Benefit Modelling

Helitha Dharmadasa - z5451805

2024-02-23

```
library(tidyverse)
library(readxl)
```

Read Data

```
superlife_df <- read_csv("../Data/Processed Data/CLEANED_2024-srcsc-superlife-inforce-dataset.csv")
```

```
## Rows: 978582 Columns: 18
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (9): Policy.number, Policy.type, Sex, Smoker.Status, Underwriting.Class,...
## dbl (9): Issue.year, Issue.age, Face.amount, Region, Death.indicator, Year.o...
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
rate_df <- read_excel("../Data/Case Study Data/srcsc-2024-lumaria-economic-data.xlsx",
    skip = 10)
```

```
mortality_df <- read_excel("../Data/Case Study Data/srcsc-2024-lumaria-mortality-table.xlsx",
    skip = 13)

adj_mortality_df <- read_excel("../Data/Processed Data/average_mortality_table.xlsx")
```

Transform Mortality tables

```
# Create lx (Number of people surviving) to life tables
mortality_df <- mortality_df %>%
    mutate(survival_rate = 1 - `Mortality Rate`, lx = cumprod(survival_rate))

adj_mortality_df <- adj_mortality_df %>%
    mutate(survival_rate = 1 - new_mortality, lx = cumprod(survival_rate))

write_csv(mortality_df, "../Data/Processed Data/mortality_baseline.csv")

write_csv(adj_mortality_df, "../Data/Processed Data/mortality_adjusted.csv")
```

