs depends on the individual's health status (healthy, disabled or dead).

Assumption 3: All the cashflows happen at the beginning of each year.

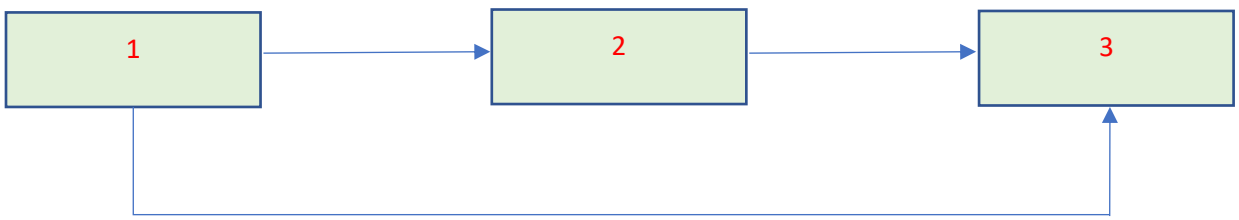
Consequently, we can have

$$LN_t = LN_0 \cdot \prod_{i=1}^{t}\left[(1+I_i)\cdot(1+T_i)\right] \text{ for } t>0 \qquad \text{(Formula 3.1)}$$

$$LN_t = (d_0 + P_0 + S_0 + IC_0 + A_0)\cdot \prod_{i=1}^{t}\left[(1+I_i)\cdot(1+T_i)\right] \text{ for } t>0 \qquad \text{(Formula 3.2)}$$

$$d_t = (d_0 + P_0 + S_0 + IC_0 + A_0)\cdot \prod_{i=1}^{t}\left[(1+I_i)\cdot(1+T_i)\right] - (P_t + S_t + IC_t + A_t) \text{ for } t>0 \qquad \text{(Formula 3.3)}$$

Assumption 4: Three different health statuses: healthy (1), disabled (2) and dead (3). The transition of the health status happens at the beginning of each year. Additionally, all the possible transitions are shown as the following chart. Once disabled, we assume the individual stays disabled until death.



3.3 Methodology

Simulation modeling solves real-world problems safely and efficiently. It is an important method of analysis which is easily verified, communicated, and understood. Across industries and disciplines, simulation modeling provides valuable solutions by giving clear insights into complex systems. Especially, in the field of actuarial science and financial mathematics, the simulation technique is usually applied for analyzing some very complicated distributions of random variables. In this research, the final target is to maximize individuals' withdrawal level from their retirement accounts in order to optimize their lifestyle while minimizing the chance of ruin due to portfolio volatility, mortality or longevity. There is no pre-existing distribution of the annual withdrawal amount in this stochastic financial process. So, this research is based on the Markov Chain Monte Carlo (MCMC) simulation process and creates a new methodology to solve the optimization problem rather than the traditional method trial and error. In the financial planning model, there are two stochastic processes: yield curves of different assets and an individual's health status. Many simulation methods have already been created by practitioners and in the financial literature. In this chapter, two different simulation techniques will be introduced. One is for the simulation of yield curves and the other is used to simulate the health status of the individual. And then a closed-form solution of this MCMC financial process will be demonstrated.

3.3.1 MCMC Simulation

3.3.1.1 Simulation of Yield Curves

In finance, there are three very important stochastic processes:

1) Arithmetic Process:

$$dS(t) = \mu \cdot dt + \sigma \cdot dW(t)$$

2) Geometric Process:

$$dS(t) = \mu \cdot S(t) \cdot dt + \sigma \cdot S(t) \cdot dW(t)$$

3) O – U Process:

$$dS(t) = \lambda \cdot (\phi - S(t)) \cdot dt + \sigma \cdot dW(t)$$

Assume that the price of each asset in a retirement portfolio is a Geometric Brownian motion. We can have the following formula to simulate the yield curve of each asset in a retirement portfolio:

$$S(T) = S(t) \cdot e^{\left(\mu - \frac{1}{2}\sigma^2\right)\cdot(T-t) + \sigma\sqrt{T-t}\cdot Z} \qquad \text{(Formula 3.4)}$$

Proof:

By using the $Itô - Doeblin$ formula for an $Itô$ process[1]:

$$df(t, X(t)) = f_t(t, X(t))dt + f_x(t, X(t))dX(t) + \frac{1}{2} f_{xx}(t, X(t))dX(t)dX(t) \qquad \text{(Formula 3.5)}$$

[1] Steven E. Shreve (2004). Chapter 4 Stochastic Calculus, *Stochastic Calculus for Finance II: Continuous-Time Models* (pp.132 – 147). Springer-Verlag New York.

let

$$f\left(t, X\left(t\right)\right)=\log\left(S\left(t\right)\right)$$

Then we can have $f_s=\frac{1}{S(t)}$, $f_{ss}=-\frac{1}{\left(S(t)\right)^2}$ and $f_t=0$

$$df=0\cdot dt+\frac{1}{S}\cdot dS+\frac{1}{2}\left(-\frac{1}{S^2}\right)\cdot\left(dS\right)^2$$

$$df=\frac{1}{S}\cdot\left(\mu\cdot S\cdot dt+\sigma\cdot S\cdot dW\right)-\frac{1}{2}\left(\frac{1}{S^2}\right)\cdot\left(\mu^2\cdot S^2\cdot\left(dt\right)^2+2\mu\cdot\sigma\cdot S^2\cdot dt\cdot dW+\sigma^2\cdot S^2\cdot\left(dW\right)^2\right)$$

According to the properties of Brownian motion $\left(dt\right)^2=0$, $dt\cdot dW\left(t\right)=0$ and $\left(dW\left(t\right)\right)^2=dt$, we can have

$$df\left(t\right)=\left(\mu-\frac{1}{2}\sigma^2\right)dt+\sigma dW\left(t\right)$$
$$\Rightarrow\int_t^T df\left(u\right)=\int_t^T\left(\mu-\frac{1}{2}\sigma^2\right)du+\int_t^T\sigma dW\left(u\right)$$
$$\Rightarrow f\left(T\right)-f\left(t\right)=\left(\mu-\frac{1}{2}\sigma^2\right)\cdot\left(T-t\right)+\int_t^T\sigma dW\left(u\right)$$

Where $\int_t^T\sigma dW\left(u\right)$ is a normally distributed random number, whose expectancy is zero and variance is $\int_t^T\sigma^2dt$. We set

$$\int_t^T\sigma dW\left(u\right)=N\left[0,\ \int_t^T\sigma^2dt\right]=N\left[0,\ \sigma^2\left(T-t\right)\right]=N\left[0,\ 1\right]\cdot\sigma\sqrt{T-t}=\sigma\sqrt{T-t}\cdot Z$$

$$Z=N\left[0,\ 1\right]$$

Z is a random number that follows the standard normal distribution. So, we can have

$$f(T)-f(t)=\left(\mu-\frac{1}{2}\sigma^2\right)\cdot(T-t)+\sigma\sqrt{T-t}\cdot Z$$

$$\Rightarrow \log S(T)-\log S(t)=\left(\mu-\frac{1}{2}\sigma^2\right)\cdot(T-t)+\sigma\sqrt{T-t}\cdot Z$$

Finally,

$$S(T)=S(t)\cdot e^{\left(\mu-\frac{1}{2}\sigma^2\right)\cdot(T-t)+\sigma\sqrt{T-t}\cdot Z} \qquad \#$$

However, the returns of different assets should be related in the simulated future cash flows. The most popular way to generate correlated random numbers is using the Cholesky decomposition.

Any symmetric positive-definite matrix, M , can be written as

$$M=U^T DU \qquad \text{(Formula 3.6)}$$

U : An upper triangular matrix

D : A diagonal matrix

If the variance-covariance matrix of assets in a portfolio, Σ , is symmetric positive-definite, it can be written as

$$\begin{aligned}\Sigma = M &= U^T DU \\ &= \left(U^T\sqrt{D}\right)\left(\sqrt{D}U\right) \\ &= \left(\sqrt{D}U\right)^T\left(\sqrt{D}U\right)\end{aligned}$$

Set $C=\sqrt{D}U$, then $\Sigma=C^T C$, C is called the Cholesky decomposition of Σ .

The correlated random returns can be generated by using the following formula

$$\text{depRand} = \text{indRand} \times C = C^T \times \left(\text{indRand}\right)^T \qquad \text{(Formula 3.7)}$$

However, if the variance-covariance matrix is not symmetric positive-definite, we need to compute the nearest correlation matrix (NCM). Nicholas J. Higham (2002) created a methodology to solve this problem in a finance application. He showed that for a certain class of weights, the nearest correlation matrix has correspondingly many zero eigenvalues and that this fact can be exploited in the computation.[2] By using R, we can just use the function "nearPD" to easily get the NCM.

3.3.1.2 Simulation of Health Status

The simulation of health status is based on the mortality and incidence rate tables. Assume that (x) is healthy at the beginning of the year. To simulate the health status at the end of this year, we generate two random numbers (R1 and R2) which are both following the Uniform distribution [0, 1]. R1 is to be compared with i_x to decide whether this person will get disabled at the beginning of the year. R2 is to be compared with $q(x,h)$ or $q(x,d)$ to decide whether this person will be still alive at the end of the year. Under the UUD (Uniform Distribution of Deaths) assumption, we have the following rules:

$R1 > i_x$	$R2 > q(x,h)$	1	Healthy alive at x+1
$R1 \le i_x$	$R2 > q(x,d)$	2	Disabled alive at x+1
$R1 > i_x$	$R2 \le q(x,h)$	3	Healthy dead at x+1
$R1 \le i_x$	$R2 \le q(x,d)$	3	Disabled dead at x+1

[2] Nicholas J. Higham. (2002) Computing the Nearest Correlation Matrix—A Problem from Finance, IMA J. Numer. Anal. 22, 329–343, 2002.

For an arbitrary example, if an individual (x) is healthy at the beginning of the year, we may have the following simulated health statuses at x + 1.

R1	i_x	Disabled condition at the beginning of the year of age x	R2	$q(x,h)$	$q(x,d)$	Death condition at the end of the year of age x	Status
0.2	0.1	Not Disabled	0.005	0.15	0.2	Dead	3
0.05		Disabled	0.1			Dead	3
0.05		Disabled	0.3			Disabled alive	2
0.3		Not Disabled	0.25			Healthy alive	1

When the simulation moves to the next year (age x + 1), if the individual is still healthy at x + 1, we need to repeat the above process. If the individual is disabled at x + 1, we just need to compare a new R2 with $q(x+1,d)$ to determine whether this person will be dead or still disabled at x + 2 since the health status cannot go back to 1 from 2 according to **Assumption 4** in section 3.2.

3.3.2 Formulating of MCMC

Recursive formulas of the retirement portfolio value at the beginning of each year (before withdrawing):

$$b_{NQ_0}$$
$$b_{Q_0}$$
$$b_0 = b_{NQ_0} + b_{Q_0}$$

$$b_{NQ_1} = \left(b_{NQ_0} - d_0\right)\cdot \overrightarrow{w_1}\left(1+\overrightarrow{r_1}\right)$$
$$b_{Q_1} = b_{Q_0}\cdot \overrightarrow{w_1}\left(1+\overrightarrow{r_1}\right)$$
$$b_1 = b_{NQ_1} + b_{Q_1} = \left(b_0 - d_0\right)\cdot \overrightarrow{w_1}\left(1+\overrightarrow{r_1}\right) = b_0\cdot \overrightarrow{w_1}\left(1+\overrightarrow{r_1}\right) - d_0\cdot \overrightarrow{w_1}\left(1+\overrightarrow{r_1}\right)$$

$$b_{NQ_2} = \left(b_{NQ_1} - d_1\right)\cdot \overrightarrow{w_2}\left(1+\overrightarrow{r_2}\right)$$
$$b_{Q_2} = b_{Q_1}\cdot \overrightarrow{w_2}\left(1+\overrightarrow{r_2}\right)$$
$$b_2 = b_{NQ_2} + b_{Q_2} = \left(b_1 - d_1\right)\cdot \overrightarrow{w_2}\left(1+\overrightarrow{r_2}\right) = \left(b_0 - d_0\right)\cdot \overrightarrow{w_1}\left(1+\overrightarrow{r_1}\right)\cdot \overrightarrow{w_2}\left(1+\overrightarrow{r_2}\right) - d_1\cdot \overrightarrow{w_2}\left(1+\overrightarrow{r_2}\right)$$
$$b_2 = \left(b_0 - d_0\right)\cdot \overrightarrow{w_1}\left(1+\overrightarrow{r_1}\right)\cdot \overrightarrow{w_2}\left(1+\overrightarrow{r_2}\right) - \left[\left(d_0 + P_0 + S_0 + IC_0 + A_0\right)\cdot\left(1+I_1\right)\cdot\left(1+T_1\right) - \left(P_1 + S_1 + IC_1 + A_1\right)\right]\cdot \overrightarrow{w_2}\left(1+\overrightarrow{r_2}\right)$$
$$b_2 = \left(b_0 - d_0\right)\cdot \overrightarrow{w_1}\left(1+\overrightarrow{r_1}\right)\cdot \overrightarrow{w_2}\left(1+\overrightarrow{r_2}\right) - \left(d_0 + P_0 + S_0 + IC_0 + A_0\right)\cdot\left(1+I_1\right)\cdot\left(1+T_1\right)\cdot \overrightarrow{w_2}\left(1+\overrightarrow{r_2}\right) + \left(P_1 + S_1 + IC_1 + A_1\right)\cdot \overrightarrow{w_2}\left(1+\overrightarrow{r_2}\right)$$
$$b_2 = \left(b_0 - d_0\right)\cdot \prod_{t=1}^{2} \overrightarrow{w_t}\left(1+\overrightarrow{r_t}\right) - \left(d_0 + P_0 + S_0 + IC_0 + A_0\right)\cdot \sum_{t=2}^{2}\left(\prod_{i=1}^{t-1}\left[\left(1+I_i\right)\cdot\left(1+T_i\right)\right]\cdot \prod_{j=t}^{2} \overrightarrow{w_j}\left(1+\overrightarrow{r_j}\right)\right) + \sum_{t=2}^{2}\left[\left(P_{t-1} + S_{t-1} + IC_{t-1} + A_{t-1}\right)\cdot \prod_{j=t}^{2} \overrightarrow{w_j}\left(1+\overrightarrow{r_j}\right)\right]$$

$$b_{NQ_3} = \left(b_{NQ_2} - d_2\right)\cdot \overrightarrow{w_3}\left(1+\overrightarrow{r_3}\right)$$
$$b_{Q_3} = b_{Q_2}\cdot \overrightarrow{w_3}\left(1+\overrightarrow{r_3}\right)$$
$$b_3 = b_{NQ_3} + b_{Q_3} = \left(b_2 - d_2\right)\cdot \overrightarrow{w_3}\left(1+\overrightarrow{r_3}\right) = \left(b_0 - d_0\right)\cdot \overrightarrow{w_1}\left(1+\overrightarrow{r_1}\right)\cdot \overrightarrow{w_2}\left(1+\overrightarrow{r_2}\right)\cdot \overrightarrow{w_3}\left(1+\overrightarrow{r_3}\right) - d_2\cdot \overrightarrow{w_3}\left(1+\overrightarrow{r_3}\right)$$
$$b_3 = \left(b_0 - d_0\right)\cdot \prod_{t=1}^{3} \overrightarrow{w_t}\left(1+\overrightarrow{r_t}\right) - \left(d_0 + P_0 + S_0 + IC_0 + A_0\right)\cdot \sum_{t=2}^{3}\left(\prod_{i=1}^{t-1}\left[\left(1+I_i\right)\cdot\left(1+T_i\right)\right]\cdot \prod_{j=t}^{3} \overrightarrow{w_j}\left(1+\overrightarrow{r_j}\right)\right) + \sum_{t=2}^{3}\left[\left(P_{t-1} + S_{t-1} + IC_{t-1} + A_{t-1}\right)\cdot \prod_{j=t}^{3} \overrightarrow{w_j}\left(1+\overrightarrow{r_j}\right)\right]$$

