Life Insurance and Mortality Swaps
Dublin R

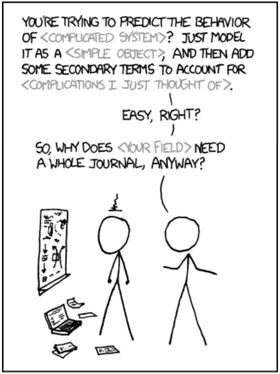
Mick Cooney
michael.cooney@applied.ai

23 June 2015

Structure of Talk

1. Pricing Life Insurance
2. MonteCarlo Simulation and Interest-rate Swaps
3. Pricing Mortality Swaps

Before I Begin...



LIBERAL-ARTS MAJORS MAY BE ANNOYING SOMETIMES, BUT THERE'S *NOTHING* MORE OBNOXIOUS THAN A PHYSICIST FIRST ENCOUNTERING A NEW SUBJECT.

Initial Idea for Pricing Mortality Swaps

Use idea from interest rate swaps:

'Swap' the cashflows

Pricing Life Insurance

$$\sum_{t=0}^{N-1} A e^{-rt} = \sum_{t=1}^{N} L\, q(t) e^{-rt}$$

where

$$\begin{aligned} A &= \text{policy premium} \\ L &= \text{death payment} \\ q(t) &= \text{curtate mortality from time } t-1 \text{ to } t \\ r &= \text{interest rate (annualised)} \\ N &= \text{term of the policy (in years)} \end{aligned}$$

Lifetables

```
##       Age         q(x)     l(x)     d(x)    L(x)    T(x)    e(x)
##  1:   0-1 6.12294e-03 100000.0 612.2943 99465.5 7866026 78.6603
##  2:   1-2 4.28382e-04  99387.7  42.5759 99366.4 7766561 78.1441
##  3:   2-3 2.74978e-04  99345.1  27.3177 99331.5 7667194 77.1774
##  4:   3-4 2.10585e-04  99317.8  20.9148 99307.3 7567864 76.1985
##  5:   4-5 1.57760e-04  99296.9  15.6650 99289.1 7468556 75.2144
##  6:   5-6 1.45108e-04  99281.2  14.4065 99274.0 7369267 74.2262
##  7:   6-7 1.27664e-04  99266.8  12.6728 99260.5 7269993 73.2369
##  8:   7-8 1.13604e-04  99254.1  11.2757 99248.5 7170732 72.2462
##  9:   8-9 9.97674e-05  99242.9   9.9012 99237.9 7071484 71.2543
## 10:  9-10 8.67807e-05  99233.0   8.6115 99228.7 6972246 70.2614
```

```
##     age         qx
##  1:   0 0.00612294
##  2:   1 0.00042838
##  3:   2 0.00027498
##  4:   3 0.00021058
##  5:   4 0.00015776
##  6:   5 0.00014511
##  7:   6 0.00012766
##  8:   7 0.00011360
##  9:   8 0.00009977
## 10:   9 0.00008678
```

Calculating the Premium

Price for 20-year policy of a 30 year old (per $100,000 assured)

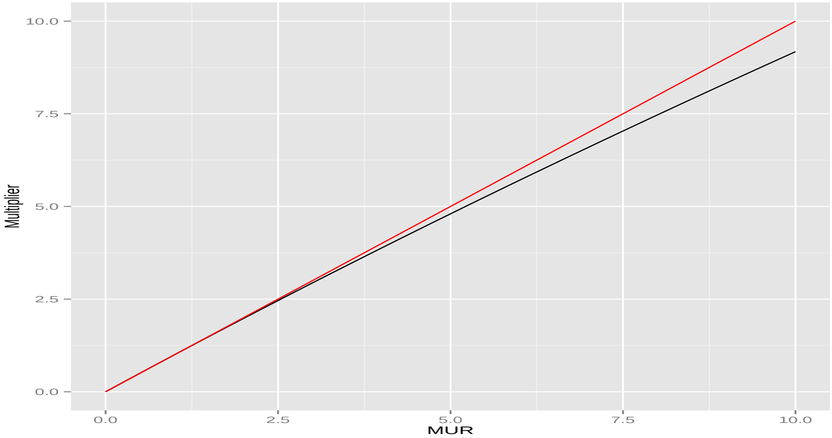
```
price.term.assurance <- function(q, A = 100000, r = 0.05, P = 0) {
    ### Calculates the price of term assurance by equating the expected values

    N <- length(q);
    c <- cumprod(1 - q);
    c <- c(1, c[1:(N - 1)]);

    LHS <- sum(c * exp(-r * 0:(N-1)));

    RHS <- P + sum(A * q * exp(-r * 1:N));

    return(RHS / LHS);
}

price.term.assurance(lifetable.dt[age > 30][age <= 50]$qx, A = 100000, r = 0.05, P = 0);


## [1] 176.966
```

