

---

# ANNUITY CALCULATOR TECHNICAL DOCUMENTATION

---

February 10, 2019

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

## Contents

1	Overview	3
2	Inputs	3
3	Creating a Life Table	4
4	Profit Functions	4
5	Simulations	6
6	Graphing	8
7	Future Considerations	10

## 1 OVERVIEW

This program uses mortality data from the [Society of Actuaries Mortality Tables](#) to simulate a number of whole life single annuity premiums prices for an insurance company and produce projected annual profits. The mortality data is used to create a life table in which to draw from to create the policy table. The program will user input of a starting age range, maturity age, monthly annuity benefit, and interest rate and will create a simulated number of policies with random starting and death ages and their calculated premiums. This program will also simulate a profit for a projected number of years to see the insurance company's annual profit. These projected years will take user input of number of years, ROI interest, investment percentage, and a sales goal for number of policies to sell per year. 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:

```

1 # Read and assign input parameters
2 user_input <- read.csv(file="input.csv", header=TRUE, sep=",")
3 input_age_start = user_input[,1]
4 input_age_end = user_input[,2]
5 maturity_age = user_input[,3]
6 monthly_annuity = user_input[,4]
7 interest_rate = user_input[,5]
8 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.

```

1 # Calculated initial variables
2 d = interest_rate / (1 + interest_rate)
3 im = 12 * (((1 + interest_rate) ** (1 / 12)) - 1)

```

```

4 dm = 12 * (1 - (1 - d) ** (1 / 12))
5 a12 = (interest_rate * d) / (im * dm)
6 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 ( $q_x$ ) 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.

```

1 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.

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

*Equation 1. The Whole Life Insurance Expected Present Value Equation*

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

*Equation 2. The Whole Life Annuity Expected Present Value Equation*

### 4 PROFIT FUNCTIONS

Functions were created to reuse as the program runs through the simulations (Listing 4).

```

1 # Function for determining Whole Life Net Single Premium Profit for company
2 #
3 # @param in_age The input age for beginning the insurance policy
4 # @param mat_age The age in which the policy matures
5 # @return A double representing the Net Single Premium that was paid for
    the policy
6 WNS_profit <- function(in_age, mat_age){
7   xEy = (lx[mat_age + 1] / lx[in_age + 1]) * (1 / (1 + interest_rate)) ** (
      mat_age - in_age)
8   return(monthly_annuity * 12 * (a12 * ax[mat_age + 1] - b12) * xEy)
9 }
10
11 # Function for determining Whole Life Net Single Premium loss for company
12 # Occurs only when death_age > maturity_age
13 #
14 # @param mat_age The age in which the policy matures
15 # @param death_age The age in which the policy holder dies
16 # @return A double representing the total paid out to the client for the
    policy
17 WNS_loss <- function(mat_age, death_age)
18   return ((death_age - mat_age) * monthly_annuity * 12)
19
20 # Calculate net profit or loss for Whole Life Net Single Premium
21 #
22 # @param in_age The input age for beginning the insurance policy
23 # @param mat_age The age in which the policy matures
24 # @param death_age The age in which the policy holder dies
25 # @return A double representing the net profit or loss for a single policy
    holder
26 WNS_gross_profit <- function(in_age, mat_age, death_age){
27   if (death_age <= maturity_age){
28     return (WNS_profit(in_age, mat_age))
29   }
30   else
31     return (WNS_profit(in_age, mat_age) - WNS_loss(mat_age, death_age))
32 }

```

**Listing 4:** Three functions to calculate the insurance company's gross 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 gross profit.

The first function, `WNS_profit()`, will calculate a whole life net single premium for a single policy holder with a unique starting age and maturity age. This value can be

With the user-defined input of iterations, this program will generate lifetimes for individual policy holders in a loop (Listing 5). These individuals will have a randomly generated starting age (age they bought the insurance) and death age with a given maturity age to determine the price of the premium and the insurance company's gross profit on that individual's life policy. This simulation loop is set up to test if every client chose the net single premium option. All data generated by this loop is stored in a data frame called `policy_table`. This table stores the starting age, maturity age, death age, policy premium price, a logical displaying if the individual died before the maturity date, and the gross profit at the end of the individual's lifetime.

[illegible]

```

21     WNS_gross_profit(input_age, maturity_age, death_age))
22 }

```

**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 policy premium price and gross profit can be determined.

The second simulation in this program (Listing 6) determines the projected yearly annual gross income for a user-defined set of years. A .csv file takes user input of number of years, ROI interest, investment percentage, and a yearly sales goal. The yearly sales goal determines how many policies are sold per year of the simulation. Before the loop initiates, the yearly sales goal number of these policies is randomly selected from the pool of individuals that were generated in the lifetime simulation. This sets the year zero profit from "selling" these policies. Then the investment percentage of this profit is "invested" for year zero.

Once the loop begins, it checks to see if the individual policy holder is still alive and if the age of the policy has reached the maturity date yet. If the maturity date is met and the client is still alive, then this will contribute to a loss for the insurance company, as the annuity payments are paid out.

After the loss is determined, the number of new policies sold is again randomly selected from the lifetime simulation pool. The new policies are added to the end of the previous policies and the polices age for all older policies is incremented by 1 (year). These new policies will add to yearly profit and a portion equal to the investment percent is added to the amount invested.

Finally, the data generated by this loop is added to a table called ROI\_tracker. This table will show year, total loss (payouts), sold\_policies, ROI (calculated by ROI interest), invested amount, and the final ROI\_adjusted\_profit. The ROI adjusted profite represents the gross annual profit for one year.