⋮ ⋮

$$b_N = (b_0 - d_0)\cdot\prod_{t=1}^{N}\overrightarrow{w_t}\left(1+\vec{r_t}\right) - \left(d_0 + P_0 + S_0 + IC_0 + A_0\right)\cdot\sum_{t=2}^{N}\left(\prod_{i=1}^{t-1}\left[(1+I_i)\cdot(1+T_i)\right]\cdot\prod_{j=t}^{N}\overrightarrow{w_j}\left(1+\vec{r_j}\right)\right) + \sum_{t=2}^{N}\left[\left(P_{t-1}+S_{t-1}+IC_{t-1}+A_{t-1}\right)\cdot\prod_{j=t}^{N}\overrightarrow{w_j}\left(1+\vec{r_j}\right)\right]$$

Set

$$f_1^k = \prod_{t=1}^{k}\overrightarrow{w_t}\left(1+\vec{r_t}\right) \qquad \text{(Formula 3.8)}$$

$$f_2^k = \sum_{t=2}^{k}\left(\prod_{i=1}^{t-1}\left[(1+I_i)\cdot(1+T_i)\right]\cdot\prod_{j=t}^{k}\overrightarrow{w_j}\left(1+\vec{r_j}\right)\right) \qquad \text{(Formula 3.9)}$$

$$f_3^k = \sum_{t=2}^{k}\left[\left(P_{t-1}+S_{t-1}+IC_{t-1}+A_{t-1}\right)\cdot\prod_{j=t}^{k}\overrightarrow{w_j}\left(1+\vec{r_j}\right)\right] \qquad \text{(Formula 3.10)}$$

And then, we can have

$$b_N = (b_0 - d_0)\cdot f_1^{k=N} - \left(d_0 + P_0 + S_0 + IC_0 + A_0\right)\cdot f_2^{k=N} + f_3^{k=N} \qquad \text{(Formula 3.11)}$$

If we define the ruin in the MCMC process as the portfolio value is smaller than zero during an individual's retirement stage, for a specific scenario, in order to maximize the annual withdrawal, it looks like that we just need to set

$$b_N = (b_0 - d_0)\cdot f_1^{k=N} - \left(d_0 + P_0 + S_0 + IC_0 + A_0\right)\cdot f_2^{k=N} + f_3^{k=N} = 0 \qquad \text{(Formula 3.12)}$$

and then solve the initial withdrawal level as

$$d_0^{k=N} = \frac{b_0\cdot f_1^{k=N} - \left(S_0 + P_0 + IC_0 + A_0\right)\cdot f_2^{k=N} + f_3^{k=N}}{f_1^{k=N} + f_2^{k=N}} \qquad \text{(Formula 3.13)}$$

However, it could be wrong for some special cases since the portfolio value may be lower than zero before N and then go back to zero at N. In other words, ruin could happen before death.

Mathematically, there are two driven factors that can make it happen:

1) The annual portfolio return rate is less than - 100%

For example, the portfolio value at the beginning of one year is -1,000 and the simulated portfolio return rate is -120%. Consequently, the portfolio value at the beginning of the next year will be

$$-1000\times\left(1-120\%\right)=200$$

2) The cash inflows from other retirement income sources (e.g. infusion of capital, deferred annuity incomes, etc.)

For example, the portfolio value at the end of one year is -1,000. However, at the beginning of the next year, there will be amount of 20,000 deferred annuity incomes. So, the portfolio value at the beginning of the next year will be

$$-1000+20{,}000=19{,}000$$

The first condition can be avoided by controlling the simulation of the yield curve. However, for the second condition, the author has created a new method to avoid it:

For the p^{th} simulation, set

$${}_{p}b_{k}=0\ ,\ k=1,\ 2,\ \ldots\ ,\ N$$

It means that each portfolio value during the retirement stage will be checked.

According to the Formula 3.13, we can have a sequence of d_0 s. They are represented as

$${}_{p}d_{0}^{k=1},\ {}_{p}d_{0}^{k=2},\ \cdots,\ {}_{p}d_{0}^{k=N}$$

So, the maximum of the initial withdrawal level for this p^{th} simulation should be

$$ {}_p d_0^{\max} = \min\left({}_p d_0^{k=1},\ {}_p d_0^{k=2},\ \cdots,\ {}_p d_0^{k=N} \right) \qquad \text{(Formula 3.14)} $$

For all the iterations, we can have the following vector which includes the maximum initial withdrawal level for each simulation.

$$ \overrightarrow{d_0^{\max}} = \left[{}_{p=1} d_0^{\max} \quad {}_{p=2} d_0^{\max} \quad \cdots \quad {}_{p=s} d_0^{\max} \right], \text{ s = the total number of iterations} $$

Consequently, the final maximum of the initial withdraw level under a specific ruin target will be

$$ d_0^{Opt} = quantile\left(\overrightarrow{d_0^{\max}},\ probs = \text{ruin target} \right) \qquad \text{(Formula 3.15)} $$

where p = 1, 2, …, the total number of iterations

Generally, if the ruin is defined as

$$ \begin{cases} {}_p b_k < M,\ k = 0,\ 1,\ 2,\ \ldots,\ N-1,\ M \geq 0 \\ or \\ {}_p b_{k=N} < B,\ B \geq 0 \end{cases} $$

where M is the threshold of the portfolio value during the retirement stage and B is the bequest needs, we can have the final closed-form solution of the maximum withdrawal level in the financial planning model:

$$
\begin{cases}
{}_p d_0^k = \dfrac{b_0 \cdot f_1^k - \left(S_0 + P_0 + IC_0 + A_0\right) \cdot f_2^k + f_3^k - M}{f_1^k + f_2^k},\ k = 0,\ 1,\ 2,\ \ldots,\ N-1 \\
{}_p d_0^k = \dfrac{b_0 \cdot f_1^k - \left(S_0 + P_0 + IC_0 + A_0\right) \cdot f_2^k + f_3^k - B}{f_1^k + f_2^k},\ k = N \\
{}_p d_0^{\max} = \min\left({}_p d_0^{k=1},\ {}_p d_0^{k=2},\ \cdots,\ {}_p d_0^{k=N} \right) \\
\overrightarrow{d_0^{\max}} = \left[{}_{p=1} d_0^{\max} \quad {}_{p=2} d_0^{\max} \quad \cdots \quad {}_{p=s} d_0^{\max} \right],\ \text{s = the total number of iterations} \\
d_0^{Opt} = quantile\left(\overrightarrow{d_0^{\max}},\ probs = \text{ruin target} \right)
\end{cases}
$$

(Formula 3.16)

Finally, we complete the formulation of the MCMC process. By using this closed-form solution of the MCMC process, we can dramatically decrease the runtime of getting the optimal withdrawal level in the financial planning model.

However, there are still two problems we need to solve:

1) N , the mortality period is a random number in each simulation.

2) The number of iterations of the MCMC process is usually tremendous.

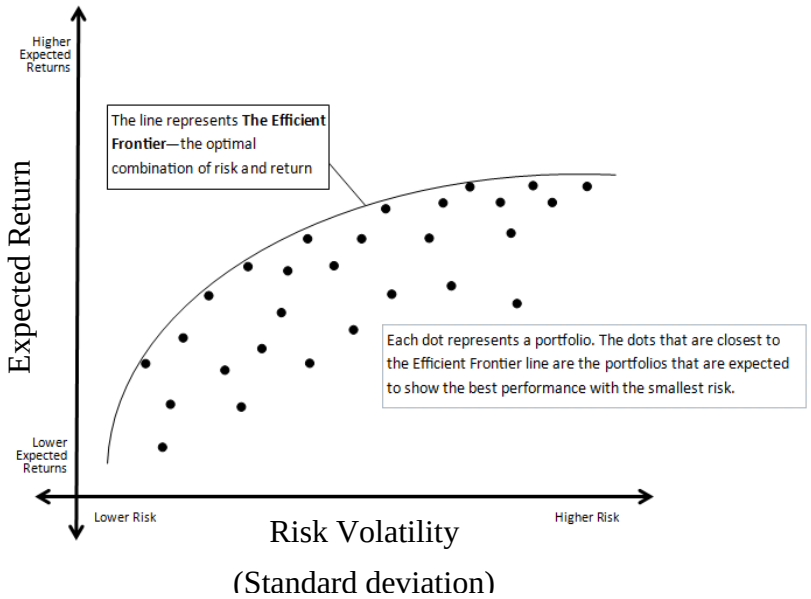
If we stick to the traditional coding method, it will be still very time consuming to get the final optimal solution. However, by using a clever way of R coding[3], the runtime could be controlled within seconds for 1,000,000 iterations.

[3] The R coding method can be found in the Appendix, Section 3 Closed-Form Solution of the Maximum Withdrawal under MCMC: # Preparation for Calculation Factors

3.3.3 Optimization of Assets Allocation

The closed-from solution of the maximum withdrawal level in section 3.3.2 is under the pre-determined asset allocation. However, the optimization of assets allocation under a specific ruin target is another very important part of the financial planning modeling process.

According to the classical efficient frontier theory, we know that more volatility will result in more expected returns. Given the volatility of a portfolio, we can find the optimal expected return on the efficient frontier line. However, we cannot directly use this theory to solve the optimal asset allocation in the financial planning model since our target is to maximize the withdrawal level rather than the expected return over some time horizon.



Obviously, a higher target ruin will lead to more weight on the risky asset. Under a higher risk tolerance, the optimal allocation of each asset will result in higher portfolio volatility. In other words, there is a positive correlation between the target ruin and the portfolio volatility. If we want to solve this optimization problem of asset allocation, the portfolio volatility will be a very

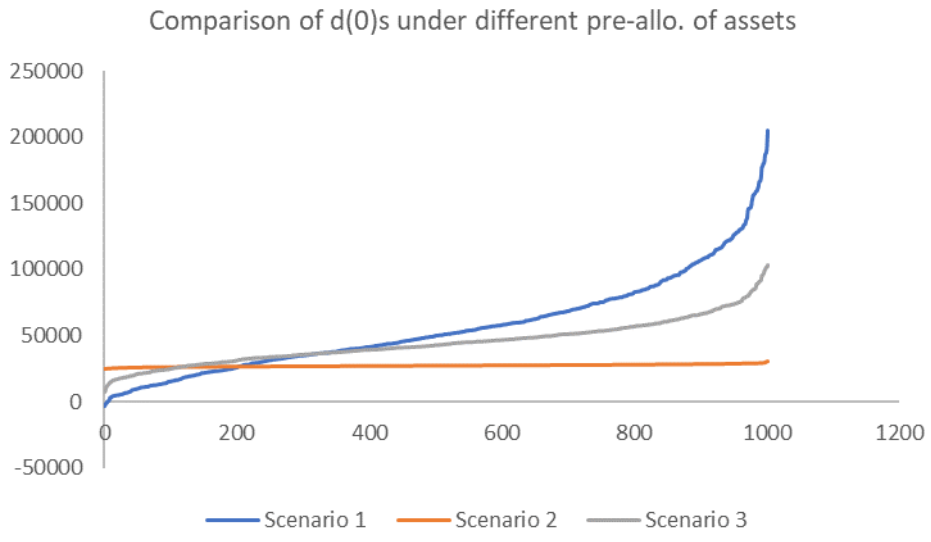
important constraint. The question is how to map the positive correlation between the target ruin and the portfolio volatility.

For example, suppose there are only two assets in a retirement portfolio.

	Mean	SD
Asset 1	2.15%	0.78%
Asset 2	9.54%	23.63%

Under the following three different scenarios of pre-allocation of assets and 1,000 simulations, we solve all ${}_p d_0^{\max}$, p = 1, 2, …, 1000 and sort them in ascending order.

	Scenario 1	Scenario 2	Scenario 3
Asset 1	0%	100%	50%
Asset 2	100%	0%	50%



X-axis represents sorted 1000 simulations and Y-axis represents ${}_p d_0^{\max}$ for each simulation.

We can find that:

1) The line of ${}_p d_0^{\max}$ will do a counterclockwise rotation with increasing portfolio volatility.

2) The cross point between different ${}_p d_0^{\max}$ lines is moving to the right with increasing portfolio volatility.

According to the above conclusion, we can use the movement of the cross point to represent the relationship between the target ruin and the portfolio volatility.

3.3.3.1 Mapping Between Target Ruin and Portfolio Volatility

1) Notations:

$\sigma =$ The portfolio volatility

$\sigma_{\min} =$ The minimum portfolio volatility

$\sigma_{\max} =$ The maximum portfolio volatility

$\sigma_{portfolio} = g(\text{target ruin})$: The mapping between the portfolio volatility and the target ruin

2) Methodology:

2.1) Under two pre-allocation of assets, one satisfies $\sigma = \sigma_{\min}$ and the other satisfies $\sigma = \sigma_{\min} + \varepsilon,\ \varepsilon \to 0+$, we calculate all the ${}_p d_0^{\max}$ to find out the quantile (q^l) where the cross point will happen, which is corresponding to the lower bound of the target ruin.

2.2)) Under two pre-allocation of assets, one satisfies $\sigma = \sigma_{\max}$ and the other satisfies $\sigma = \sigma_{\max} - \varepsilon,\ \varepsilon \to 0+$, we calculate all the ${}_p d_0^{\max}$ to find out the quantile (q^u) where the cross point will happen, which is corresponding to the upper bound of the target ruin.

According to many tests with full credibility, the author found that

$$\begin{cases} q^l = 0 \\ q^u \approx 0.5 \end{cases}$$

It shows that

1) If the target ruin is greater than 50%, the optimal allocation will be always 100% on the risky assets.

2) The function $\sigma_{portfolio} = g(\text{target ruin})$ is concave

Consequently, the function $\sigma_{portfolio} = g(\text{target ruin})$ has been built as

$$\sigma_{portfolio} = \min(\alpha \log(\text{target ruin}) + \beta,\ \sigma_{\max}) \qquad \text{(Formula 3.17)}$$

where α and β can be solved by using the following equation system:

$$\begin{cases} \alpha \log(\varepsilon) + \beta = \sigma_{\min} \\ \alpha \log(0.5) + \beta = \sigma_{\max} \end{cases} \qquad \text{(Formula 3.18)}$$

ε is a very tiny positive real number and usually set to be 0.001.

3.3.3.2 Methodology

$$
\begin{cases}
\text{Variables} & \vec{w} = \begin{bmatrix} w_1 & w_2 & \cdots & w_m \end{bmatrix} \\
\text{Objective Fucntion} & d_0^{Opt} = quantile\left(\vec{d_0^{\max}},\ probs = \text{ruin target}\right) \\
\text{Contraints} & \begin{cases} \vec{w} \cdot \sum \cdot \vec{w}' = \min(\alpha \log(\text{target ruin}) + \beta,\ \sigma_{\max}) \\ \sum_{i=1}^{m} w_i = 1 \\ w_i \geq 0 \end{cases}
\end{cases}
\qquad \text{(Formula 3.19)}
$$

This is a non-linear optimization problem. According to the complexity of the objective function, it is very difficult to get the closed-form solution of the optimal allocation of assets. However, we can use the Augmented Lagrange Multiplier Method to solve this problem. Actually, Yinyu Ye (1987) created an algorithm in his Ph.D. thesis to solve this general non-linear optimization problem[4]. And based on his algorithm, the R package "Rsolnp" has been built. By using the "solnp" function in this package, the optimal solution will be calculated within seconds under 1,000 simulations.

[4] Yinyu Ye (1987). Interior Algorithms for Linear, Quadratic, and Linearly Constrained Non-Linear Programming. Ph.D. Thesis, Department of EES, Stanford University.

Chapter 4 Analysis and Results

4.1 General Information and Assumptions

In the financial plan modeling, three sets of probabilities are in play in any projection: mortality, morbidity and the probability of "portfolio" ruin, or risk and return characteristics of the portfolio.

Portfolio risk and return characterizes in this analysis come from the Monte Carlo technique based on the geometric Brownian motion assumption which is stated in chapter 3. The mean and standard deviation of each asset return come from the historical data of the financial market. In this chapter, we assume that there are three different investable assets in the market. Additionally, individuals or clients usually have a pre-determined asset allocation that come from their personal experience or their financial advisors. The portfolio information is shown in Table 1.

