Minimizing the Probability of Ruin in Retirement

NOTE: the project is done for Lucas Weatherill (OnTrack Retirement).

The app is based on CHRISTOPHER J. ROOK C++ implementation from the doc.

How to run

Compile and run

```
# MinimizeRuinProbability.exe [<optional parameter: number of threads>]
```

Input files are created automatically with default values.
A single parameter is optionally accepted which is the number of tasks the user wants to process concurrently as it runs. If no value is supplied the code determines the maximum number of concurrent processing units on the machine running it and uses this value. The maximum number of concurrent processing units is generally the number of CPU cores.

Configure

There are two input files that are created automatically in "./config/" directory if do not exist.

control.txt

Format:

```
StockMean StockVar BondMean BondVar StockBondCov RF_Max E_R PrunePwr
P_R P_α
NumRand Details
```

Where:

StockMean, StockVar, BondMean, BondVar, StockBondCov - the means/variances match the historical values. They are not changing too much with time, so you can leave it constant always (see Appendix F of the original doc).

RF_{Max} – max Ruin Factor(RF) that is used in calculations and results.

E_R - expense ratio charged by the financial institution per time t. 0.005 value, means 0.5% are charged.

PrunePwr – is used to improve the performance for random years in retirement (*NumRand* > 0).

P_R - discretization precision for RF calculation, i.e. how many buckets (discrete values per 1 RF value) to use to calculate numerical approximation to the optimal decumulation strategy (See 3.7.4 section of the original doc).

P_α - discretization precision for *α*.

NumRand – use 0 for fixed number of years in retirement, or N for 1+ persons for random retires calculations (See 4.6.2 from original doc for details), with age, based on ageprobs.txt.

Details – if *NumRand* is 0: <T_D - number of years in retirement>. If *NumRand* > 0: pairs of two values: <gender letter M or F> <person age>.

ageprobs.txt

Columns in this file represent: age, male death probability, and female death probability for a 50-year old. They are computed from life-tables published at SSA.gov and each column sums to 1.

Performance tips

1. The app performance highly depends on data types are used for calculations. In the original C++ code, long double is used with 50 significant simbols after the point (in windows x64 Visual Studio environment). This floating point precision is not supported by hardware so leads to serious performance degradation.

   From practical point of view it's enough to use *double* data type with 15 significant digits, that is supported by hardware directly, and it works more than 10x times faster than other more accuracy data types. Still it gives good results with precision over $\sim 10^{-10}$ comparing with the original version.

2. In the original doc examples you can see P_R=10000 and P_α=1000 are used sometimes, and sure it leads to hours of calculations even on server machine with 40 virtual cores. It has sense from scientific point of view probably, but from practical point of view it's enough to use P_R=1000 and P_α=100.

Output

4 files as result:

- *MinimizeRuinProbability.exe.log* - log file, all you have on console is saved there.
- *FinalAlphaResults_H.csv* - α values for a given RF value and time - optimal asset allocation in stocks.
- *FinalProbResults_H.csv* - V_R values for a given RF value and time – minimum probability of ruin table.
- *FinalResults_V.txt* - concatenation of two previous files. Each row is time point [t], ruin factor RF(t), minimum probability of ruin V_R(t, RF(t)), and optimal asset allocation α_R(t, RF(t)).

How to use

The app gives the answer to the question – what's the optimal amount should be invested to stocks (and the rest to bonds) and what's the probability of ruin in this case, with the given value of retirement account and the given initial withdrawal rate adjusted for inflation each year. This optimal portfolio balance (and ruin probability) recalculated each year, based on the level of inflation and portfolio returns. That gives you the optimal investment algo to be used within all retirement period.

Example

Let's say we have 2 male persons, retirement age 65, with the initial retirement account $700.000 after removing funds to cover first year spending.

