ANNUITY CALCULATOR TECHNICAL DOCUMENTATION

January 31, 2019

Heather McKinnon, Brian Hooper, Divya Chandrika Kalla
Dr. Davendra
CS 565 Project 1
Central Washington University

Contents

1 Overview

This program uses mortality data from the Society of Actuaries Mortality Tables to calculate whole life single annuity premiums prices for an insurance company. The program will take a variety of user input, such as starting age, maturity age, monthly annuity benefit, and interest rate and will create a life table in which to calculate the price of the premium. This program will also simulate a number of lifetimes, given by another user input, to see the insurance company's profit over those lifetimes at the given interest rate. These lifetimes will have random starting age, maturity age, and an age of death. Several graphs are produced to help illustrate these calculations.

2 Inputs

The mortality data sets from the Society of Actuaries website contain a column for age and a column for the associated probability of death. It is downloaded as a .csv file and read into a variable, mortality_data, in this program. Another .csv file is edited by the user and read into several other variables:

```
# Read and assign input parameters
user_input <- read.csv(file="input.csv", header=TRUE, sep=",")
input_age_start = user_input[,1]
input_age_end = user_input[,2]
maturity_age = user_input[,3]
monthly_annuity = user_input[,4]
interest_rate = user_input[,5]
term_length = user_input[,6]
iterations = user_input[,7]
```

Listing 1: A user-edited file is read into the variable user_input then each value is extracted from user_input into individual variables.

With the interest rate stored in a variable, some initial variables that will be used in creating the life table in the next step are calculated.

```
# Calculated initial variables
d = interest_rate / (1 + interest_rate)
im = 12 * (((1 + interest_rate) ** (1 / 12)) - 1)
dm = 12 * (1 - (1 - d) ** (1 / 12))
a12 = (interest_rate * d) / (im * dm)
```

```
b12 = (interest_rate - im) / (im * dm)
```

Listing 2: Calculating constant variables that will be used in calculations to create the life table.

3 Creating a Life Table

After input is read in, a life table is initialized with the the age and mortality data (qx) that was read in from the .csv from the Society of Actuaries website. Having the initial age and mortality values is necessary to populate the rest of the table columns.

```
life_table <- data.frame(age, qx)
```

Listing 3: The line to create the life table data frame. The values read in from the mortality data table, age and associated mortality, are used to create the rest of the table.

After the data frame is set a series of calculations is run to set the values for the rest of the table. This life table is created ultimately to calculate values from the insurance (Eq. 1) and annuity (Eq. 2) equations. These values calculate the expected present value of the insurance premium from the starting age.

$$A_x = \sum_{k=0}^{\infty} v^{k+1} *_k|q_x \tag{1}$$

Equation 1. The Whole Life Insurance Expected Present Value Equation

$$\ddot{a}_x = \sum_{k=0}^{\infty} v^k *_k p_x \tag{2}$$

Equation 2. The Whole Life Annuity Expected Present Value Equation

4 Profit Functions

A few functions are created to reuse as the program runs through the simulations later. The functions and their documentation can be seen below.

```
# Function for determining Whole Life Net Single Premium Profit for company
#
# @param in_age The input age for beginning the insurance policy
# @param mat_age The age in which the policy matures
# @return A double representing the Net Single Premium that was paid for
    the policy
WNS_profit <- function(in_age, mat_age){
  xEy = (lx[mat_age + 1] / lx[in_age + 1]) * (1 / (1 + interest_rate)) ** (
    mat_age - in_age)
  return(monthly_annuity * 12 * (a12 * ax[mat_age + 1] - b12) * xEy)
}

# Function for determining Whole Life Net Single Premium loss for company
# Occurs only when death_age > maturity_age
#
# @param mat_age The age in which the policy matures
# @param death_age The age in which the policy holder dies
# @return A double representing the total paid out to the client for the
    policy
WNS_loss <- function(mat_age, death_age)
  return ((death_age - mat_age) * monthly_annuity * 12)

# Calculate net profit or loss for Whole Life Net Single Premium
#
# @param in_age The input age for beginning the insurance policy
# @param mat_age The age in which the policy matures
# @param death_age The age in which the policy holder dies
# @return A double representing the net profit or loss for a single policy
    holder
WNS_net_profit <- function(in_age, mat_age, death_age){
  if (death_age < maturity_age){
    #    profit = ((input_age - dead_age) * desired_monthly_benefit)
    return (WNS_profit(in_age, mat_age))
  }
  else{
    #    loss = ((dead_age - maturity_age) * desired_monthly_benefit)
    return (WNS_profit(in_age, mat_age) - WNS_loss(mat_age, death_age))
  }
}
```

Listing 4: Three functions to calculate the insurance company's net profit for whole single net premiums. One calculates the profit (or the premium price), one the loss as annuity payments are made to the policy holder, and one to return the net profit.

The first function, WNS_profit, will calculate a whole life net single premium for a given starting age and maturity age. This value can be used to check the validity of the calculations for the whole program with known data. The second function, WNS_loss, calculates and returns any loss from the insurance company if the age of maturity is reached before death of the policy holder. The third function, WNS_net_profit, calculates and returns the combined profit and loss for the company.