Table 1

Assets

	Mean	Sd	Pre-determined Allocation
Asset A	2.15%	0.78%	20%
Asset B	5.28%	10.78%	50%
Asset C	6.58%	17.28%	30%

Mortality and morbidity table characteristics come from the MCMC approaches in chapter 3. A standard Monte Carlo randomization depends on two things: the number of iterations and the

period of simulation (time horizon). For example, if the maximum age in the mortality table is 117 and a client's current age is 65, the period of simulation or the time horizon of each scenario will be 117 – 65 = 52. In the methodology of chapter 3, we use "1" to represent healthy, "2" to represent disabled (or unhealthy), "3" to represent dead. For instance, one simulation of this individual's annual health status could be

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 2 & 2 & 2 & 3 & \cdots & 3 \end{bmatrix}_{1\times 53}$$

In this scenario, the individual will keep healthy from the current age 65 to age 68 and then become disabled for three years and then die at age 72. Especially, for this healthy status scenario, we only need to predict the annual assets returns for the future 72 – 65 = 7 years. So, if the number of iterations of yield curve projection (Iter. num. of y.c.) is 1,000 and the number of iterations of healthy condition (Iter. num. of h.c.) is 1,000, in order to get the full credibility of the MCMC process, the total number of simulations will be 1,000,000. All of these simulation assumptions and the other economic assumptions, like inflation rate, inflation rate increment (Inf.Growth), adjustment factor of the living needs when disabled (Live.need.adj) can be found in Table 2.

Table 2

Economic and Simulation Assumptions

Inflation	2.00%	Inf.Growth	0.03%
Live.need.adj	-10%	Longevity Target Age	95
Iter. num. of y.c.	1,000	Iter. num. of h.c.	1,000
(y.c. – yield curve)		(h.c – health condition)	

Besides the above information, the detailed client information is also needed for the financial plan modeling. For instance, the current age, gender, smoking status, current assets, retirement annual spending goal, desired success rate (risk tolerance or ruin target), the threshold requirement of the portfolio, bequest needs and some other sources of retirement income (e.g. pension, social security, capital infusion, etc.). All of them can be found in Tables 3 and 4.

Table 3

Current Client Details

Current Age	65	Gender	F
Current Assets	1,000,000	Smoke Status	NS
Ret. Spending Goal	70,000	Desired Success Rate	95%
Port. Threshold	-	Bequest Needs	-

Table 4

Retirement Income (Except for Port. Withdrawal)

Soc Sec	20,000	Pension	-
Soc Sec Start Age	65	Pension Start Age	65
Soc Sec Growth Rate	0%	Pension Growth Rate	0%

4.2 Basic Outputs and Analysis

4.2.1 The Optimal Initial Withdrawal Level d_0^{Opt}

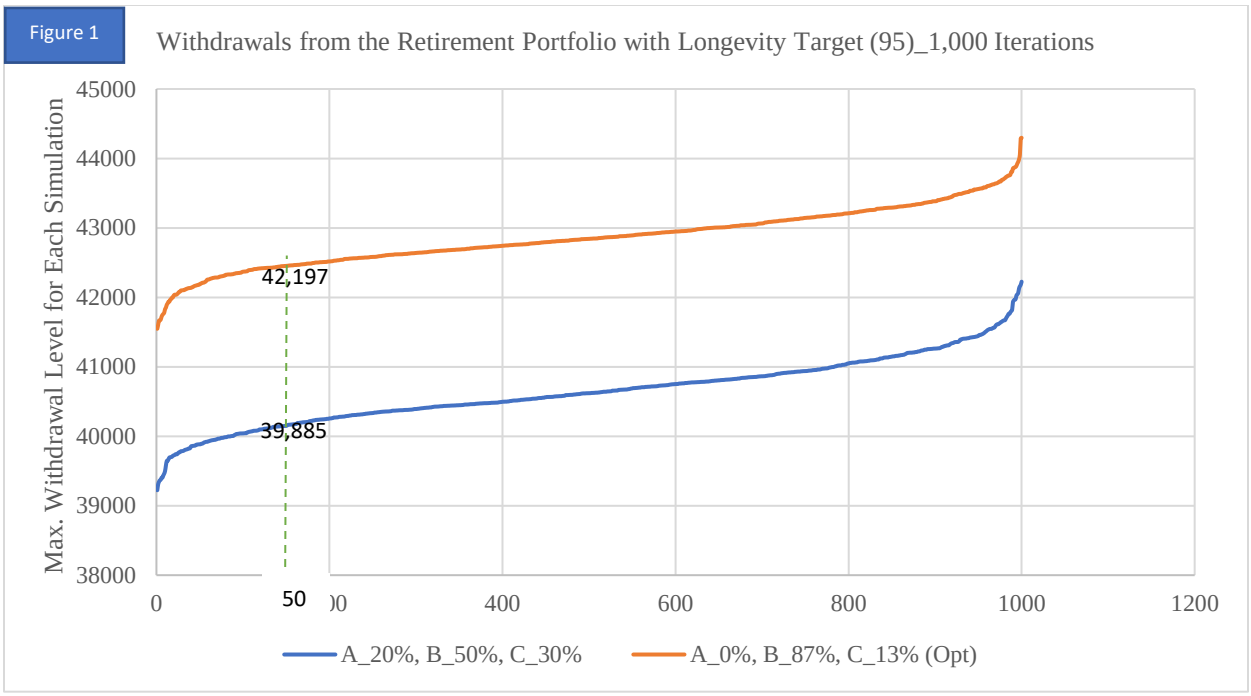
According to chapter 3, the optimal initial withdrawal level is defined as the maximum withdrawal level under the optimal asset allocation. From the inputs in Table 1, there is a pre-determined asset allocation from the client:

(Asset A_20%, Asset B_50%, Asset C_30%)

Firstly, by using the methodology in section 3.3.3.2, the optimal asset allocation can be solved very quickly (2.23 seconds / 1,000 simulations). It is shown as the following vector:

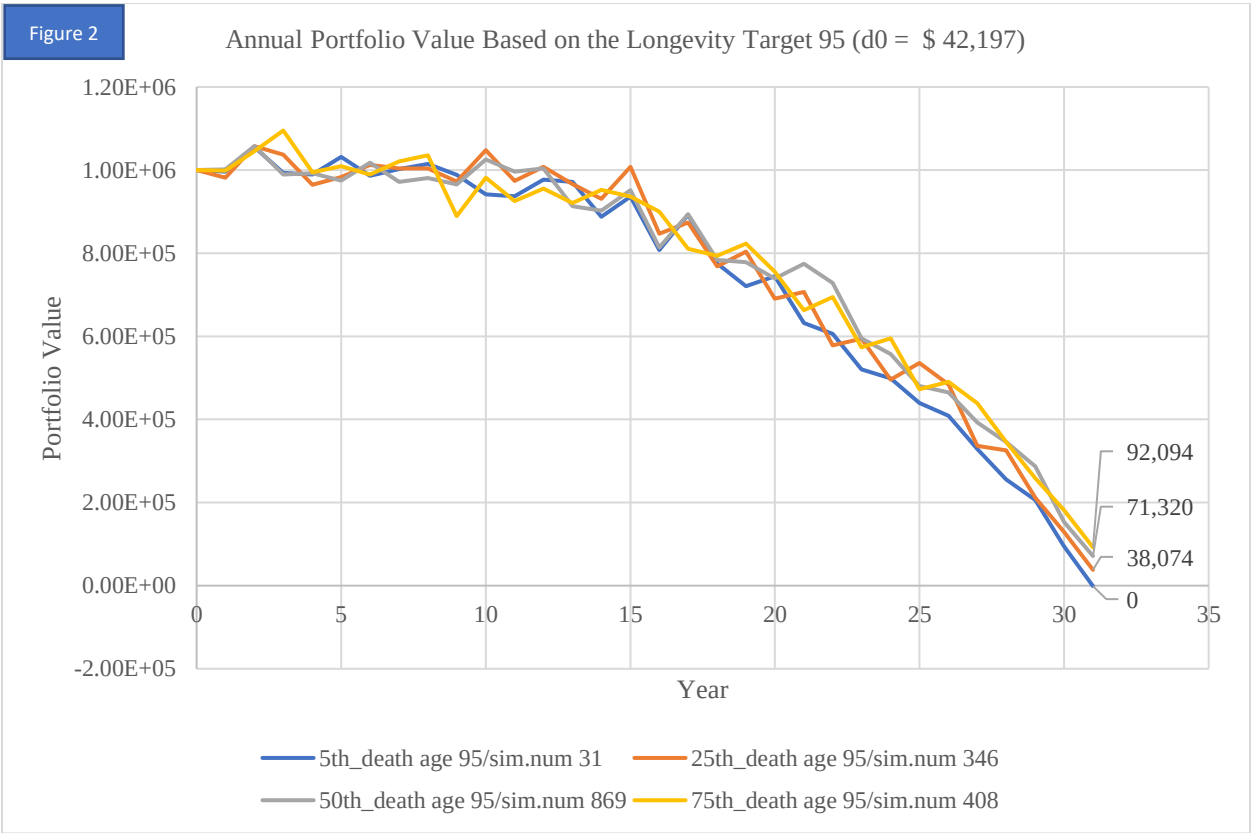
(Asset A_0%, Asset B_87%, Asset C_13%)

And then, by using the closed-form solution, we can solve ${}_pd_0^{\max}$ for each simulation. If we still use the traditional Monte Carlo method, where the simulation period is determined by a fixed longevity target age 95, do 1000 simulations and sort all ${}_pd_0^{\max}$ in ascending order. We can get the two lines of ${}_pd_0^{\max}$ in Figure 1. One is under the pre-determined asset allocation and the other is under the optimal asset allocation. The increment of the optimal withdrawal level d_0^{Opt} is very clear in Figure 1.



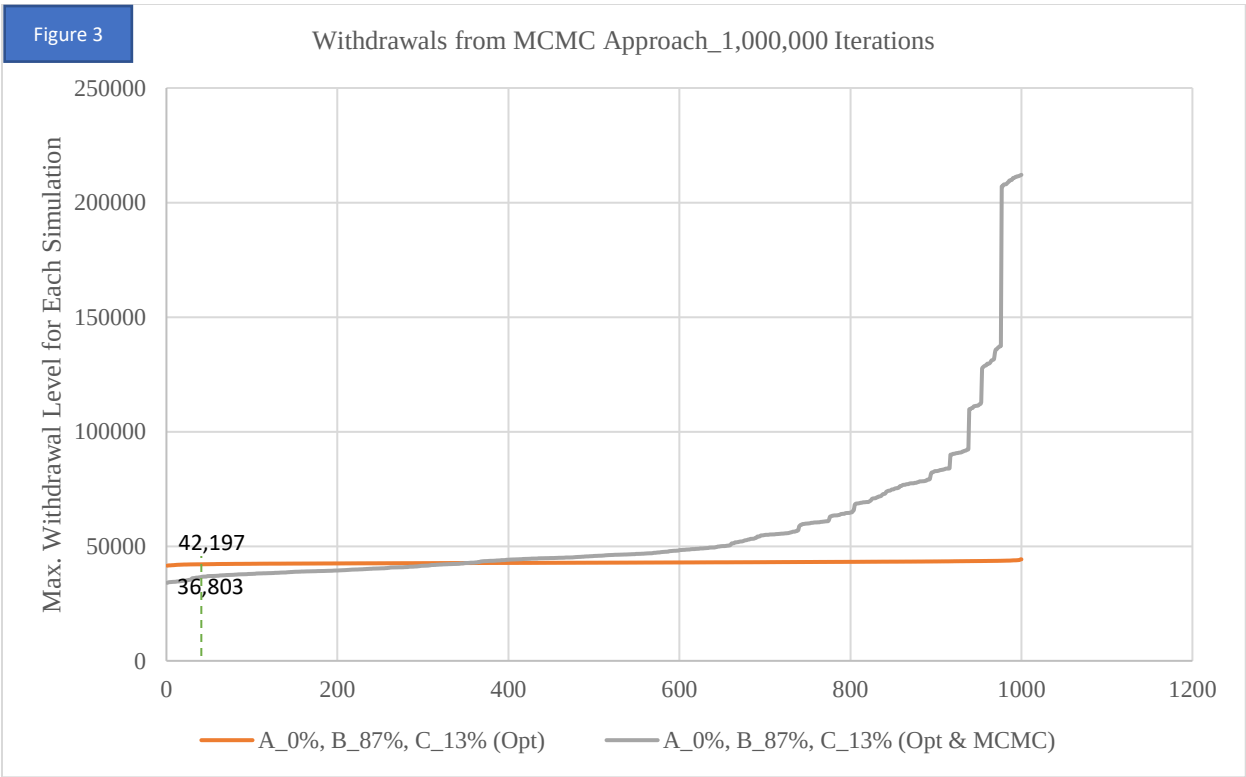
In Figure 1, under the pre-determined asset allocation, the fixed longevity target (95) and the desired success rate (95%), the maximum initial withdrawal is $ 39,885. However, if the optimization model of the asset allocation has been applied, the optimal withdrawal will be increased to $ 42,197.

In Figure 2, the 5th percentile line (blue) shows that the portfolio value at age 95 is exactly 0. It proves that the closed-form solution from the methodology is precise. Furthermore, the simulation number (31) of the 5th percentile line of the annual portfolio value can be easily located by using R. Figure 2 also shows the other percentile lines (25th, 50th and 75th) of the annual portfolio value. Under the fixed longevity target 95, whether a simulation is “good” or “bad” only depends on the simulated yield curve of each asset in the retirement portfolio.



However, the financial planning model based on a fixed longevity target is not complete. In fact, whether the longevity target (age 95) is a proper and safe assumption is very critical for retirement financial plan modeling. After incorporating the complete MCMC approaches which include the asset returns prediction and the simulation of health conditions, the result indicates that **age 95 is an improper assumption**. Furthermore, it could lead to a very high probability of ruin.

Figure 3



In Figure 3, under the optimal asset allocation, the MCMC process and the desired success rate (95%), the maximum initial withdrawal is decreased to $ 36,803. It shows **that the popular assumption of longevity risk (95) is improper and very risky**.

In fact, if the client still chooses to spend $ 42,197 at the beginning of retirement and increase the spending by an inflation factor for future years, **the ruin probability will bump up to 32.8%.** The summary of the result is shown in Table 5.

Table 5

	d_0^{Opt}	Ruin Prob. in MCMC
The longevity target (95)	42,197	32.8%
MCMC	36,803	5.0%

To be clearer, the different percentile lines of the annual portfolio value are shown in Figure 4.