Here is the *control.txt* file we are using:

```
0.082509 0.0402696529 0.021409 0.0069605649 0.0007344180 2.5 0.005 4.00
1000 100
1 M 65
```

In result we have *FinalResults_V.txt*:

```
...
0 0.0390000000 0.0176536334869191000000000000000000000000000000000000 0.350000000
0 0.0400000000 0.0207275506316957000000000000000000000000000000000000 0.360000000
0 0.0410000000 0.0241273425680343000000000000000000000000000000000000 0.370000000
0 0.0420000000 0.0278580945637325000000000000000000000000000000000000 0.380000000
0 0.0430000000 0.0319220565270613000000000000000000000000000000000000 0.390000000
0 0.0440000000 0.0363187252487050000000000000000000000000000000000000 0.400000000
0 0.0450000000 0.0410449849717383000000000000000000000000000000000000 0.410000000
0 0.0460000000 0.0460952954812525000000000000000000000000000000000000 0.420000000
...
0 0.0600000000 0.1439423603632370000000000000000000000000000000000000 0.620000000
0 0.0610000000 0.1522574905170670000000000000000000000000000000000000 0.640000000
...
```

First number in the row means 0 years from retire start, i.e. when our men are 65 years old,
The second number is Ruin Factor(RF) we have (explained below),
3rd - ruin's probability for the corresponding RF (means the chance retiree will not be able to make withdrawal in his last year before the death),
and the last one (α) - an optimal asset allocation, i.e. percents of the account should be invested to stocks, and the rest to bonds.

First year

RF for the first year - RT(0) equals to the amount a man would like to withdraw each year.

Both men should choose what's the amount they want to spend each year.

Imagine, first one want to spend 4% of his initial account each year ($28.000), so we have RT(0) = 0.04 for him, with the minimum ruin probablity 2.07% in case of we have 36% in stocks portfolio. And based on the model it's the best choise for the portfolio for the first year.

The second one want to spend $42.000 per year (6%), so he has TR(0) = 0.06, and the best way for him is to invest 62% to stocks, 38% to bonds, and he has 14.39% ruin's probability in this case.

Second year

One year passed. For example, inflation was 2.5%. First retiree portfolio return is +1.5%, second retiree portfolio is +5%.
Inflation adjusted returns are -1% and 2.5% respectively.

Now we want to calculate RT(1), to do optimal portfolio rebalancing.

We use the formula: $RF(t)=RF(t-1)/[\hat{r}_{(t,\alpha)} - RF(t-1)]$.
First we calculate $\hat{r}_{(t,\alpha)} = (1+r_{(t,\alpha)})(1-E_R)$ - inflation/expense adjusted compounded return on the portfolio;
E_R - expense ratio charged by the financial institution per time t (0.5%, i.e. 0.005 in our formula);
$r_{(t,\alpha)}$ - inflation-adjusted rate of return. It is (1-0.01) = 0.99 for the first retiree, and (1+0.025) = 1.025 for the second one.

I.e. $\hat{r}_{(t,\alpha)}$ equals to 0.99*0.995=0.98505 for the first, and 1.025*0.995=1,019875 for the second retiree.
I.e. RT(1) equals to 0.04/(0.98505 - 0.04) = 0,0423258028675731 for the first and 0.06/(1,019875 - 0.06) = 0,0625081390806095 for the second retiree.

Let's check our *FinalResults_V.txt* again:

```
...
1 0.0420000000 0.0239914323644930000000000000000000000000000 0.370000000
1 0.0430000000 0.0276209582670290000000000000000000000000000 0.380000000
...
1 0.0620000000 0.1478529196827970000000000000000000000000000 0.630000000
1 0.0630000000 0.1559913711683230000000000000000000000000000 0.650000000
...
```

We should use the nearest RT value (each value from the file is the center of the corresponding RT bucket).
I.e. we are using RF 0.042 for the first and 0.063 for the second.
So, the optimal portfolio strategy for the second year is to use 37% in stocks for the first and 65% in stocks for the second retiree.