Calculating the MUR multiplier

```
A  <- 100000;
qx <- lifetable.dt[age >= 30][age < 50]$qx;
r  <- 0.05;

calculate.multiple.diff <- function(MUR, mult) {
    MUR * price.term.assurance(qx, A = A, r = r, P = 0) -
        price.term.assurance(mult * qx, A = A, r = r, P = 0);
}

MUR.values <- seq(0, 10, by = 0.25);

MUR.mult <- sapply(MUR.values, function(MUR) {
    optimize(function(mult) abs(calculate.multiple.diff(MUR, mult)), c(0, 20))$minimum;
});

print(MUR.mult);
```

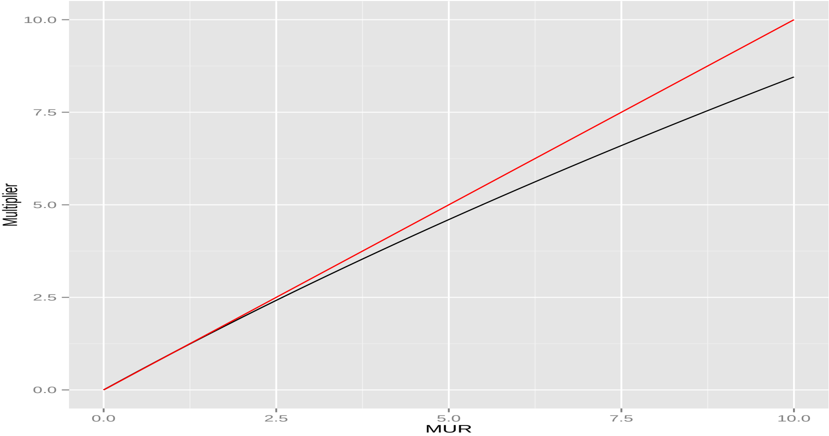
```
##  [1] 7.36803e-05 2.52050e-01 5.02719e-01 7.52001e-01 1.00000e+00
##  [6] 1.24667e+00 1.49207e+00 1.73615e+00 1.97899e+00 2.22063e+00
## [11] 2.46100e+00 2.70016e+00 2.93818e+00 3.17498e+00 3.41064e+00
## [16] 3.64516e+00 3.87858e+00 4.11083e+00 4.34202e+00 4.57214e+00
## [21] 4.80115e+00 5.02912e+00 5.25605e+00 5.48193e+00 5.70682e+00
## [26] 5.93069e+00 6.15357e+00 6.37547e+00 6.59639e+00 6.81637e+00
## [31] 7.03537e+00 7.25346e+00 7.47062e+00 7.68689e+00 7.90224e+00
## [36] 8.11667e+00 8.33023e+00 8.54296e+00 8.75478e+00 8.96575e+00
## [41] 9.17589e+00
```

Calculating the MUR multiplier



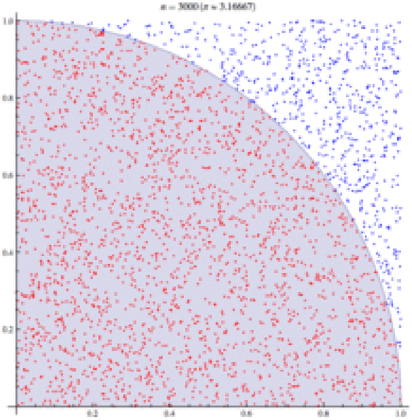
Calculating the MUR multiplier

Try for 45-year old with 15-year policy:



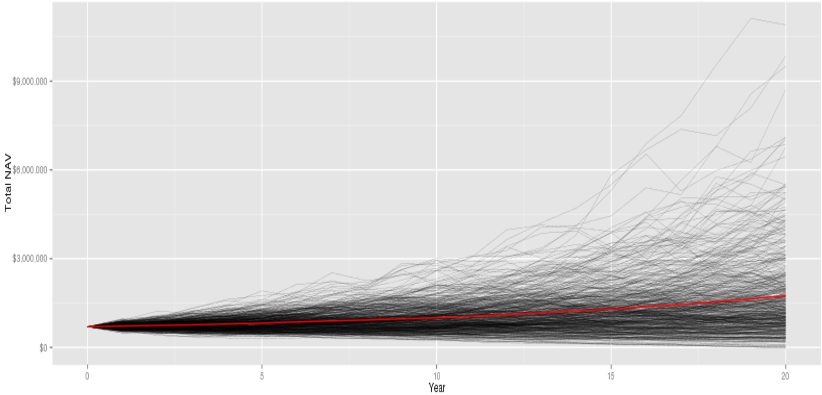
MonteCarlo Methods

Calculating the value of π:

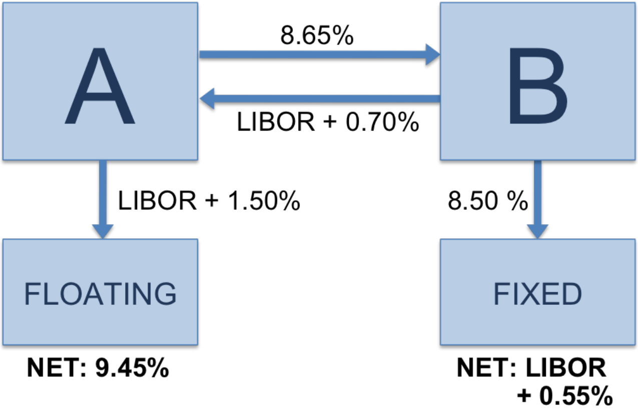


MonteCarlo Methods

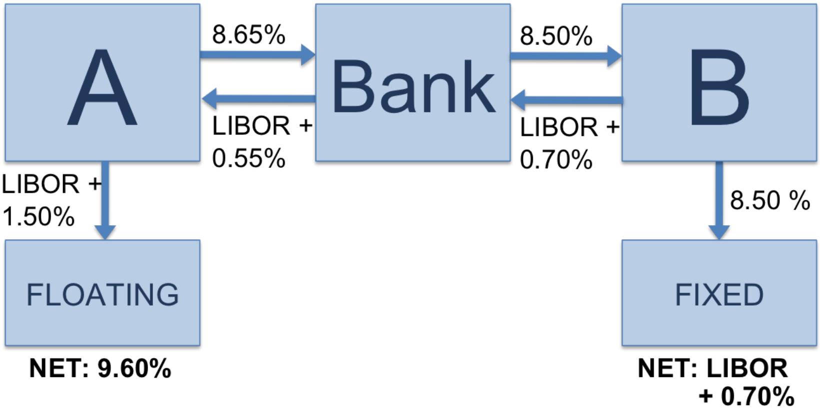
Calculating the NAV of a fund of correlated assets with a yearly drawdown:



Interest Rate Swaps



Interest Rate Swaps



Assumptions

Starting point — some generalisable for flexibility

1. No consideration of credit risk
2. Swap has fixed lifetime
3. All annuities have annual payments received at same time
4. Fixed cost of capital over lifetime
5. All annuitants have undergone underwriting evaluation
6. APV calculations require a specific lifetable
7. All annuities have a lifetime at least as long as the swap lifetime

Swap Annuity Portfolio

Random portfolio of annuities:

```
##       customerid age MUR   amount
##   1:  C97326460  39 350 $100,000
##   2:  C99971880  30 175  $80,000
##   3:  C65134119  34 225  $50,000
##   4:  C35313144  33 200  $90,000
##   5:  C17550793  45 200  $30,000
##  ---
## 196:  C37440555  40 200  $50,000
## 197:  C60347677  42 200  $60,000
## 198:  C93383952  45 200  $70,000
## 199:  C08447895  35 150  $60,000
## 200:  C64003360  41 325  $90,000
```