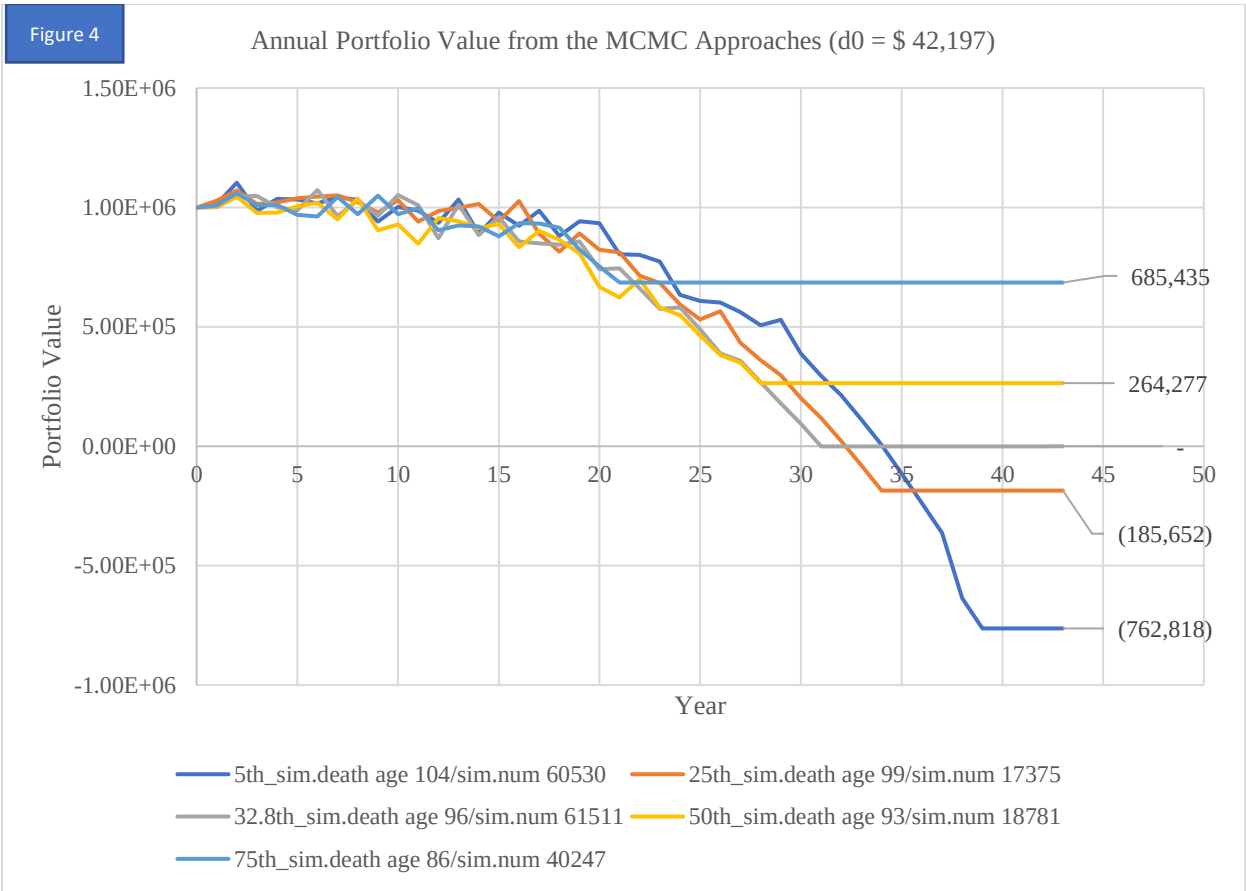


Figure 4 shows that if the client still chooses $ 42,197 as the "optimal" initial withdrawal level for the retirement stage, the ruin probability will be 32.8% according to the MCMC approaches. In fact, compared with the simulated death age of the 32.8th percentile line (grey), which is 96, the simulated death age of the 5th percentile line (dark blue) is 104, the one of the 25th percentile line (orange red) is 99. Both of them fail mainly because the unexpected longer lifetime.

By using this closed-form solution of the MCMC process, the percentile lines of portfolio value can be located very efficiently. In this case, the 5th percentile line of portfolio value corresponds the 60,530th simulation. In this simulation, the annual health conditions are projected as

$$\left[\underbrace{1\ 1\ \cdots\ 1}_{11}\ \underbrace{2\ 2\ \cdots\ 2}_{27}\ 3\ 3\ \cdots\ 3\right]$$

According to the above simulated health condition vector, the death age is

65 + 11 + 27 + 1 = 104

Moreover, we can see that the individual will become disabled from age 76 (65 + 11) to age 103. During this unhealthy period, the annual living needs will be decreased by 10% (Table 2).

The 25th percentile line of portfolio value corresponds to the 60,530th simulation. In this simulation, the annual health conditions are projected as

$$\left[\underbrace{1\ 1\ \cdots\ 1}_{22}\ \underbrace{2\ 2\ \cdots\ 2}_{11}\ 3\ 3\ \cdots\ 3\right]$$

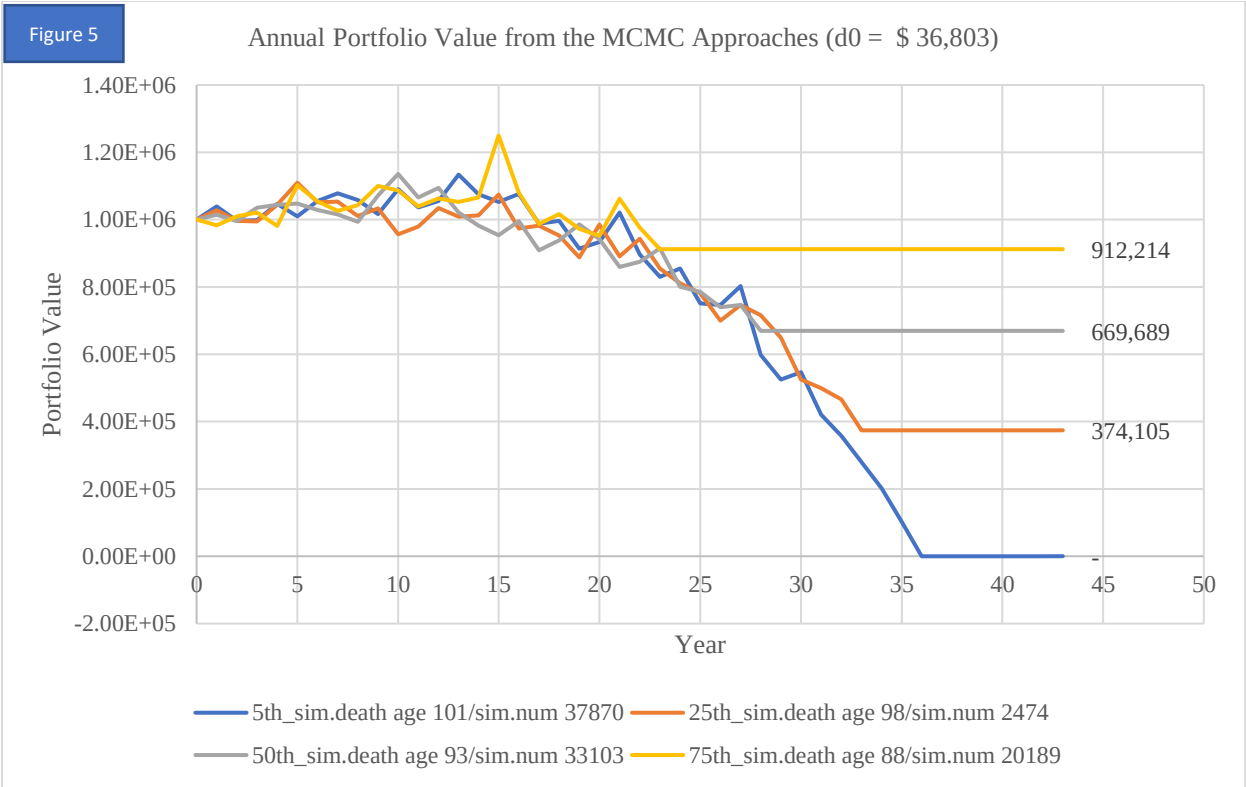
So, the death age in this simulation is

65 + 22 + 11 + 1 = 99

And the individual will become disabled from age 87 (65 + 22) to age 98.

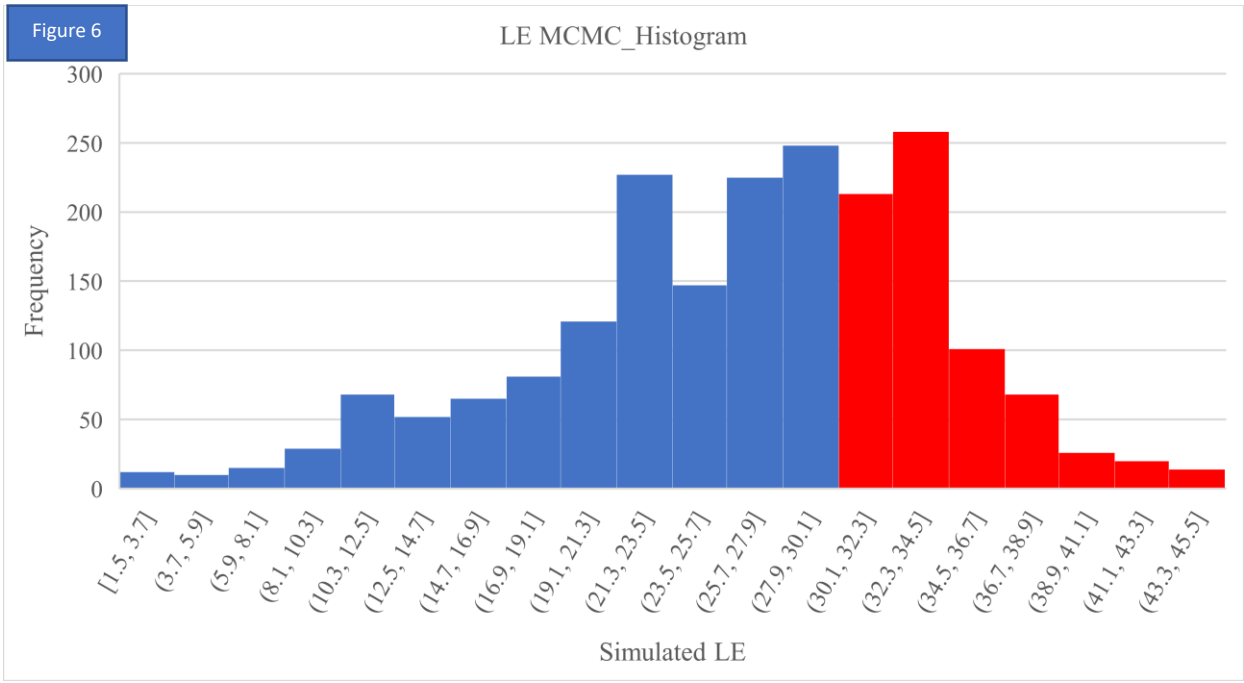
Consequently, according to the MCMC approaches, the client should choose $ 36,803 as the real optimal solution for the retirement financial planning strategy. The percentile lines of the annual portfolio value are shown in Figure 5.

Figure 5



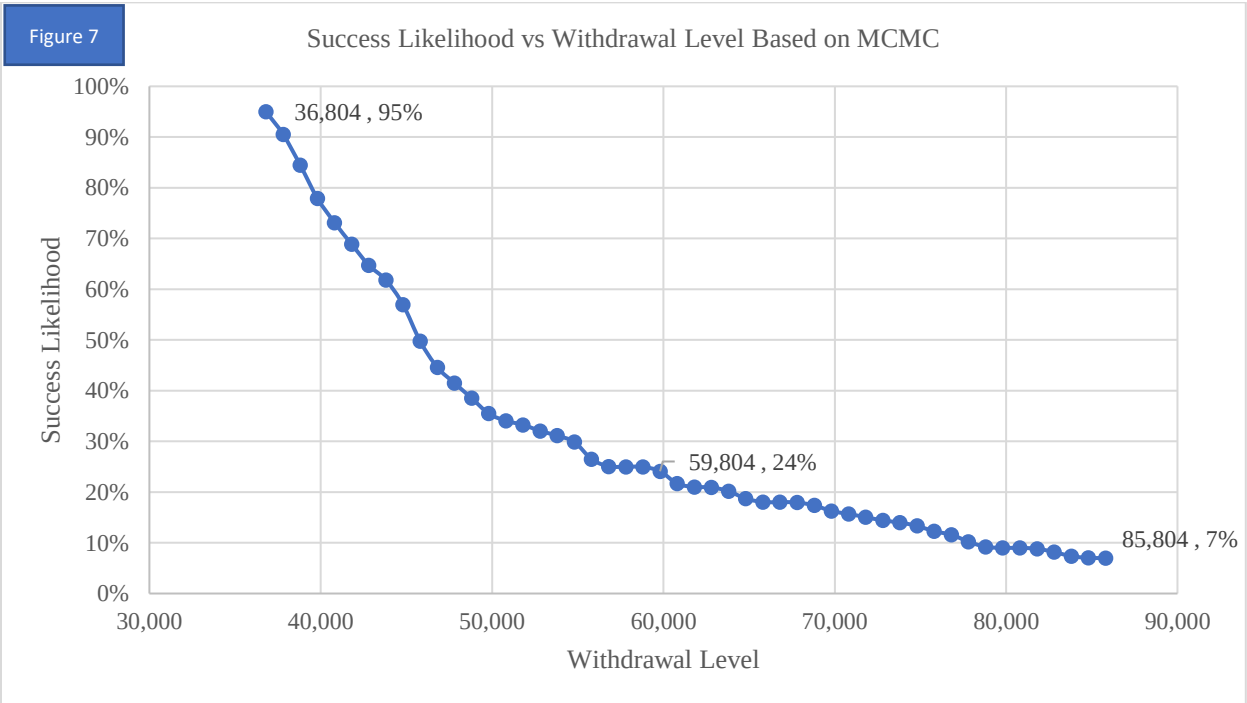
The reason that age 95 is not a safe assumption can be explained by the MCMC process of the future lifetime in Figure 6.

Figure 6



MCMC	Mean	SD
LE	26.44	7.80

According to the MCMC result of the life expectancy (LE) in Figure 6, the mean of LE is 26.44, the standard deviation of LE is 7.80 and the probability that LE is greater than 30 years (the red area) is 35%. That's why 95 is not a proper assumption of longevity risk in the financial plan modeling. The relationship between the withdraw level and the successful likelihood based on the MCMC process can be showed in Figure 7.



It shows that the success likelihood is decreasing while the withdrawal level is increasing. Moreover, the decreasing speed of success likelihood becomes slower when the withdraw level becomes larger.

4.2.2 Comparison of the Runtime

Besides the accuracy of this new methodology of the retirement financial planning, there is also a huge improvement in runtime by using R programming. Previously, the most difficult part for applying the MCMC approaches into the real business world is that the traditional method (Trial and Error or Goal Seek) usually takes an incredibly long time to get the final solution. In fact, sometimes, it will take several hours, days, even weeks to get the optimal asset allocation and the withdrawal level. It is definitely unacceptable for a business application.

However, the formulaic methodology can perfectly solve this problem. In Table 6, the comparison of the runtimes under different methods is shown. It is another contribution from this new financial planning model.

Table 6

Comparison of Runtime

	Fixed Longevity Target		MCMC Approach	
Num. of Iterations	1,000		1,000,000	
Runtime	Traditional Method	Formulating Method	Traditional Method	Formulating Method
	4 ~ 8 Hours	0.03 ~ 0.17 Sec	Several days or weeks	10.16 ~ 273.23 Sec

4.2.3 Required Initial Assets under Retirement Spending Goal

From the basic inputs in Table 3, the client's retirement spending goal is $ 70,000. However, the optimal solution under the MCMC approach is $ 36,803, and with the annual social security income of $ 20,000, the optimal retirement spending is $ 56,803. By using this formulaic methodology, the required initial assets can be also be solved quickly and precisely. The result is shown in Table 7.

Table 7

Required Initial Assets from MCMC Approaches

Retirement Spending Goal	Desired Success Rate			
	95%	90%	85%	80%
70,000	1,315,017	1,281,866	1,258,544	1,237,305
75,000	1,434,293	1,398,367	1,373,034	1,350,072
80,000	1,553,595	1,514,808	1,487,460	1,462,879
85,000	1,672,947	1,631,321	1,601,978	1,575,592

4.3 Example with Threshold of Portfolio Value and Bequest Needs

If the client has specific requirements on the threshold of the portfolio value during the retirement stage and the bequest need at death (Table 8), this new methodology can also get the accurate optimal solution and related information very quickly.

In Table 8, we can see that for this client, the threshold of portfolio value is $ 20,000 and the bequest needs is $ 50,000. In other words, the optimal withdrawal level should satisfy the constraint that the annual portfolio value cannot be below $ 20,000 during the whole retirement period. Meanwhile, the portfolio value should be at least $ 50,000 at the individual's death.