Create benefit modelling function

```
benefit_model <- function(policy_duration, mortality_table, interest_rate, inflation_rate) {
    # Function that takes generates $1 EPVs for different payment structures.
    # Two types of EPV's are calculated, single payout - paying out $1 on
    # death, and annuity_dues paying $1 yearly until death or artifical cap.
    # Inputs: policy_duration - (20 Years Term or 120 Years whole life)
    # mortality_table - Table containing lx values interest_rate - interest
    # rate to use for discounting inflation_rate - inflation rate to calculate
    # real rate Outputs: output_df$EPV_single - EPV of a insurance polcy paying
    # out death output_df$EPV_annuity_due - EPV on an annuity paying until
    # death output_df$EPV_single_18 - EPV of an insurance policy that starts
    # 18+ paying $1 on death output_df$EPV_single_50 - EPV of an insurance
    # policy that starts at 18+ up to 88 output_df$EPV_annuity_due_18 - EPV of
    # an annuity starting at 18+ output_df$EPV_annuity_due_50 - EPV of an
    # annuity starting 50+

    real_rate = ((interest_rate - inflation_rate)/(1 + inflation_rate))

    v = 1/(1 + real_rate)

    output_df <- tibble(age = 1:120, EPV_single = rep(0, 120), EPV_annuity_due = rep(0,
        120), EPV_single_18 = rep(0, 120), EPV_single_50 = rep(0, 120), EPV_annuity_due_18 = rep(0,
        120), EPV_annuity_due_50 = rep(0, 120))

    # Ages 1-119 Note: 119 because we don't have Age 121 in life table for 120
    # calc.
    for (starting_age in 1:119) {

        single = 0
        annuity_due = 0
        single_18 = 0
        single_50 = 0
        annuity_due_18 = 0
        annuity_due_50 = 0

        # Rolling window of policy dur or capped at life table limits
        # (truncation error?)
        age_max <- min(starting_age + policy_duration, 119)

        for (death_age in starting_age:age_max) {

            t = death_age - starting_age

            # P(starting age is alive until death age)
            tpx = mortality_table$lx[death_age]/mortality_table$lx[starting_age]

            # P(death age dies in the next year)
            qxt = 1 - (mortality_table$lx[death_age + 1]/mortality_table$lx[death_age])

            single = single + v^(t + 1) * tpx * qxt  #Paid out EOY of Death
            annuity_due = annuity_due + v^(t) * tpx  #Paid out SOY of every year alive
```

```
            if (death_age >= 18) {
                # Paid on death if older than 18
                single_18 = single_18 + v^(t + 1) * tpx * qxt

                # Paid SOY yearly starting at 18
                annuity_due_18 = annuity_due_18 + v^(t) * tpx
            }

            if (death_age >= 50) {
                # Paid SOY yearly starting at 50
                annuity_due_50 = annuity_due_50 + v^(t) * tpx
            }

            if (death_age == 27) {
                # Paid at 50 if alive at 50
                single_50 = v^(t) * tpx
            } else if (starting_age > 27 & starting_age <= 88) {
                single_50 = 1
            }
        }

        output_df$EPV_single[starting_age] = single
        output_df$EPV_annuity_due[starting_age] = annuity_due
        output_df$EPV_single_18[starting_age] = single_18
        output_df$EPV_single_50[starting_age] = single_50
        output_df$EPV_annuity_due_18[starting_age] = annuity_due_18
        output_df$EPV_annuity_due_50[starting_age] = annuity_due_50

    }

    return(output_df)
}
```

```
# Call benefit_model for different terms
interest_rate <- mean(rate_df$`1-yr Risk Free Annual Spot Rate`)
inflation_rate <- mean(rate_df$Inflation)

T20_EPV_df <- benefit_model(20, mortality_df, interest_rate, inflation_rate) %>%
    mutate(EPV_single_adj = benefit_model(20, adj_mortality_df, interest_rate, inflation_rate)$EPV_singl

SPWL_EPV_df <- benefit_model(120, mortality_df, interest_rate, inflation_rate) %>%
    mutate(EPV_single_adj = benefit_model(120, adj_mortality_df, interest_rate, inflation_rate)$EPV_sing
```

```
# Average face values for the two policy types
T20_FV <- superlife_df %>%
    filter(Policy.type == "T20") %>%
    summarise(Mean = mean(Face.amount)) %>%
    pull(Mean)

SPWL_FV <- superlife_df %>%
    filter(Policy.type == "SPWL") %>%
    summarise(Mean = mean(Face.amount)) %>%
    pull(Mean)
```

```r
# Engagement rate (same for all interventions)
engagement_rate = 0.25

# Calculate expense dataframes for the two policies
T20_expense_df <- tibble(Age = T20_EPV_df$age,
                         smoking = T20_EPV_df$EPV_single_50 * 2065 * 0.18,
                         screening = T20_EPV_df$EPV_annuity_due_50 * 65,
                         fitness =  T20_EPV_df$EPV_annuity_due_18 * 18,
                         safety = T20_EPV_df$EPV_annuity_due_18 * 12.5)

SPWL_expense_df <- tibble(Age = SPWL_EPV_df$age,
                         smoking = SPWL_EPV_df$EPV_single_50 * 2065 * 0.18,
                         screening = SPWL_EPV_df$EPV_annuity_due_50 * 65,
                         fitness =  SPWL_EPV_df$EPV_annuity_due_18 * 18,
                         safety = SPWL_EPV_df$EPV_annuity_due_18 * 12.5)

# Calculate total EPVs to compare the different programs
benefit_df <- tibble(Age = SPWL_EPV_df$age,
                         T20_baseline = T20_EPV_df$EPV_single * T20_FV,
                         T20_intervention = ((1-engagement_rate) * T20_EPV_df$EPV_single + engagement_rate 
                           engagement_rate * (T20_expense_df$smoking +
                           T20_expense_df$screening +
                           T20_expense_df$fitness +
                           T20_expense_df$safety),

                         SPWL_baseline = SPWL_EPV_df$EPV_single * SPWL_FV,
                         SPWL_intervention = ((1-engagement_rate) * SPWL_EPV_df$EPV_single + engagement_rat
                           engagement_rate * (SPWL_expense_df$smoking +
                           SPWL_expense_df$screening +
                           SPWL_expense_df$fitness +
                           SPWL_expense_df$safety))

benefit_df <- benefit_df %>%
  mutate(T20_profit_flag = factor(ifelse(T20_intervention < T20_baseline, "Profit", "Loss")),
         SPWL_profit_flag = factor(ifelse(SPWL_intervention < SPWL_baseline, "Profit", "Loss")))

write_csv(benefit_df, "../Data/Processed Data/Benefit_Modelling.csv")
```

```r
# Calculate benefit comparison dataframe aggregated to different age brackets
summary_df <- benefit_df %>%
    filter(Age < 120) %>%
    mutate(Age_bracket = cut(Age, breaks = c(1, 23, 45, 60, 85, 120), labels = c("1-22",
        "23-44", "45-59", "60-84", "85+"), right = FALSE))



summary_df <- summary_df %>%
    group_by(Age_bracket) %>%
    summarise(across(c(T20_baseline, T20_intervention, SPWL_baseline, SPWL_intervention),
        ~round(mean(.), 0)))

write_csv(summary_df, "../Data/Processed Data/Benefits_by_Age_Group.csv")
```