Simulation Approach

Simulation example: 10 simulations of 5 years

1 – annuitant still alive

0 – annuitant has deceased

```
##       sim1 sim2 sim3 sim4 sim5 sim6 sim7 sim8 sim9 sim10
## year1    1    1    1    1    1    1    1    1    1     1
## year2    1    1    0    1    1    1    1    1    1     1
## year3    1    1    0    1    1    1    1    1    0     1
## year4    1    1    0    1    0    1    1    1    0     1
## year5    1    1    0    1    0    1    1    1    0     1
```

Lifetable Issues

- For actuarial discounts of cashflows, swap needs an agreed lifetable
- Currently using the National Vital Statistics Reports 2010
- If same lifetable used for valuation and only guaranteeing discounted payments, expected value of swap is 0.
- MonteCarlo simulation can use different lifetables
- Could extend this to have different lifetable for each annuitant

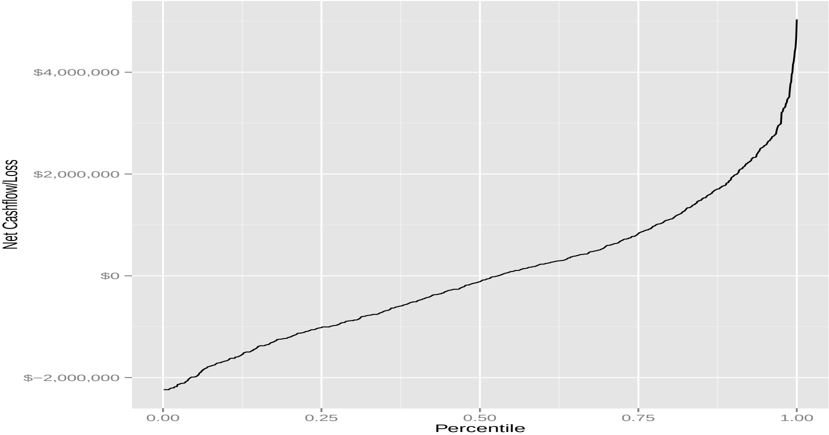
Running the Simulation

Run the calculation:

```
mortport.dt  <- fread("mortswap_portfolio.csv");
lifetable.dt <- fread("lifetable.csv");

n.sim <- 1000;

mortswap.value.sim <- calculate.mortality.swap.price(mortport.dt,
                                                     lifetable           = lifetable.dt,
                                                     hedge.apv.cashflows = TRUE,
                                                     interest.rate       = 0.05,
                                                     years.in.force      = 20,
                                                     n.sim               = n.sim,
                                                     return.all.data     = FALSE);

print(dollar_format(largest_with_cents = 1e8)(mortswap.value.sim[1:20]));


##  [1] "$-283,314.84"   "$-60,789.31"    "$649,297.44"    "$-1,497,627.24"
##  [5] "$-2,237,366.32" "$-975,230.15"   "$266,167.89"    "$928,428.12"
##  [9] "$1,533,972.60"  "$-179,043.92"   "$244,749.59"    "$-1,232,653.58"
## [13] "$112,477.15"    "$3,482,586.54"  "$-1,374,876.09" "$161,213.93"
## [17] "$1,673,164.92"  "$425,828.72"    "$1,707,908.76"  "$2,226,819.71"
```

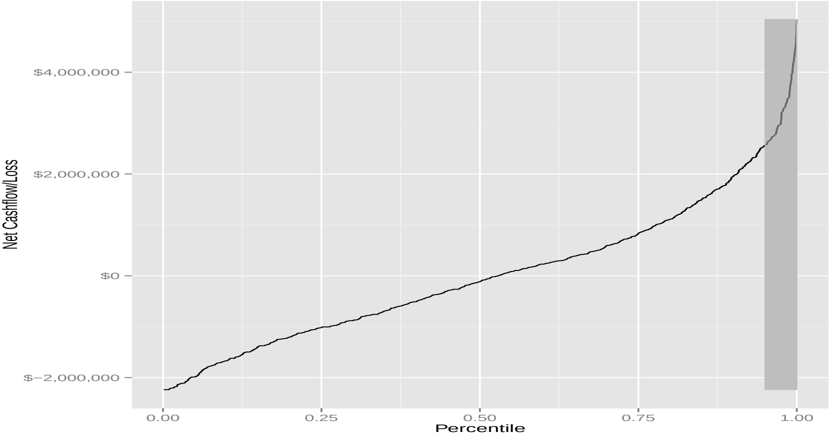
Viewing the Output

Simulated cashflows
(excludes initial premium):



Viewing the Output

Simulated cashflows
(excludes initial premium):



Pricing Tail Risk

How to price the tail risk?