Table 8

Current Client Details

Current Age	65	Gender	F
Current Assets	1,000,000	Smoke Status	NS
Ret. Spending Goal	70,000	Desired Success Rate	95%
Port. Threshold	20,000	Bequest Needs	50,000

Table 9

Optimal Solution of Retirement Financial Planning Based on MCMC Approaches_1,000,000 Iterations

	With Thd. & Beq.Nd.	Without Thd. & Beq.Nd.
Opt. Asset Allocation	A_0%, B_87%, C_13%	A_0%, B_87%, C_13%
d_0^{Opt}	36,436	36,804
Runtime	15.38 sec	10.16 sec
Required Initial Assets	1,323,686	1,315,017
Sim.Num. of the 5th Percentile line of Port.Value	# 37805	# 37870

In Table 9, the d_0^{Opt} is decreased from \$ 36,804 to \$ 36,436. The runtime is increased a little bit because under the different threshold and bequest needs, more annual portfolio values will be checked in the algorithm in order to make sure that the constraints will be always satisfied during the whole retirement period. Obviously, more initial assets are required so that the retirement spending goal will be achieved. The percentile lines of the annual portfolio value are shown in Figure 8.

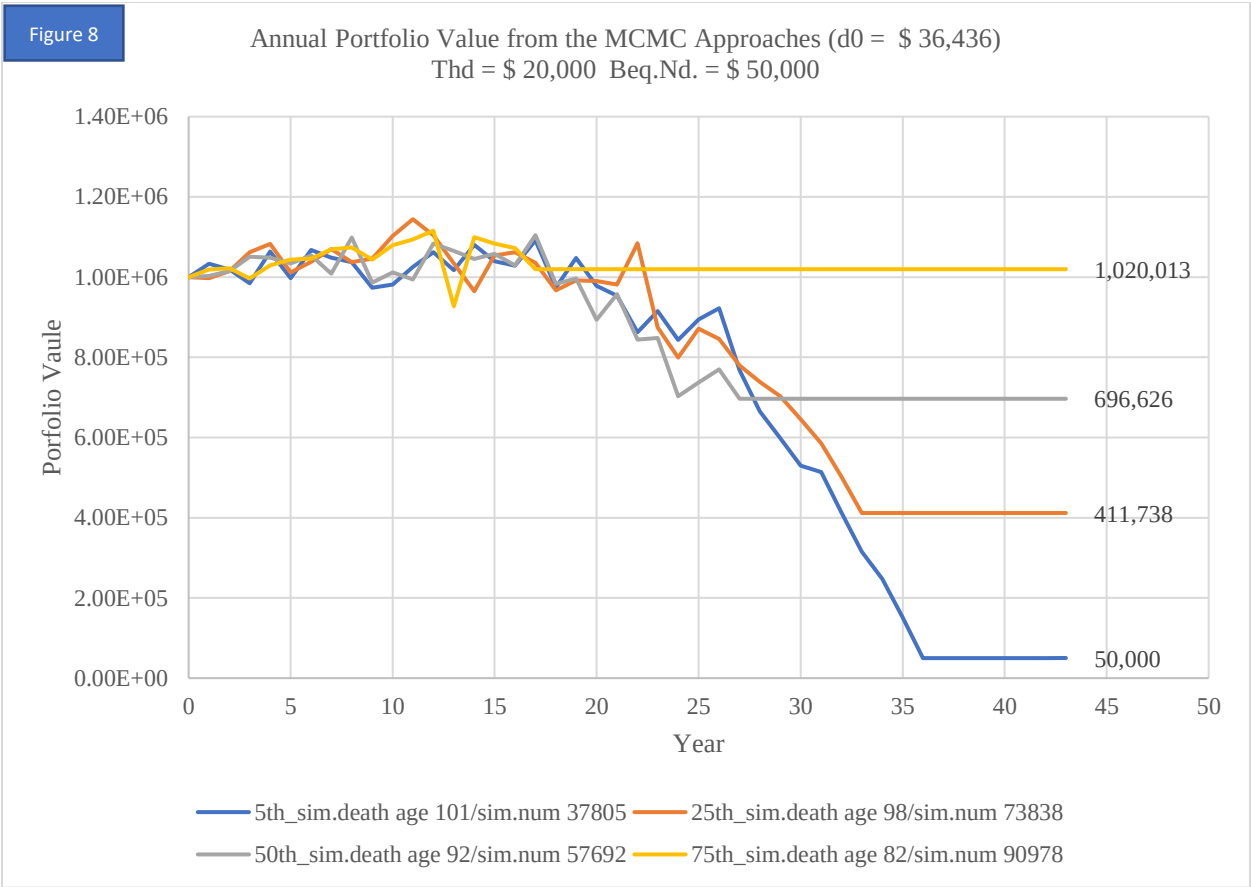


Figure 8 indicates that the 5th percentile line of the annual portfolio value corresponds to the 37805th simulation. The death age is 101 and the portfolio value at death is exactly $ 50,000 (bequest needs). Additionally, the annual portfolio value is always above the threshold $ 20,000 during the retirement period.

4.4 Example with SPIA and DIA

Another advantage of this methodology is that insurance and financial products can be easily be incorporated into the modeling. The impact of the cashflows (inflow or outflow) generated by insurance and financial products can be reflected in the vector A_t (section 3.1.2).

For example, in the general case (section 4.1), besides the annual pension income $ 20,000, this individual also bought a SPIA (single premium immediate lifetime annuity) and a DIA (deferred income annuity) at the current age 65. The annual SPIA income is $ 10,000. The deferred period of DIA is the healthy life expectancy[5] (HLE = 20[6]) of this individual and the annual DIA income is $ 30, 000. The payout ratio[7] of SPIA is 0.06 and the payout ratio of DIA is 0.36. So, the net current asset is 1,000,000 – 10,000 / 0.06 – 30,000 / 0.36 = 763,889.

So, in the formulating process, we will have

$$b_0 = 763,889$$

$$A_t = \left[\underbrace{10,000 \quad \cdots \quad 10,000}_{20} \quad \underbrace{40,000 \quad \cdots \quad 40,000}_{\text{the maximum age in mortality table - 20}} \right]$$

The comparison between the annual retirement spending (with or without SPIA and DIA) is shown in Table 10.

[5] Jeyaraj Vadiveloo. “Our calculator will guess how many healthy years of life you have left”. *The Conversation* Web. October 16. 2017.
“http://theconversation.com/our-calculator-will-guess-how-many-healthy-years-of-life-you-have-left-84498”
[6] Healthy Life Expectancy Calculator developed by the Goldenson Center at the University of Connecticut “https://apps.goldensoncenter.uconn.edu/HLEC/”
[7] Payout ratio = annual income / price
The two payout ratios in the section are based on “Annuity 2000 Basic Table – Female” and the interest rate is assumed to be 2.5%.

Table 10

Optimal Solution of Retirement Financial Planning Based on MCMC Approaches_1,000,000 Iterations

	With SPIA & DIA	Without SPIA or DIA
Opt. Asset Allocation	A_0%, B_87%, C_13%	A_0%, B_87%, C_13%
d_0^{Opt}	30,000	36,804
Initial living incomes	60,000	56,804
Runtime	15.24 sec	10.16 sec
Required Initial Assets	1,250,751	1,315,017
Sim.Num. of the 5th Percentile line of Port.Value	# 37585	# 37870

We can see that with SPIA and DIA, the optimal initial withdrawal level is decreased from 36,804 to 30,000. However, if we consider the SPIA income and pension, the total initial living income is increased to 60,000 (30,000 + 10,000 + 20,000) from 56,804 (36,804 + 20,000). And the required initial assets number is also decreased. So, this financial plan strategy (with SPIA and DIA) can improve the individual's financial position during retirement stage based on the assumed SPIA and DIA purchase rates.

Figure 9 Annual Portfolio Value from the MCMC Approaches (d0 = $ 30,000) with SPIA & DIA

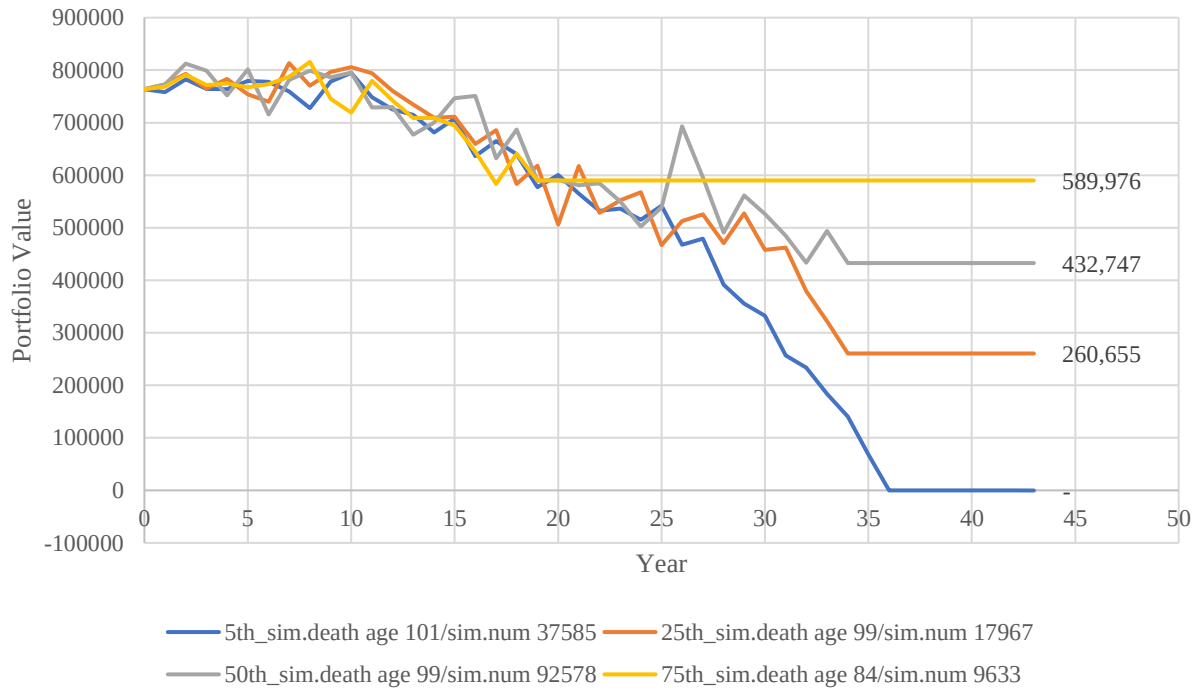


Figure 9 shows that the 5th percentile line of the annual portfolio value corresponds to the 37585th simulation. The death age is 101 and the portfolio value at death is exactly zero. Additionally, the annual portfolio value is always above zero during the retirement period.

More financial or insurance products can be incorporated into the modelling by adjusting the vector A_t and the initial current asset b_0.

Chapter 5 Conclusions and Practical Application

Here are a few key observations and conclusions:

1. The MCMC financial planning model creates two methodologies which can be used to optimize the maximum level of annual spending based on a specific level of ruin or quantile using a stochastic process based on the MCMC approach:

a) The formulaic methodology of the MCMC process

By using this method, the closed-form solution of the optimal quantile is derived. Moreover, through the R application, the runtime is dramatically decreased.

b) Non-linear optimization modelling of the asset allocation

Based on the formulaic methodology of the MCMC process, the objective function has been built. By mapping from the portfolio volatility to an individual's risk tolerance (target ruin/desired success rate/success likelihood), the constraints of the optimization problem have been built.

2. Under an individual's specific risk tolerance, requirement of the threshold of the portfolio value and bequest needs, the MCMC financial planning model is a very powerful tool which can supply the following information immediately:

a) The optimal asset allocation in a retirement portfolio

b) The maximum annual withdrawal level (the optimal quantile in a stochastic process)

c) The maximum annual withdrawal under different probabilities of ruin.

d) The required initial retirement assets based on the retirement spending goal

e) The success likelihood (or ruin probability) given an initial withdrawal level

3. The MCMC financial planning model shows the popular longevity assumption ensuring sufficient spending until age 95 is improper and very risky. According to the results in Chapter 4, the ruin probability will be increased from 5% to 32.8% if the maximum withdrawal level under the fixed longevity target 95 is used. It means that this assumption will result in a very high risk of the shortage of assets during the retirement period.

4. The longevity risk is very critical in the retirement financial planning model. The result in Figure 6 shows that the unexpected longevity is the main driven factor which will lead to ruin during the retirement stage.

5. The R programming application plays a very important role in this MCMC financial plan modelling.

Previously, due to the limited calculation capability, for example Excel, the MCMC approaches cannot be efficiently used in industry since the number of iterations to come up with the optimal solution results in an unacceptable runtime issue. However, by using the vector/matrix calculation in R and the functional programming in R, there is a huge improvement in runtime (Section 4.2.2), which makes the practical application of the MCMC approaches feasible in the financial planning world.

To summarize, in contrast to the traditional methodology of solving the optimal withdrawal level from a retirement portfolio, this research has created a new formulaic method based on the MCMC process to obtain a closed-form solution of the maximum withdrawal under a pre-determined asset allocation while satisfying an individual's risk tolerance. Based on this methodology, this research has also built a non-linear optimization model using an adjusted

Lagrange method to solve the optimal allocation of each asset in a portfolio, which can result in the real optimal withdrawal level for the retirement financial plan strategy. Furthermore, by using the R programming tool, the runtime of getting the optimal solution is dramatically decreased.

Furthermore, according to the flexibility of this new methodology, the healthy life expectancy (HLE) and unhealthy life expectancy (ULE) can be also incorporated into this financial plan model. Just like the example in section 4.4, an individual can use HLE to determine the length of deferred period of DIA and use ULE to determine how many years the DIA income should last for the future. By using this new methodology, we can analyze and quantify the impact of the combined investment strategy (with annuities and other insurance or finance products) in order to find out the optimal strategy to maximize the an individual's retirement life style.

References

Steven E. Shreve 2004**.** *Stochastic Calculus for Finance II: Continuous-Time Models.* Springer-Verlag New York.

Nicholas J. Higham. 2002. "Computing the Nearest Correlation Matrix—A Problem from Finance", IMA J. Numer. Anal. 22, 329–343, 2002.

J. S. Dagpunar. 2007. *Simulation and Monte Carlo: With applications in finance and MCMC.* Publisher: Wiley; 1 edition (March 12, 2007)

Manfred Gilli, Dietmar Maringer, Enrico Schumann. 2011. *Numerical Methods and Optimization in Finance (1st Edition)*. Missouri City, TX: Academic Press

Bengen, William P. 1994. "Determine Withdrawal Rates Using Historical Data." *Journal of Financial Planning* 17 (3): 172 - 180

Blanchett, David. 2014. "Exploring the Retirement Consumption puzzle." *Journal of Financial Planning* 27 (5): 34 - 42

Klinger, William J. 2007. "Using Decision Rules to Create Retirement Withdrawal Profiles." *Journal of Financial Planning* 20 (8): 60 - 67

Frank, Larry R., John B. Mitchell, and David M. Blanchett. 2011. "Probability-of- Failure-Based Decision Rules to Manage Sequence Risk in Retirement." *Journal of Financial Planning* 24 (11): 44 - 53

Arrow, K. J. and Hurwicz, L. 1972. "An Optimality Criterion for Decision-Making under Ignorance", in Carter,C. F. and Ford, J. L. (eds) *Uncertainty and Expectations in Economics*. Fairfield, NJ: Augustus M. Kelley

Artzner, P., Delbaen, F., Eber, J.M. and Heath, D. 1999. “Coherent Measures of Risk” *Mathematical Finance* 9 (3) 203–228.

Chambers, C. P. 2007. “Ordinal Aggregation and Quantiles” *Journal of Economic Theory*, 137, 416 - 431.