5 Simulations

With the set user input of iterations for lifetime simulations, this program will loop through a randomly generated starting age, maturity age, and death age to determine the insurance company's profit or loss on that individual's life policy. This simulation is set up to test if every randomly generated client chose the net single premium option and adds the loss or gain after each iteration in the variable, profit.

```
profit <- 0
for(i in 1:iterations) {
  # Generate random integer starting age
  if(input_age_start >= input_age_end) {
    input_age = input_age_start
  } else {
    input_age = sample(input_age_start:input_age_end, 1)
  }

  # Pick a random death date based on mortality table
  death_age = input_age
  while(death_age < length(mortality_data$mortality) && runif(1, 0.0, 1.0)
   > mortality_data$mortality[death_age]) {
    death_age = death_age + 1
  }

  # Calculate profit
  profit <- profit + WNS_net_profit(input_age, maturity_age, death_age)
}
```

Listing 5: The loop to run simulations for user-defined number of iterations. Policy starting age is randomly chosen in the age range between age start and age end and the death age is determined by using the mortality percentage form the life table for each year the policy holder lives. Once the death age and start age is determined, the profit on that policy can be determined.

6 Graphing

Four graphs are generated to illustrate some of the output data: age effect on mortality, age on the annuity expected present value (ä), whole life net single premium profit accumulation after each life simulation iteration, and how increasing age with the user-defined maturity age and monthly benefit of net single premium prices. Sample output graphs can be seen in the figures 1, 2, 3, and 4 below.

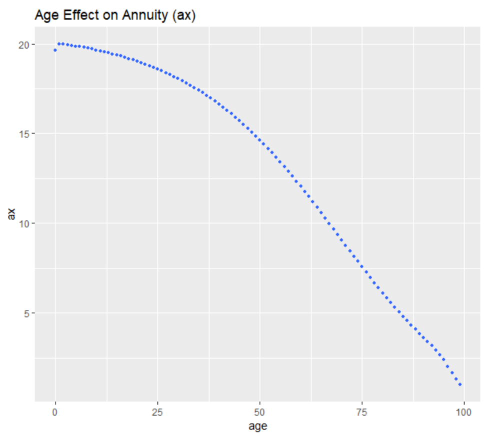

Figure 1: Chance of mortality at a given age.

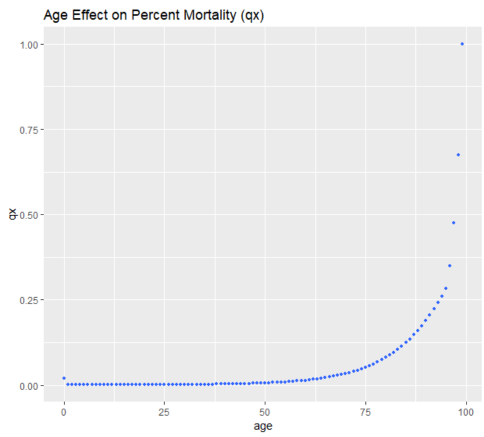

Figure 2: Annuity expected present value at a given age.

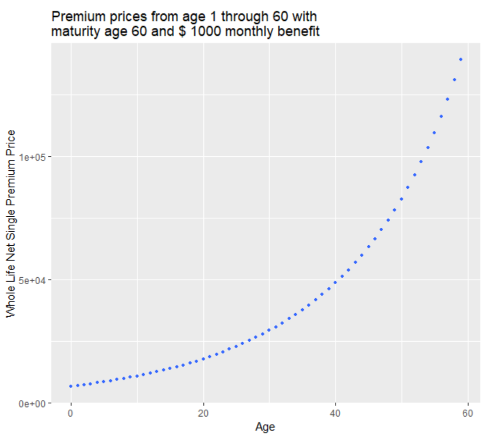

Figure 3: Prices of whole life single net premium at given age range, maturity age, and desired annuity benefit.

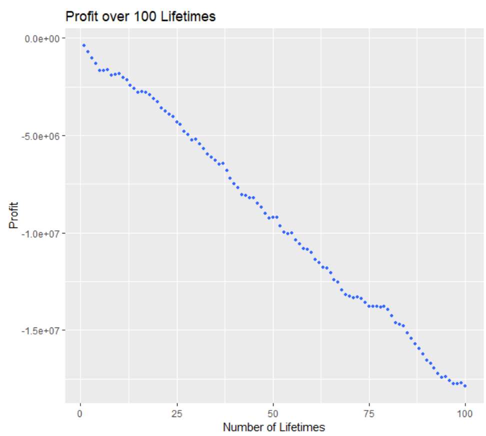

Figure 4: Company's profit over a user-defined number of lifetimes of whole life single net premium policy holders.

References

[1] Society of Actuaries Mortality and Other Rate Tables. Available: https://mort.soa.org/?_ga=2.155909691.1059898131.1547157104-186254515.1517263599.