Michael Lewis *"In Nature's Casino"* – NYT Aug 2007
http://www.nytimes.com/2007/08/26/magazine/26neworleans-t.html?pagewanted=all

Consensus around $4\times$ expected loss

Need to calculate tail averages

Pricing Tail Risk

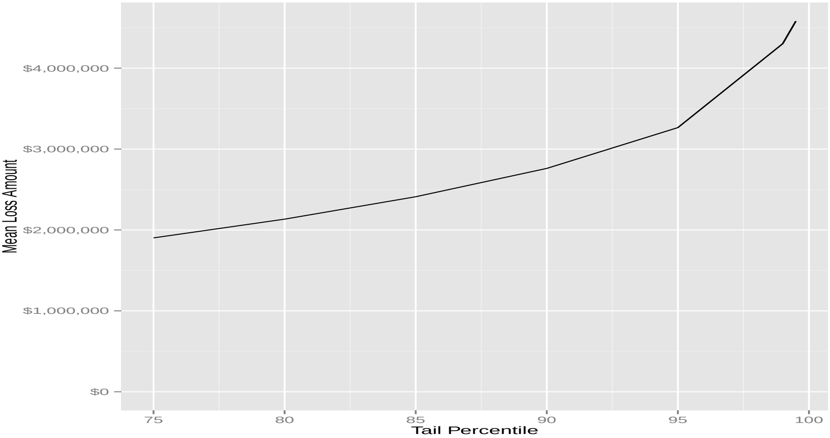
```
calculate.quantile.averages <- function(x, probs) {
    return(sapply(quantile(x, probs), function(qtl) mean(x[x >= qtl])));
}

calculate.quantile.averages(0:10, 0.8);
```

```
## 80%
##   9
```

```
calculate.quantile.averages(0:100, 0.8)
```

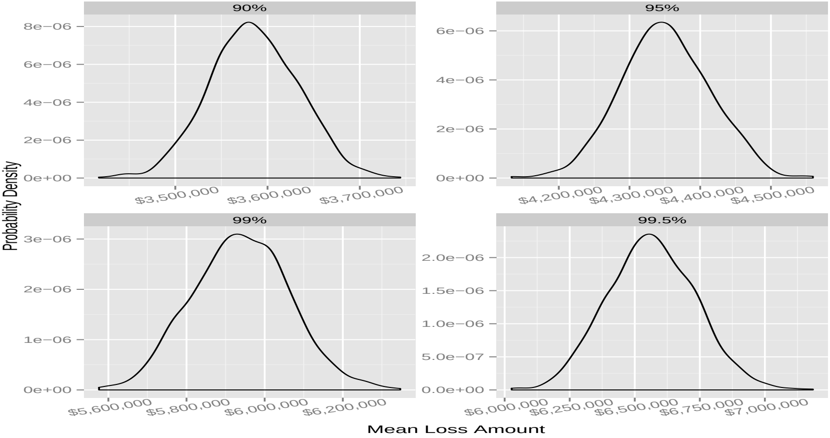
```
## 80%
##  90
```

Pricing Tail Risk



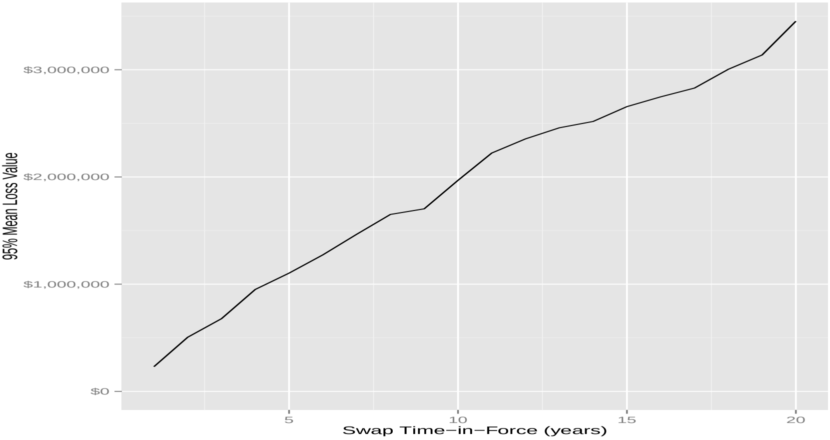
Pricing Tail Risk

Ensemble of 1,000 valuations of 10,000 iterations:



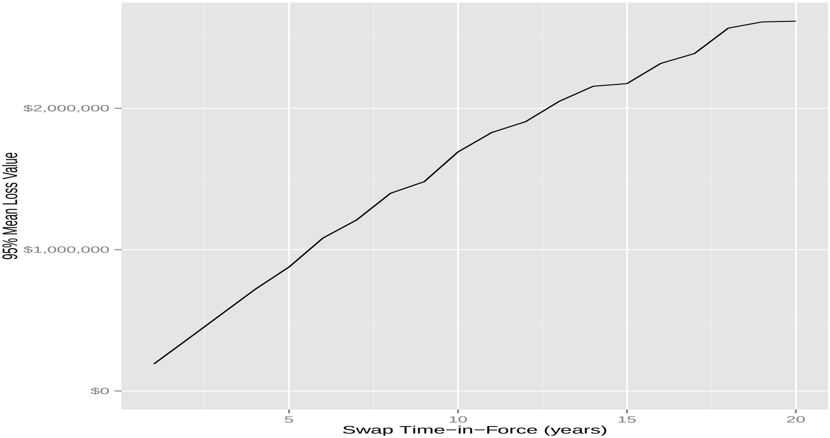
Time-in-Force Dependency

Scaling of 95% mean with time-in-force of swap:



Time-in-Force Dependency

Scaling of 90% mean with time-in-force of swap:

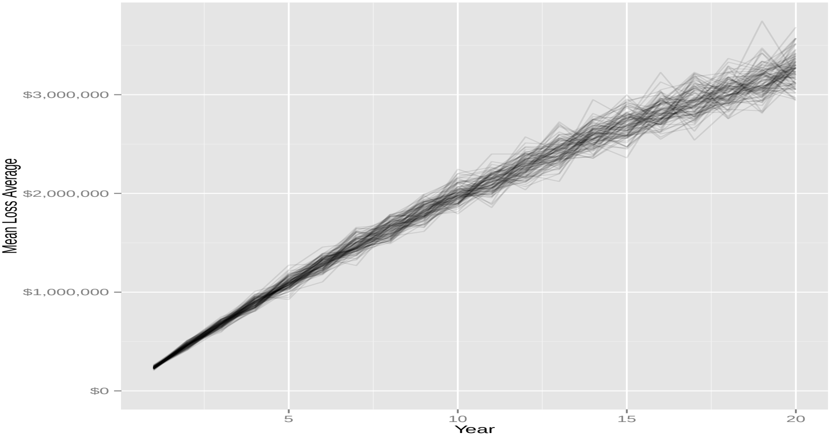


Time-in-Force Dependency (Ensemble)

Scaling of 95% mean with time-in-force of swap:

```
times.in.force <- 1:20;
n.sim          <- 1000;
n.ens          <- 10;

calculate.tif <- function(tif) {
    mortswap.value.sim <- calculate.mortality.swap.price(mortport.dt,
                                                          lifetable           = lifetable.dt,
                                                          hedge.apv.cashflows = TRUE,
                                                          interest.rate       = 0.05,
                                                          years.in.force      = tif,
                                                          n.sim               = n.sim,
                                                          return.all.data     = FALSE);


    iterval <- calculate.quantile.averages(mortswap.value.sim, 0.95);
}

tif.ensemble <- sapply(times.in.force, function(iter.tif) {
    replicate(n.ens, calculate.tif(iter.tif))
});
```

Time-in-Force Dependency (Ensemble)



Summary

R package: `mcmortswap`

`https://bitbucket.org/appliedai/mcmortswap`

Email: michael.cooney@applied.ai

Slides available on github:

`https://github.com/kaybenleroll/dublin_r_workshops`