O’Neill, B. 2001. “Risk Aversion in International Relations Theory” *International Studies Quarterly*, 45, 617 - 640

Segal, U. and Sobel, J. 2002. “Min, Max, and Sum” *Journal of Economic Theory* 106 (1): 126–150

Cohen, M. 1992. “Security Level, Potential Level, Expected Utility: A Three-criterion Decision Model under Risk” *Theory and Decision*, 33, 101 - 134

Cotton, Cary., Alex Mears, and Dirk Cotton. 2016. “Competing Risks: Death and Ruin.” *Journal of Financial Planning* 15 (2): 34 - 40

Suarez, E. Dante, Antonio Suarez, Daniel T. Waltz. 2015. “The Perfect Withdraw Amount: A Methodology for Creating Retirement Account Distribution Strategies.” *Financial Services Review* 24 (4): 331 - 358

Rostek, M. 2006. “Quantile Maximization in Decision Theory,” PhD Dissertation, Ch. 1, Yale University

Savage, L. J. 1954. *The Foundations of Statistics*. New York: Wiley, revised and enlarged, New York: Dover, 1972

Appendix - R Programming

1. Simulation of Yield Curves

```
YieldCurveSimulation  <-  function (numObs, numIter, ass_, corr_, FileName) {
  setwd("E:\\PhD Graduation\\Model")
  # Readin Mean and SD
  assets  <-  readWorkbook("Yield Curve Simulation.xlsm", sheet = "Assets",
                           startRow = ass_, colNames = TRUE,
                           rowNames = FALSE, detectDates = FALSE,
                           skipEmptyRows = TRUE, skipEmptyCols = TRUE,
                           rows = NULL, cols = NULL,
                           check.names = FALSE, namedRegion = NULL,
                           na.strings = "NA", fillMergedCells = FALSE)
  # Readin Correlation Coefficient Matrix
  assets.corr  <-  unname(as.matrix(readWorkbook("Yield Curve Simulation.xlsm", sheet =
"Correlation Coefficient",
                                                 startRow = corr_, colNames = FALSE,
                                                 rowNames = FALSE, detectDates = FALSE,
                                                 skipEmptyRows = TRUE, skipEmptyCols = TRUE,
```

```
                                    rows = c(9 : 27), cols = c(4 : 22),
                                    check.names = FALSE, namedRegion = NULL,
                                    na.strings = "NA", fillMergedCells = FALSE)))
  numAssets  <-  nrow(assets.corr)
  # Create random number 3-D array
  priceRet  <-  array(0, dim = c(numObs + 1,numAssets,numIter))
  priceRet[1, , ]  <-  1
  randRet   <<-  array(0, dim = c(numObs,numAssets,numIter))
  randNum   <-  array(rnorm(n = numObs * numAssets * numIter, mean = 0, sd = 1), dim =
c(numObs,numAssets,numIter))
  # Get the covariance matrix of assets, which will be definitely a positive definite matrix by
using 'nearPD'
  assets.cov  <-  nearPD(cor2cov(assets.corr, assets$Sd))
  saveRDS(assets.cov, "E:\\PhD Graduation\\Model\\assetsCov.rds")
  # Cholesky Decomposition
  cholM  <-  chol(assets.cov)
  # Geometric Brownian motion simulation
  time.Intv  <-  matrix(rep(1 : numObs, numAssets), ncol = numAssets) # T - t
  for (i in 1 : numIter) {
```

```
  corr.wt  <-  randNum[ , , i] %*% cholM

  priceRet[2 :  dim(priceRet)[1], , i]  <-  t(exp((assets$Mean - 0.5 * assets$Sd ^ 2) * t(time.Intv)

                                  + assets$Sd * (t(time.Intv)) ^ 0.5 * t(corr.wt)))

  randRet[ , , i]  <<-  diff(priceRet[ , , i]) / priceRet[1 : numObs, , i]

 }

 saveRDS(randRet, FileName)

}
```

2. Simulation of Health Status

```
HealthConditionSimulation  <-  function (numIter, current.age, gender, sm.ind, drn.fac.qh,
drn.fac.qd, drn.fac.i, FileName) {

  setwd("E:\\PhD Graduation\\Model")

  # Loadin mortality and incidence rates tables ++++++++++++++++++++++

  qxh.MNS  <-  unname(as.matrix(readWorkbook("Health Condition Simulation.xlsm", sheet =
"q(x, h)",
                                startRow = 3, colNames = FALSE,
                                rowNames = FALSE, detectDates = FALSE,
                                skipEmptyRows = TRUE, skipEmptyCols = TRUE,
                                rows = NULL, cols = c(3 : 110),
                                check.names = FALSE, namedRegion = NULL,
                                na.strings = "NA", fillMergedCells = FALSE)))

  qxh.MNS[is.na(qxh.MNS)]  <-  1

  qxh.MS   <-  unname(as.matrix(readWorkbook("Health Condition Simulation.xlsm", sheet =
"q(x, h)",
                                startRow = 3, colNames = FALSE,
                                rowNames = FALSE, detectDates = FALSE,
                                skipEmptyRows = TRUE, skipEmptyCols = TRUE,
```

```
                                rows = NULL, cols = c(113 : 220),

                                check.names = FALSE, namedRegion = NULL,

                                na.strings = "NA", fillMergedCells = FALSE)))

  qxh.MS[is.na(qxh.MS)]  <-  1

  qxh.FNS  <-  unname(as.matrix(readWorkbook("Health Condition Simulation.xlsm", sheet =
"q(x, h)",

                                startRow = 3, colNames = FALSE,

                                rowNames = FALSE, detectDates = FALSE,

                                skipEmptyRows = TRUE, skipEmptyCols = TRUE,

                                rows = NULL, cols = c(223 : 330),

                                check.names = FALSE, namedRegion = NULL,

                                na.strings = "NA", fillMergedCells = FALSE)))

  qxh.FNS[is.na(qxh.FNS)]  <-  1

  qxh.FS  <-  unname(as.matrix(readWorkbook("Health Condition Simulation.xlsm", sheet =
"q(x, h)",

                               startRow = 3, colNames = FALSE,

                               rowNames = FALSE, detectDates = FALSE,

                               skipEmptyRows = TRUE, skipEmptyCols = TRUE,

                               rows = NULL, cols = c(333 : 440),
```

```
                                check.names = FALSE, namedRegion = NULL,

                                na.strings = "NA", fillMergedCells = FALSE)))

  qxh.FS[is.na(qxh.FS)] <- 1

  qxd.M  <-  unname(as.matrix(readWorkbook("Health Condition Simulation.xlsm", sheet =
"q(x, d)",

                                startRow = 3, colNames = FALSE,

                                rowNames = FALSE, detectDates = FALSE,

                                skipEmptyRows = TRUE, skipEmptyCols = TRUE,

                                rows = NULL, cols = c(3 : 110),

                                check.names = FALSE, namedRegion = NULL,

                                na.strings = "NA", fillMergedCells = FALSE)))

  qxd.M[is.na(qxd.M)]  <-  1

  qxd.F  <-  unname(as.matrix(readWorkbook("Health Condition Simulation.xlsm", sheet = "q(x,
d)",

                                startRow = 3, colNames = FALSE,

                                rowNames = FALSE, detectDates = FALSE,

                                skipEmptyRows = TRUE, skipEmptyCols = TRUE,

                                rows = NULL, cols = c(113 : 220),

                                check.names = FALSE, namedRegion = NULL,
```

```
                    na.strings = "NA", fillMergedCells = FALSE)))

qxd.F[is.na(qxd.F)] <- 1

ix.M <- unname(as.matrix(readWorkbook("Health Condition Simulation.xlsm", sheet = "i(x)",

                    startRow = 3, colNames = FALSE,

                    rowNames = FALSE, detectDates = FALSE,

                    skipEmptyRows = TRUE, skipEmptyCols = TRUE,

                    rows = NULL, cols = c(3 : 110),

                    check.names = FALSE, namedRegion = NULL,

                    na.strings = "NA", fillMergedCells = FALSE)))

ix.M[is.na(ix.M)] <- 1

ix.F <- unname(as.matrix(readWorkbook("Health Condition Simulation.xlsm", sheet = "i(x)",

                    startRow = 3, colNames = FALSE,

                    rowNames = FALSE, detectDates = FALSE,

                    skipEmptyRows = TRUE, skipEmptyCols = TRUE,

                    rows = NULL, cols = c(113 : 220),

                    check.names = FALSE, namedRegion = NULL,

                    na.strings = "NA", fillMergedCells = FALSE)))

ix.F[is.na(ix.F)] <- 1
```

```
# Generate random numbers ++++++++++++++++++++++

#col  <-  20000

#row  <-  nrow(ix.F) + 1

#num.rand  <-  col * row

#mat.rand  <-  matrix(runif(num.rand, min = 0, max = 1), ncol = col, nrow = row)

#saveRDS(mat.rand, "RandomNumbers.rds")



mat.rand  <-  readRDS("RandomNumbers.rds")

# MCMC Simulation ++++++++++++++++++++++++++++++

# 1. Select mortality & incidence rates tables

if (gender == "F" & sm.ind == "NS") {

  qxh <- 1 - (1 - qxh.FNS[ , current.age + 1]) ^ drn.fac.qh

  qxd <- 1 - (1 - qxd.F[ , current.age + 1]) ^ drn.fac.qd

  ix <- 1 - (1 - ix.F[ , current.age + 1]) ^ drn.fac.i

} else if (gender == "M" & sm.ind == "NS") {

  qxh <- 1 - (1 - qxh.MNS[ , current.age + 1]) ^ drn.fac.qh

  qxd <- 1 - (1 - qxd.M[ , current.age + 1]) ^ drn.fac.qd

  ix <- 1 - (1 - ix.M[ , current.age + 1]) ^ drn.fac.i
```

```
} else if (gender == "M" & sm.ind == "S") {

  qxh <- 1 - (1 - qxh.MS[ , current.age + 1]) ^ drn.fac.qh

  qxd <- 1 - (1 - qxd.M[ , current.age + 1]) ^ drn.fac.qd

  ix <- 1 - (1 - ix.M[ , current.age + 1]) ^ drn.fac.i

} else {

  qxh <- 1 - (1 - qxh.FS[ , current.age + 1]) ^ drn.fac.qh

  qxd <- 1 - (1 - qxd.F[ , current.age + 1]) ^ drn.fac.qd

  ix <- 1 - (1 - ix.F[ , current.age + 1]) ^ drn.fac.i

}

# 2. MCMC process

# 2.1 Generate MC status matrix

mat.mcmc <<- matrix(0, nrow = length(ix) + 1 - current.age, ncol = numIter)

mat.mcmc[1, ] <<- 1 # Assume that individuals start with healthy

mat.mcmc[nrow(mat.mcmc), ] <<- 3 # everyone will die in the end

# 2.2 Simulate health status

for (i in 1 : numIter) {


  for (j in 2 : (nrow(mat.mcmc) - 1)) {
```

```
    if (mat.mcmc[j - 1, i] == 1) {
      # If starting with healthy
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
      if (mat.rand[j - 1, 2 * i - 1] > ix[j - 1] & mat.rand[j - 1, 2 * i] > qxh[j - 1]) {
        mat.mcmc[j, i] <<- 1
      } else if (mat.rand[j - 1, 2 * i - 1] <= ix[j - 1] & mat.rand[j - 1, 2 * i] > qxd[j - 1]) {
        mat.mcmc[j, i] <<- 2
      } else {
        mat.mcmc[j, i] <<- 3
      }
      #
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
++++++++++++++++++++
    } else if (mat.mcmc[j - 1, i] == 2) {
      # If starting with disabled
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
      if (mat.rand[j - 1, 2 * i] > qxd[j - 1]) {
        mat.mcmc[j, i] <<- 2
      } else {
```

```
        mat.mcmc[j, i] <<- 3

      }

      #
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
++++++++++++++++++++
    } else {

      # If starting with death
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

      mat.mcmc[j, i] <<- 3

    }

    #
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
++++++++++++++++++++
  } # end of j

 } # end of i

 saveRDS(mat.mcmc, "HealthConditionSimulation.rds")
```

```
# LE


le <- (length(which(mat.mcmc != 3)) - 1) / numIter + 0.5

mat.mcmc.le <- mat.mcmc

mat.mcmc.le[which(mat.mcmc.le != 3)] <- 1

mat.mcmc.le[which(mat.mcmc.le == 3)] <- 0

vec.le <- apply(mat.mcmc.le, 2, sum) + 0.5


if (drn.fac.qh == drn.fac.qd & drn.fac.qd == drn.fac.i & drn.fac.i == 1) {

  saveRDS(vec.le, "vector_LE_Benc.rds")

} else {

  saveRDS(vec.le, "vector_LE.rds")

}


# HLE


hle <- (length(which(mat.mcmc == 1)) - 1) / numIter + 0.5

mat.mcmc.hle <- mat.mcmc
```

```
mat.mcmc.hle[which(mat.mcmc.hle != 1)] <- 0

vec.hle <- apply(mat.mcmc.hle, 2, sum) + 0.5



if (drn.fac.qh == drn.fac.qd & drn.fac.qd == drn.fac.i & drn.fac.i == 1) {

  saveRDS(vec.hle, "vector_HLE_Benc.rds")

} else {

  saveRDS(vec.hle, "vector_HLE.rds")

}



# DLE



dle <- (length(which(mat.mcmc == 2))) / numIter

vec.dle <- vec.le - vec.hle



if (drn.fac.qh == drn.fac.qd & drn.fac.qd == drn.fac.i & drn.fac.i == 1) {

  saveRDS(vec.dle, "vector_DLE_Benc.rds")

} else {

  saveRDS(vec.dle, "vector_DLE.rds")
```

```
}

# Closed-form of the LE, HLE and DLE
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

kpx_h  <-  rep(1, (dim(qxh.MNS)[1] - current.age)) # kpx_h[1] = 0_px_h = 1

for (k in 1 : (length(kpx_h) - 1)) {

  kpx_h[k + 1]  <-  kpx_h[k] * (1 - ix[k]) * (1 - qxh[k])

}


kpx_d  <-  rep(1, (dim(qxh.MNS)[1] - current.age)) # kpx_d[1] = 0_px_d = 1

for (k in 1 : (length(kpx_d) - 1)) {

  kpx_d[k + 1]  <-  kpx_d[k] * (1 - qxd[k])

}


qxd  <-  qxd[1 : (dim(qxh.MNS)[1] - current.age)]

pxd  <-  1 - qxd


k_slash_qxt_d  <-  matrix(1,

                          nrow = (dim(qxh.MNS)[1] - current.age),
```

```
                    ncol = (dim(qxh.MNS)[1] - current.age)) # The structure of k_slash_qxt_d: The
structure of k_slash_qxt_d in HLEcalculator_CF.png
  k_slash_qxt_d[1, ]  <-  qxd
  for (j in 1 : ncol(k_slash_qxt_d)) {
    for(i in 2 : nrow(k_slash_qxt_d)) {
      k_slash_qxt_d[i, j]  <-  cumprod(pxd[j : ncol(k_slash_qxt_d)])[i - 1] * qxd[i + j - 1]
    }
  }
  k_slash_qxt_d[is.na(k_slash_qxt_d)]  <-  0

  k_slash_qxt_tau  <-  rep(1, (dim(qxh.MNS)[1] - current.age)) #  k_slash_qxt_tau[1] = qx+0_tau
  k_slash_qxt_tau[1]  <-  (1 - ix[1]) * qxh[1] + ix[1] * qxd[1]

  for (k in 1 : (length(k_slash_qxt_tau) - 1)) {
    sum_fac  <-  0
    for (t in 0 : k) {
      sum_fac  <-  sum_fac + kpx_h[t + 1] * ix[t + 1] * k_slash_qxt_d[k + 1 - t, t + 1]
    }
    k_slash_qxt_tau[k + 1]  <-  kpx_h[k + 1] * (1 - ix[k + 1]) * qxh[k + 1] + sum_fac
```

```
}

LE  <-  0
for (t in 1 : (length(k_slash_qxt_tau) - 1)) {
  LE  <-  LE + t * k_slash_qxt_tau[t + 1]
}
LE_Comp  <-  LE + 0.5

HLE  <-  0
for (t in 1 : (length(kpx_h) - 1)) {
  HLE  <-  HLE + t * kpx_h[t + 1] * (ix[t + 1] + (1 - ix[t + 1]) * qxh[t + 1])
}

HLE_Comp  <-  HLE + 0.5

DLE_Comp  <-  LE_Comp - HLE_Comp
# Closed-form of the LE, HLE and DLE
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++End
```

```
 HLE_result  <<-  list(LE    = le,

                       LE_CF = LE_Comp,

                       sdLE  = sd(vec.le),

                       HLE   = hle,

                       HLE_CF= HLE_Comp,

                       sdHLE = sd(vec.hle),

                       DLE   = dle,

                       DLE_CF= DLE_Comp,

                       sdDLE = sd(vec.dle))


  saveRDS(HLE_result, "HLE_result.rds")


}
```