```

1 for (i in 2:company_years){
2   # checking the current age of the policy holder (if mature, and the
   policy holder is not dead yet)
3   one_year_loss = 0
4   for (j in 1:nrow(policies)){
5     if (isTRUE(policies$StartAge[j] + policies$policyAge[j] > policies$
       MatAge[j]) && isTRUE(policies$StartAge[j] + policies$policyAge[j] <
       policies$DeathAge[j])) {

```

```

6     loss <- WNS_loss(policies$MatAge[j], policies$DeathAge[j])
7     one_year_loss = one_year_loss + loss
8   }
9 }
10
11 # add new policies sold
12 new_policies <- policy_table[sample(nrow(policy_table), policy_sales_goal)
13   ,]
14 new_policies$policyAge <- c(0)
15 policies <- rbind(policies, new_policies)
16 policies$policyAge <- policies$policyAge + 1 # increment policy ages
17
18 # concatenate data from loop to ROI variables
19 year <- c(year, i-1)
20 total_loss <- c(total_loss, one_year_loss)
21 sold_policies <- c(sold_policies, sum(new_policies$PolicyCost))
22 ROI <- c(ROI, invested[i-1] * ROI_interest)
23 # This reinvests the ROI for the year and the investment_percent value of
24   the policies_sale_goal sold for the year
25 invested <- c(invested, ROI[i] + invested[i-1] + (sold_policies[i] *
26   investment_percent))
27 ROI_adjusted_profit <- c(ROI_adjusted_profit, (invested[i] + sold_
28   policies[i] - (sold_policies[i] * investment_percent) - one_year_loss))
29 } # End yearly profit projections

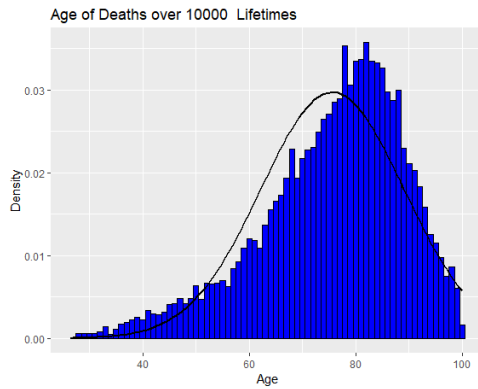
```

**Listing 6:** The for loop for simulating projected profit for a set number of years for the insurance company selling whole single net premium annuities.

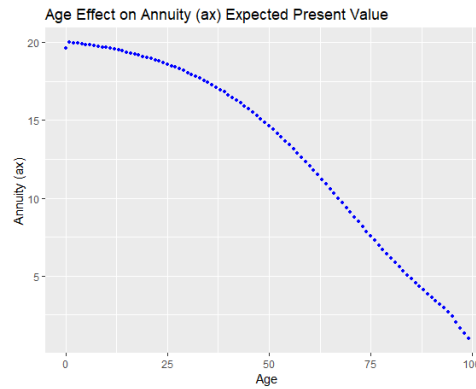
## 6 GRAPHING

Four graphs are generated to illustrate some of the output data: a histogram of death ages generated by the simulated lifetimes loop, age effect on the annuity expected present value ( $\ddot{a}_x$ ), how increasing age with the user-defined maturity age and monthly benefit of net single premium prices, and a projected gross income for a user-defined set of years with sales, investments, ROIs, and paying client's their annuities. Sample output graphs can be seen in the figures [1](#), [2](#), [3](#), and [4](#) below.

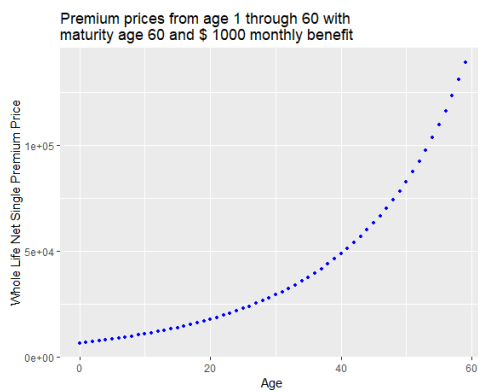




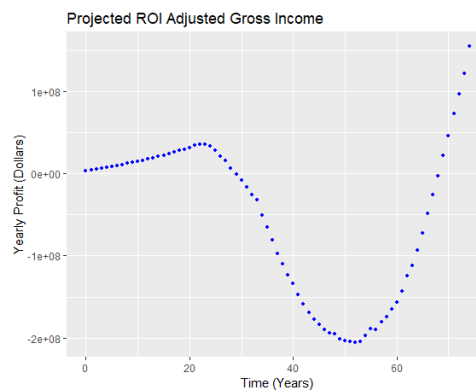
**Figure 1:** Histogram of deaths at a given age.



**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 projected gross profit per year adjusted with ROI from investments and selling a user-defined number of whole life single net premium policies per year.

## 7 FUTURE CONSIDERATIONS

There are a few future considerations for expanding this program. One obvious one would be adding yearly, monthly, and all n-term policy premiums. As it is, this program calculates only whole net single premiums prices. This would probably take a major overhaul of existing code and make it considerably longer. Another small modification might be to have a single, easy way for a user to test just one policy. This is entirely dependant on the user's needs however, but may help if the user wants to give a single client an estimate. A third consideration may be adjusting the loop that forecasts yearly gross profits. As it stands, the profit projection is calculated with an initial set of policy sales but there are no policy holders with a policy age over zero years when the loop starts. This creates data that looks very profitable in the first few years because there are no annuity checks being paid out. However, the data generated can still be useful, and again depends on the user's needs.

## REFERENCES

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