3. Closed-Form Solution of the Maximum Withdrawal under MCMC

```
MaxRI_MCMC_TB  <-  function () {

 # Execute "Config_Readin.R" firstly


 setwd("E:\\PhD Graduation\\Model")


 # Current Client Details, Inflation & Spending Assumptions ++++++++++++++++++++++

 setup  <-  readWorkbook("FPM-v0.xlsm", sheet = "Setup", startRow = 8,
                         colNames = FALSE, rowNames = FALSE, detectDates = FALSE,
                         skipEmptyRows = TRUE, skipEmptyCols = TRUE,
                         cols = c(5 : 15),
                         check.names = FALSE, namedRegion = NULL,
                         na.strings = "NA", fillMergedCells = FALSE)


 curr_age       <-  as.numeric(setup[1, 2])
 gender         <-  setup[3, 2]
 curr_assets    <-  as.numeric(setup[4, 2])
 long_targ_age  <-  as.numeric(setup[1, 4])
```

```
  desi_succ_rate  <-  as.numeric(setup[2, 4])

  smoke_status    <-  setup[3, 4]

  ret_spen_goal   <-  as.numeric(setup[4, 4])

  infl_rate       <-  as.numeric(setup[1, 6])

  taper           <-  as.numeric(setup[1, 8])

  infl_incr       <-  as.numeric(setup[2, 6])

  iter_num_yc     <-  as.numeric(setup[2, 8])

  taper_start_age <-  as.numeric(setup[3, 6])

  taper_end_age   <-  as.numeric(setup[3, 8])

  iter_num_hc     <-  as.numeric(setup[4, 8])

  port_thd        <-  as.numeric(setup[5, 2])

  bequest         <-  as.numeric(setup[5, 4])

  assets_select   <-  eval(parse(text = setup[5, 6]))

  pre_allo        <-  t(matrix(rep(eval(parse(text = setup[5, 8])), dim(randRetOrig)[1]), ncol =
dim(randRetOrig)[1]))


  randRet  <-  randRetOrig[ , assets_select, ]

  # Current Client Details, Inflation & Spending Assumptions ++++++++++++++++++End
```

```
  # MCMC Selection
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
  mcmc  <-  get(paste0("MCMC", "_", curr_age, "_", gender, "_", smoke_status))


  # Remove unused variables
  #varlist  <-  ls()
  #uselessVarList  <-  grep("MCMC*", varlist, value = FALSE)
  #rm(list = varlist[uselessVarList])


  # For one scenario of health conditions, we need to do 1000 simulations of yields
  #iter_num_hc  <-  500 # The iteration number of MCMC of health condition
  mcmc  <-  mcmc[-1, 1 : iter_num_hc] # iteration number of health condition simulations is set to 500, t in mcmc starts from 1 not 0
  mcmc  <-  mcmc[ , rep(1 : ncol(mcmc), rep(iter_num_yc, ncol(mcmc)))] # each column in mcmc will be duplicated for iter_num_yc times


  # Indentify death time and disability period
  index.death  <-  which(mcmc == 3)
  index.disability  <-  which(mcmc == 2)
```

```
  # Number of rows of mcmc

  row.mcmc  <-  nrow(mcmc)



  rm("mcmc")

  # MCMC Selection
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++End



  # Post-retirement Pre-allocation of assets ++++++++++++++++++++++++++++++++++++

  #pre_allo  <-  unname(as.matrix(readWorkbook("FPM-v0.xlsm", sheet = "Port_Pre-Allo",
startRow = 9,

  #                                colNames = FALSE, rowNames = FALSE, detectDates = FALSE,

  #                                skipEmptyRows = TRUE, skipEmptyCols = TRUE,

  #                                rows = c(9 : 108), cols = c(6 : 24),

  #                                check.names = FALSE, namedRegion = NULL,

  #                                na.strings = "NA", fillMergedCells = FALSE)))

  # Post-retirement Pre-allocation of assets ++++++++++++++++++++++++++++++++++End
```

```
 # Income
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
+++
 income  <-  unname(as.matrix(readWorkbook("FPM-v0.xlsm", sheet = "Income", startRow = 9,
                              colNames = FALSE, rowNames = FALSE, detectDates = FALSE,
                              skipEmptyRows = TRUE, skipEmptyCols = TRUE,
                              rows = c(9 : 108), cols = c(5 : 9),
                              check.names = FALSE, namedRegion = NULL,
                              na.strings = "NA", fillMergedCells = FALSE)))
 inc.vec  <-  apply(income, 1, sum)
 inc.vec  <-  matrix(rep(inc.vec), nrow = row.mcmc, ncol = iter_num_yc * iter_num_hc)
 index.death.adj  <-  index.death[- which(index.death %in% c(1 : ncol(inc.vec) * row.mcmc))] +
1 # since inc.vec starts from t = 0
 inc.vec[index.death.adj]  <-  0
 # Income
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
End


 # Preparation for Calculation Factors
++++++++++++++++++++++++++++++++++++++++++++
```

```
  wr  <-  matrix(0, nrow = dim(randRet)[1], ncol = iter_num_yc)


  for (i in 1 : dim(wr)[2]) {

    wr[ , i]  <-  diag((1 + randRet[ , , i]) %*% t(pre_allo))

  }


  wr  <-  wr[1 : row.mcmc, ]

  wr  <-  matrix(rep(wr, iter_num_hc), nrow = nrow(wr))


  wr[index.death]  <-  1


  #raw_IT  <-  seq(from = infl_rate, by = infl_incr, length.out = row.mcmc)

  #raw_IT[(taper_start_age + 1 - curr_age) : (taper_end_age + 1 - curr_age)]  <-
raw_IT[(taper_start_age + 1 - curr_age) : (taper_end_age + 1 - curr_age)] + taper

  #cumu_IT  <-  cumprod(1 + raw_IT)


  cumu_I   <-  cumprod(seq(from = 1 + infl_rate, by = infl_incr, length.out = row.mcmc))

  cumu_IT  <-  matrix(rep(cumu_I, iter_num_hc * iter_num_yc), nrow = row.mcmc)

  cumu_IT[index.death]  <-  0
```

```
  cumu_IT[index.disability]  <-  cumu_IT[index.disability] * (1 + taper)


  # Preparation for Calculation Factors
++++++++++++++++++++++++++++++++++++++++++End


  # Calculation of the maximum retirement income
++++++++++++++++++++++++++++++++++

  n  <-  c(3 : dim(wr)[1]) # n starts from 3 (or b# starts from 3), otherwise there will be a
dimensional issue

  RI.Collection  <-   rep(0, ncol(wr))

  f1  <-  matrix(0, nrow = length(n), ncol = ncol(wr))

  f2  <-  matrix(0, nrow = length(n), ncol = ncol(wr))

  f3  <-  matrix(0, nrow = length(n), ncol = ncol(wr))


  for (i in 1 : length(n)) {


    # f1 ++++++++++++++++++++++++++++++++++++

    f1[i, ]   <-  colProds(wr[1 : n[i], ])

    # f1 ++++++++++++++++++++++++++++++++++End
```

```
# f2 ++++++++++++++++++++++++++++++++++++

IT      <-  cumu_IT[1 : (n[i] - 1), ]

wrTemp  <-  wr[2 : n[i], ][c(nrow(wr[2 : n[i], ]) : 1), ]

wrProd  <-  apply(wrTemp, 2, cumprod)

wrAdj   <-  wrProd[c(nrow(wr[2 : n[i], ]) : 1), ]

ITwr    <-  IT * wrAdj

f2[i, ]   <-  colSums(ITwr)

# f2 ++++++++++++++++++++++++++++++++++End


# f3 ++++++++++++++++++++++++++++++++++++

SPIC    <-  inc.vec[2 : n[i], ]

SPICwr  <-  SPIC * wrAdj

f3[i, ]   <-  colSums(SPICwr)

# f3 ++++++++++++++++++++++++++++++++++End


# Closed-form solution ++++++++++++++++++

d0_vec  <-  (curr_assets * f1[i, ] -
```

```
            inc.vec[1] * f2[i, ] + f3[i, ]) / (f1[i, ] + f2[i, ]) -
              port_thd / (f1[i, ] + f2[i, ])
  # Closed-form solution +++++++++++++++++End


  RI.Collection  <-  rbind(RI.Collection, d0_vec)


}


# At this moment i = length(n)
d0_vec_n  <-  (curr_assets * f1[i, ] -
            inc.vec[1] * f2[i, ] + f3[i, ]) / (f1[i, ] + f2[i, ]) -
              bequest / (f1[i, ] + f2[i, ])


RI.Collection  <-  as.matrix(RI.Collection[-1 , ])
RI.Collection[nrow(RI.Collection), ]  <-  d0_vec_n


min_RI.Collection.MCMC  <<- apply(RI.Collection, 2, min)
max.RI.MCMC            <<-  unname(quantile(min_RI.Collection.MCMC, 1 - desi_succ_rate))
```

```
  # Calculation of the maximum retirement income
++++++++++++++++++++++++++++++++End


  # Calculation of the annual account value
++++++++++++++++++++++++++++++++++++++++

  n3_acc_value  <-  (curr_assets - max.RI.MCMC) * f1 - (max.RI.MCMC + inc.vec[1, ]) * f2 +
f3 # the first row is n = 3

  n0_acc_value  <-  rep(curr_assets, iter_num_yc)

  n1_acc_value  <-  curr_assets * wr[1, ] - max.RI.MCMC * wr[1, ]

  n2_acc_value  <-  (curr_assets - max.RI.MCMC) * colProds(wr[1 : 2, ]) - (max.RI.MCMC +
inc.vec[1, ]) * cumu_IT[1] * wr[2, ] + inc.vec[2, ] * wr[2, ]

  ann_acc_value_MCMC <<-  unname(rbind(n0_acc_value, n1_acc_value, n2_acc_value,
n3_acc_value))

  low_acc_value_MCMC <<-  apply(ann_acc_value_MCMC, 2, min)

  sort_low_value_MCMC<<-  sort(low_acc_value_MCMC)

  index_low_val_MCMC <<-  match(sort_low_value_MCMC, low_acc_value_MCMC)


  # Check the b# (account value) by using recursive method +++++++++

  #iter_num  <-  1 # Randomly pick one simulation
```

```
  #

  #bn     <-  rep(0, nrow(ann_acc_value_MCMC))

  #bn[1]  <-  curr_assets

  #bn[2]  <-  (bn[1] - max.RI.MCMC)*wr[1, iter_num]

  #

  #for (j in 3 : nrow(ann_acc_value_MCMC)) {

  #  bn[j] <- (bn[j - 1] - ((max.RI.MCMC + inc.vec[1]) * cumu_IT[j - 2] - inc.vec[j - 1])) * wr[j -
1 , iter_num]

  #}

  #comp_result  <-  ann_acc_value_MCMC[ , iter_num] - bn

  # Check the b# (account value) by using recursive method ++++++End


  # Calculation of the annual account value
++++++++++++++++++++++++++++++++++++End


}
```

4. Ruin Calculator Given a Withdrawal Level under MCMC

```
RI_Ruin_IniAss_MCMC_TB  <-  function (Ret.Inc.MCMC) {

 # Execute "Config_Readin.R" firstly


 setwd("E:\\PhD Graduation\\Model")


 # Current Client Details, Inflation & Spending Assumptions ++++++++++++++++++++++
 setup  <-  readWorkbook("FPM-v0.xlsm", sheet = "Setup", startRow = 8,
                         colNames = FALSE, rowNames = FALSE, detectDates = FALSE,
                         skipEmptyRows = TRUE, skipEmptyCols = TRUE,
                         cols = c(5 : 15),
                         check.names = FALSE, namedRegion = NULL,
                         na.strings = "NA", fillMergedCells = FALSE)


 curr_age       <-  as.numeric(setup[1, 2])
 gender         <-  setup[3, 2]
 curr_assets    <-  as.numeric(setup[4, 2])
 long_targ_age  <-  as.numeric(setup[1, 4])
```

```
desi_succ_rate  <-  as.numeric(setup[2, 4])

smoke_status    <-  setup[3, 4]

ret_spen_goal   <-  as.numeric(setup[4, 4])

infl_rate       <-  as.numeric(setup[1, 6])

taper           <-  as.numeric(setup[1, 8])

infl_incr       <-  as.numeric(setup[2, 6])

iter_num_yc     <-  as.numeric(setup[2, 8])

taper_start_age <-  as.numeric(setup[3, 6])

taper_end_age   <-  as.numeric(setup[3, 8])

iter_num_hc     <-  as.numeric(setup[4, 8])

port_thd        <-  as.numeric(setup[5, 2])

bequest         <-  as.numeric(setup[5, 4])

assets_select   <-  eval(parse(text = setup[5, 6]))

pre_allo        <-  t(matrix(rep(eval(parse(text = setup[5, 8])), dim(randRetOrig)[1]), ncol =
dim(randRetOrig)[1]))


randRet  <-  randRetOrig[ , assets_select, ]

# Current Client Details, Inflation & Spending Assumptions ++++++++++++++++++End
```

```
  # MCMC Selection
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
  mcmc  <-  get(paste0("MCMC", "_", curr_age, "_", gender, "_", smoke_status))


  # Remove unused variables
  #varlist  <-  ls()
  #uselessVarList  <-  grep("MCMC*", varlist, value = FALSE)
  #rm(list = varlist[uselessVarList])


  # For one scenario of health conditions, we need to do 1000 simulations of yields
  #iter_num_hc  <-  500 # The iteration number of MCMC of health condition
  mcmc  <-  mcmc[-1, 1 : iter_num_hc] # iteration number of health condition simulations is set
to 500
  mcmc  <-  mcmc[ , rep(1 : ncol(mcmc), rep(iter_num_yc, ncol(mcmc)))] # each column in
mcmc will be duplicated for iter_num_yc times


  # Indentify death time and disability period
  index.death  <-  which(mcmc == 3)
  index.disability  <-  which(mcmc == 2)
```

```
  # Number of rows of mcmc

  row.mcmc  <-  nrow(mcmc)



  rm("mcmc")

  # MCMC Selection
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++End



  # Post-retirement Pre-allocation of assets ++++++++++++++++++++++++++++++++++++

  #pre_allo  <-  unname(as.matrix(readWorkbook("FPM-v0.xlsm", sheet = "Port_Pre-Allo",
startRow = 9,

  #                                colNames = FALSE, rowNames = FALSE, detectDates = FALSE,

  #                                skipEmptyRows = TRUE, skipEmptyCols = TRUE,

  #                                rows = c(9 : 108), cols = c(6 : 24),

  #                                check.names = FALSE, namedRegion = NULL,

  #                                na.strings = "NA", fillMergedCells = FALSE)))

  # Post-retirement Pre-allocation of assets ++++++++++++++++++++++++++++++++++End
```

```
 # Income
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
+++
 income  <-  unname(as.matrix(readWorkbook("FPM-v0.xlsm", sheet = "Income", startRow = 9,
                              colNames = FALSE, rowNames = FALSE, detectDates = FALSE,
                              skipEmptyRows = TRUE, skipEmptyCols = TRUE,
                              rows = c(9 : 108), cols = c(5 : 9),
                              check.names = FALSE, namedRegion = NULL,
                              na.strings = "NA", fillMergedCells = FALSE)))
 inc.vec  <-  apply(income, 1, sum)
 inc.vec  <-  matrix(rep(inc.vec), nrow = row.mcmc, ncol = iter_num_yc * iter_num_hc)
 index.death.adj  <-  index.death[- which(index.death %in% c(1 : ncol(inc.vec) * row.mcmc))] +
1
 inc.vec[index.death.adj]  <-  0
 # Income
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
End


 # Preparation for Calculation Factors
++++++++++++++++++++++++++++++++++++++++++
```

```
  wr  <-  matrix(0, nrow = dim(randRet)[1], ncol = iter_num_yc)


  for (i in 1 : dim(wr)[2]) {

    wr[ , i]  <-  diag((1 + randRet[ , , i]) %*% t(pre_allo))

  }


  wr  <-  wr[1 : row.mcmc, ]

  wr  <-  matrix(rep(wr, iter_num_hc), nrow = nrow(wr))


  wr[index.death]  <-  1


  #raw_IT  <-  seq(from = infl_rate, by = infl_incr, length.out = row.mcmc)

  #raw_IT[(taper_start_age + 1 - curr_age) : (taper_end_age + 1 - curr_age)]  <-
raw_IT[(taper_start_age + 1 - curr_age) : (taper_end_age + 1 - curr_age)] + taper

  #cumu_IT  <-  cumprod(1 + raw_IT)


  cumu_I   <-  cumprod(seq(from = 1 + infl_rate, by = infl_incr, length.out = row.mcmc))

  cumu_IT  <-  matrix(rep(cumu_I, iter_num_hc * iter_num_yc), nrow = row.mcmc)

  cumu_IT[index.death]  <-  0
```

```
  cumu_IT[index.disability]  <-  cumu_IT[index.disability] * (1 + taper)


  # Preparation for Calculation Factors
++++++++++++++++++++++++++++++++++++++++++End


  # Calculation of the maximum retirement income
++++++++++++++++++++++++++++++++++

  n  <-  c(3 : dim(wr)[1]) # n starts from 3 (or b# starts from 3), otherwise there will be a
dimensional issue

  RI.Collection  <-   rep(0, ncol(wr))

  f1  <-  matrix(0, nrow = length(n), ncol = ncol(wr))

  f2  <-  matrix(0, nrow = length(n), ncol = ncol(wr))

  f3  <-  matrix(0, nrow = length(n), ncol = ncol(wr))


  for (i in 1 : length(n)) {


    # f1 ++++++++++++++++++++++++++++++++++++

    f1[i, ]   <-  colProds(wr[1 : n[i], ])

    # f1 ++++++++++++++++++++++++++++++++++End
```

```
# f2 ++++++++++++++++++++++++++++++++++++

IT      <-  cumu_IT[1 : (n[i] - 1), ]

wrTemp  <-  wr[2 : n[i], ][c(nrow(wr[2 : n[i], ]) : 1), ]

wrProd  <-  apply(wrTemp, 2, cumprod)

wrAdj   <-  wrProd[c(nrow(wr[2 : n[i], ]) : 1), ]

ITwr    <-  IT * wrAdj

f2[i, ]   <-  colSums(ITwr)

# f2 ++++++++++++++++++++++++++++++++++End


# f3 ++++++++++++++++++++++++++++++++++++

SPIC    <-  inc.vec[2 : n[i], ]

SPICwr  <-  SPIC * wrAdj

f3[i, ]   <-  colSums(SPICwr)

# f3 ++++++++++++++++++++++++++++++++++End


# Closed-form solution ++++++++++++++++++

#d0_vec  <-  (curr_assets * f1[i, ] -
```

```
  #             inc.vec[1, ] * f2[i, ] + f3[i, ]) / (f1[i, ] + f2[i, ])

  # Closed-form solution +++++++++++++++++End


  #RI.Collection  <-  rbind(RI.Collection, d0_vec)


 }


 #RI.Collection  <-  as.matrix(RI.Collection[-1 , ])

 #min_RI.Collection  <- apply(RI.Collection, 2, min)

 #max.RI.MCMC  <<-  unname(quantile(min_RI.Collection, 1 - desi_succ_rate))

 # Calculation of the maximum retirement income
++++++++++++++++++++++++++++++++End


 # Calculation of the annual account value
++++++++++++++++++++++++++++++++++++++++

 n3_acc_value  <-  (curr_assets - Ret.Inc.MCMC) * f1 - (Ret.Inc.MCMC + inc.vec[1, ]) * f2 +
f3 # the first row is n = 3

 n0_acc_value  <-  rep(curr_assets, iter_num_yc)

 n1_acc_value  <-  curr_assets * wr[1, ] - Ret.Inc.MCMC * wr[1, ]
```

```
  n2_acc_value  <-  (curr_assets - Ret.Inc.MCMC) * colProds(wr[1 : 2, ]) - inc.vec[1, ] *
cumu_IT[1] * wr[2, ] + inc.vec[2, ] * wr[2, ]


  ann_acc_value_RI_MCMC <<-  unname(rbind(n0_acc_value, n1_acc_value, n2_acc_value,
n3_acc_value))


  end_acc_value_RI_MCMC <<-
ann_acc_value_RI_MCMC[nrow(ann_acc_value_RI_MCMC), ]


  low_acc_value_RI_MCMC <<-  apply(ann_acc_value_RI_MCMC[-
nrow(ann_acc_value_RI_MCMC), ], 2, min)


  sort_low_value_RI_MCMC<<-  sort(low_acc_value_RI_MCMC)

  index_low_val_RI_MCMC <<-  match(sort_low_value_RI_MCMC,
low_acc_value_RI_MCMC)


  ruin_RI_MCMC     <<-  length(union(which(low_acc_value_RI_MCMC < port_thd),
which(end_acc_value_RI_MCMC < bequest))) / ncol(ann_acc_value_RI_MCMC)

  exp_Ini_Ret_Inc  <<-  ret_spen_goal - inc.vec[1]
```

```
  ini_ass_RI_MCMC  <<-  unname(quantile(apply((exp_Ini_Ret_Inc * (f1 + f2) + inc.vec[1, ] *
f2 - f3) / f1 + max(port_thd, bequest) / f1, 2, max), desi_succ_rate))



  # Calculation of the annual account value
++++++++++++++++++++++++++++++++++++End


}
```

5. Optimization of Asset Allocation under MCMC

```
Opt_W  <-  function () {# Optimizaiton of the allocations of assets

 # Execute "Config_Readin.R" firstly


 setwd("E:\\PhD Graduation\\Model")


 # Current Client Details, Inflation & Spending Assumptions ++++++++++++++++++++++
 setup  <-  readWorkbook("FPM-v0.xlsm", sheet = "Setup", startRow = 8,
                         colNames = FALSE, rowNames = FALSE, detectDates = FALSE,
                         skipEmptyRows = TRUE, skipEmptyCols = TRUE,
                         cols = c(5 : 15),
                         check.names = FALSE, namedRegion = NULL,
                         na.strings = "NA", fillMergedCells = FALSE)


 curr_age        <-  as.numeric(setup[1, 2])
 gender          <-  setup[3, 2]
 curr_assets     <-  as.numeric(setup[4, 2])
 long_targ_age   <-  as.numeric(setup[1, 4])
```

```
desi_succ_rate  <-  as.numeric(setup[2, 4])

smoke_status    <-  setup[3, 4]

ret_spen_goal   <-  as.numeric(setup[4, 4])

infl_rate       <-  as.numeric(setup[1, 6])

taper           <-  as.numeric(setup[1, 8])

infl_incr       <-  as.numeric(setup[2, 6])

iter_num_yc     <-  as.numeric(setup[2, 8])

taper_start_age <-  as.numeric(setup[3, 6])

taper_end_age   <-  as.numeric(setup[3, 8])

iter_num_hc     <-  as.numeric(setup[4, 8])

port_thd        <-  as.numeric(setup[5, 2])

bequest         <-  as.numeric(setup[5, 4])

assets_select   <-  eval(parse(text = setup[5, 6]))

pre_allo        <-  t(matrix(rep(eval(parse(text = setup[5, 8])), dim(randRetOrig)[1]), ncol =
dim(randRetOrig)[1]))


randRet     <-  randRetOrig[ , assets_select, ]

assets.cov  <-  assets.cov.Orig[assets_select, assets_select]

# Current Client Details, Inflation & Spending Assumptions ++++++++++++++++++End
```

```
 # MCMC Selection
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
 mcmc  <-  get(paste0("MCMC", "_", curr_age, "_", gender, "_", smoke_status))


 # Remove unused variables
 #varlist  <-  ls()
 #uselessVarList  <-  grep("MCMC*", varlist, value = FALSE)
 #rm(list = varlist[uselessVarList])


 # For one scenario of health conditions, we need to do 1000 simulations of yields
 #iter_num_hc  <-  500
 mcmc  <-  mcmc[ , 1 : iter_num_hc] # iteration number of health condition simulations is set to
500
 mcmc  <-  mcmc[ , rep(1 : ncol(mcmc), rep(iter_num_yc, ncol(mcmc)))]


 # Indentify death time and disability period
 index.death  <-  which(mcmc == 3)
 index.disability  <-  which(mcmc == 2)
```

```
  # Number of rows of mcmc

  row.mcmc  <-  nrow(mcmc)



  rm("mcmc")

  # MCMC Selection
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++End



  # Post-retirement Pre-allocation of assets ++++++++++++++++++++++++++++++++++++

  #pre_allo  <-  unname(as.matrix(readWorkbook("FPM-v0.xlsm", sheet = "Port_Pre-Allo",
startRow = 9,

  #                                colNames = FALSE, rowNames = FALSE, detectDates = FALSE,

  #                                skipEmptyRows = TRUE, skipEmptyCols = TRUE,

  #                                rows = c(9 : 108), cols = c(6 : 24),

  #                                check.names = FALSE, namedRegion = NULL,

  #                                na.strings = "NA", fillMergedCells = FALSE)))

  # Post-retirement Pre-allocation of assets ++++++++++++++++++++++++++++++++++End
```

```
 # Income
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
+++
 income  <-  unname(as.matrix(readWorkbook("FPM-v0.xlsm", sheet = "Income", startRow = 9,
                              colNames = FALSE, rowNames = FALSE, detectDates = FALSE,
                              skipEmptyRows = TRUE, skipEmptyCols = TRUE,
                              rows = c(9 : 108), cols = c(5 : 9),
                              check.names = FALSE, namedRegion = NULL,
                              na.strings = "NA", fillMergedCells = FALSE)))
 inc.vec  <- apply(income, 1, sum)
 # Income
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
End


 # Objectvie function
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
 opt_fun  <-  function (W) {


   #W  <-  W / sum(W)
```

```
   # Preparation for Calculation Factors
++++++++++++++++++++++++++++++++++++++++++++
   wr  <-  matrix(0, nrow = dim(randRet)[1], ncol = iter_num_yc)


   for (i in 1 : dim(wr)[2]) {
     wr[ , i]  <-  (1 + randRet[ , , i]) %*% W
   }


   raw_IT   <-  seq(from = infl_rate, by = infl_incr, length.out = row.mcmc)
   cumu_IT  <-  cumprod(1 + raw_IT)
   cumu_IT[(taper_start_age + 1 - curr_age) : (taper_end_age + 1 - curr_age)]  <-
cumu_IT[(taper_start_age + 1 - curr_age) : (taper_end_age + 1 - curr_age)] * (1 + taper)


   cumu_I  <-  cumprod(seq(from = 1 + infl_rate, by = infl_incr, length.out = row.mcmc))
   # Preparation for Calculation Factors
++++++++++++++++++++++++++++++++++++++++++++End
```

```
   # Calculation of the maximum retirement income
++++++++++++++++++++++++++++++++++
   n  <-  c(3 : (long_targ_age - curr_age + 1)) # n starts from 3 (or b# starts from 3), otherwise there will be a dimensional issue
   RI.Collection  <-   rep(0, ncol(wr))
   f1  <-  matrix(0, nrow = length(n), ncol = iter_num_yc)
   f2  <-  matrix(0, nrow = length(n), ncol = iter_num_yc)
   f3  <-  matrix(0, nrow = length(n), ncol = iter_num_yc)

   for (i in length(n) : length(n)) {

     # f1 ++++++++++++++++++++++++++++++++++++
     f1[i, ]   <-  colProds(wr[1 : n[i], ])
     # f1 ++++++++++++++++++++++++++++++++++End

     # f2 ++++++++++++++++++++++++++++++++++++
     IT      <-  cumu_IT[1 : (n[i] - 1)]
     wrTemp  <-  wr[2 : n[i], ][c(nrow(wr[2 : n[i], ]) : 1), ]
     wrProd  <-  apply(wrTemp, 2, cumprod)
```

```
wrAdj   <-  wrProd[c(nrow(wr[2 : n[i], ]) : 1), ]

ITwr    <-  IT * wrAdj

f2[i, ]   <-  colSums(ITwr)

# f2 ++++++++++++++++++++++++++++++++++End



# f3 ++++++++++++++++++++++++++++++++++++

SPIC    <-  inc.vec[2 : n[i]]

SPICwr  <-  SPIC * wrAdj

f3[i, ]   <-  colSums(SPICwr)

# f3 ++++++++++++++++++++++++++++++++++End



# Closed-form solution ++++++++++++++++++

d0_vec  <-  (curr_assets * f1[i, ] -

            inc.vec[1] * f2[i, ] + f3[i, ]) / (f1[i, ] + f2[i, ])

# Closed-form solution +++++++++++++++++End



RI.Collection  <-  rbind(RI.Collection, d0_vec)
```

```
  }

  RI.Collection  <-  as.matrix(RI.Collection[-1 , ])

  max_d0  <-  1 / log(unname(quantile(RI.Collection, 1 - desi_succ_rate))) # Since the default objective function of solnp() in min(), so we need "1 / ..."
  # Calculation of the maximum retirement income
++++++++++++++++++++++++++++++++End

  return(max_d0)

 }
 # Objectvie function
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++End

 # Equation constraint
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
 equ_fun  <-  function(W) {
  #W  <-  W / sum(W)
```

```
  sum_weight  <-  sum(W)

  return(sum_weight)

 }

 # Equation constraint
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++End



 # Inequality constraint
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

 port_sigma  <-  function(W) {

  #W  <-  W / sum(W)

  port_sd  <-  (t(W) %*% assets.cov %*% W) ^ 0.5

  return((port_sd))

 }

 # Inequality constraint
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++End



 # Upper/lower bound of the s.d. of the portfolio ++++++++++++++++++++++++++++++++

 #W  <-  matrix(rep(1 / ncol(assets.cov), ncol(assets.cov)), ncol = 1)

 #sigma.lower  <-  solnp(pars = W,
```

```
#                   fun = port_sigma,
#                   eqfun = equ_fun,
#                   eqB = 1,
#                   LB = rep(0, ncol(assets.cov)),
#                   UB = rep(1, ncol(assets.cov)))
#
#port_sigma_neg  <-  function(W) {
#  #W  <-  W / sum(W)
#  port_sd  <-  - (t(W) %*% assets.cov %*% W) ^ 0.5
#  return((port_sd))
#}
#
#sigma.lower  <-  solnp(pars = W,
#                   fun = port_sigma_neg,
#                   eqfun = equ_fun,
#                   eqB = 1,
#                   LB = rep(0, ncol(assets.cov)),
#                   UB = rep(1, ncol(assets.cov)))$values[2]
```

```
#

min.port.sigma  <<-  min(diag(assets.cov) ^ 0.5)

max.port.sigma  <<-  max(diag(assets.cov) ^ 0.5)



epsilon  <-  0.001

ub_tr    <-  0.5



para     <-  solve(a = matrix(c(log(epsilon), log(ub_tr), 1, 1), nrow = 2, ncol = 2), b=
c(min.port.sigma, max.port.sigma))



# Upper/lower bound of the s.d. of the portfolio ++++++++++++++++++++++++++++++End



# Use solnp to solve adjusted efficient frontier ++++++++++++++++++++++++++++++++

W  <-  matrix(rep(1 / ncol(assets.cov), ncol(assets.cov)), ncol = 1)

#W  <- matrix(rep(0.01, ncol(assets.cov)), ncol = 1)

solnp.result  <<-  solnp(pars = W,

                         fun = opt_fun,

                         eqfun = equ_fun,

                         ineqfun = port_sigma,
```

```
                eqB = 1,

                ineqUB = min(para[1] * log(1 - desi_succ_rate) + para[2], max.port.sigma),

                ineqLB = min.port.sigma,

                LB = rep(0, ncol(assets.cov)),

                control = list(inner.iter = 1000))

  # Use solnp to solve adjusted efficient frontier ++++++++++++++++++++++++++++++End


}
```