

# Spouse model implemented with K13

Andreas Slättelid

## Spouse model

```
library(tidyverse)
library(scales)
library(data.table)
library(dtplyr) #want to test if dt speeds up the df
```

We will be considering a spouse model, the insured will get a yearly pension  $P$ , in case one of the insured dies. Let insured number one have age  $x$  and age of insured number two have age  $y$ .

The state space will be:  $S = \{0, 1, 2, 3\}$

```
x <- 40      #age of insured numb.1
y <- 30      #age of insured numb.2
T <- 80      #length of contract
P <- 50000   #pension
r <- 0.03    #interest rate
```

The policy functions will look like:

$$a_0^{Pre}(n) = \begin{cases} -\pi & , n = 0, \dots, T-1 \\ 0 & , else \end{cases}$$
$$a_1^{Pre}(n) = a_2^{Pre}(n) = \begin{cases} P & , n = 0, \dots, T-1 \\ 0 & , else \end{cases}$$

The formula for the reserve in state 0 will look like:

$$V_0^+(t, A) = -\pi \sum_{n=t}^{T-1} \frac{v(n)}{v(t)} p_{00}^{(x,y)}(t, n) + P \left( \sum_{n=t}^{T-1} \frac{v(n)}{v(t)} p_{01}^{(x,y)}(t, n) + \sum_{n=t}^{T-1} \frac{v(n)}{v(t)} p_{02}^{(x,y)}(t, n) \right)$$

We assume the individuals to have independent lives, and their sample-spaces are  $S_{(i)} = \{*, \dagger\}$ , this gives:

$$p_{00}^{(x,y)}(t, n) = p_{**}^x(t, n) p_{**}^y(t, n)$$
$$p_{01}^{(x,y)}(t, n) = p_{*\dagger}^x(t, n) p_{**}^y(t, n)$$
$$p_{02}^{(x,y)}(t, n) = p_{**}^x(t, n) p_{*\dagger}^y(t, n)$$

Here I have taken state 1 to be the state where individual aged  $x$  dies. Let's assume that we deal with a couple where person aged  $x$  is a man, and person aged  $y$  is a woman.

```

#The weights given by finanstilssynet
w <- function(x, G){
  #male: G = "M"
  #female: G = "F"
  #x: age in calendar year t,
  if (G == "M"){
    return (min(2.671548-0.172480*x + 0.001485*x**2, 0))
  }
  else {
    return(min(1.287968-0.101090*x+ 0.000814*x**2,0))
  }
}

mu_kol_2013 <- function(x, G){
  #male
  if (G == "M"){
    return((0.241752+0.004536*10**(0.051*x))/1000)
  }
  #female
  else {
    return((0.085411+0.003114*10**(0.051*x))/1000)
  }
}

#turning mu into a function of u, so that we can use integrate
mu <- function(u, x, G, Y = 2022){
  return(mu_kol_2013(x+u, G)*(1 + w(x+u, G)/100)^(Y+u-2013))
}

p_surv <- function(x, G, Y, t, s){

  if (t == s){
    return(1)
  }

  f <- Vectorize(mu)
  integral <- integrate(f, lower = t, upper = s, x=x, Y=Y, G=G)$value

  ans <- exp((-1)*integral)
  return(ans)
}

v <- function(t){
  return(exp(-(r*t)))
}

```

Again, by the equivalence principle we determine the yearly premium  $\pi$  such that  $V_0^+(0, A) = 0$ , this gives:

$$\pi = \frac{P \left( \sum_{n=0}^{T-1} v(n) [p_{01}^{(x,y)}(0, n) + p_{02}^{(x,y)}(0, n)] \right)}{\sum_{n=0}^{T-1} v(n) p_{00}^{(x,y)}(0, n)}$$

```

#both survive:
p_00 <- function(t, n){
  p_surv(x, G="M", Y=2022, t=t, s = n)*p_surv(y, G="F", Y=2022, t=t, s=n)
}

#man dies, woman survive:
p_01 <- function(t,n){
  (1 - p_surv(x, G="M", Y = 2022, t=t, s = n ))*p_surv(y, G = "F", Y=2022, t=t, s = n)
}

#man survive, woman die:
p_02 <- function(t,n){
  p_surv(x, G="M", Y=2022, t=t, s = n )*(1 - p_surv(y, G="F", Y=2022, t=t, s = n))
}

#wife remains alive:
p_11 <- function(t,n){
  p_surv(y, G = "F", Y=2022, t=t, s = n)
}

#man remains alive:
p_22 <- function(t,n){
  p_surv(y, G = "M", Y=2022, t=t, s = n)
}

upper_summand <- function(n){
  prob <- p_01(0,n) + p_02(0,n)
  return(v(n)*prob)
}

lower_summand <- function(n){
  prob <- p_00(0,n)
  return(v(n)*prob)
}

(yearly_premium <- P*sum(map_dbl(0:(T-1), upper_summand))/sum(map_dbl(0:(T-1), lower_summand)))

## [1] 7618.899

```

```

#reserve in state 0:
V_0 <- function(t){

  summand_1 <- function(t, n){
    (v(n)/v(t))*p_00(t, n)
  }

  summand_2 <- function(t, n){
    (v(n)/v(t))*(p_01(t,n) + p_02(t,n))
  }

  ans <- (-1)*yearly_premium*sum(map_dbl(t:(T-1),summand_1, t = t)) + P*sum(
    map_dbl(t:(T-1), summand_2, t = t))

  return(ans)
}

#reseve in state 1:
V_1 <- function(t){

  summand <- function(t, n){
    (v(n)/v(t))*p_11(t,n)
  }

  ans <- P*sum(map_dbl(t:(T-1), summand, t = t))

  return(ans)
}

#reserve in state 2:
V_2 <- function(t){

  summand <- function(t,n){
    (v(n)/v(t))*p_22(t,n)
  }

  ans <- P*sum(map_dbl(t:(T-1), summand, t = t))

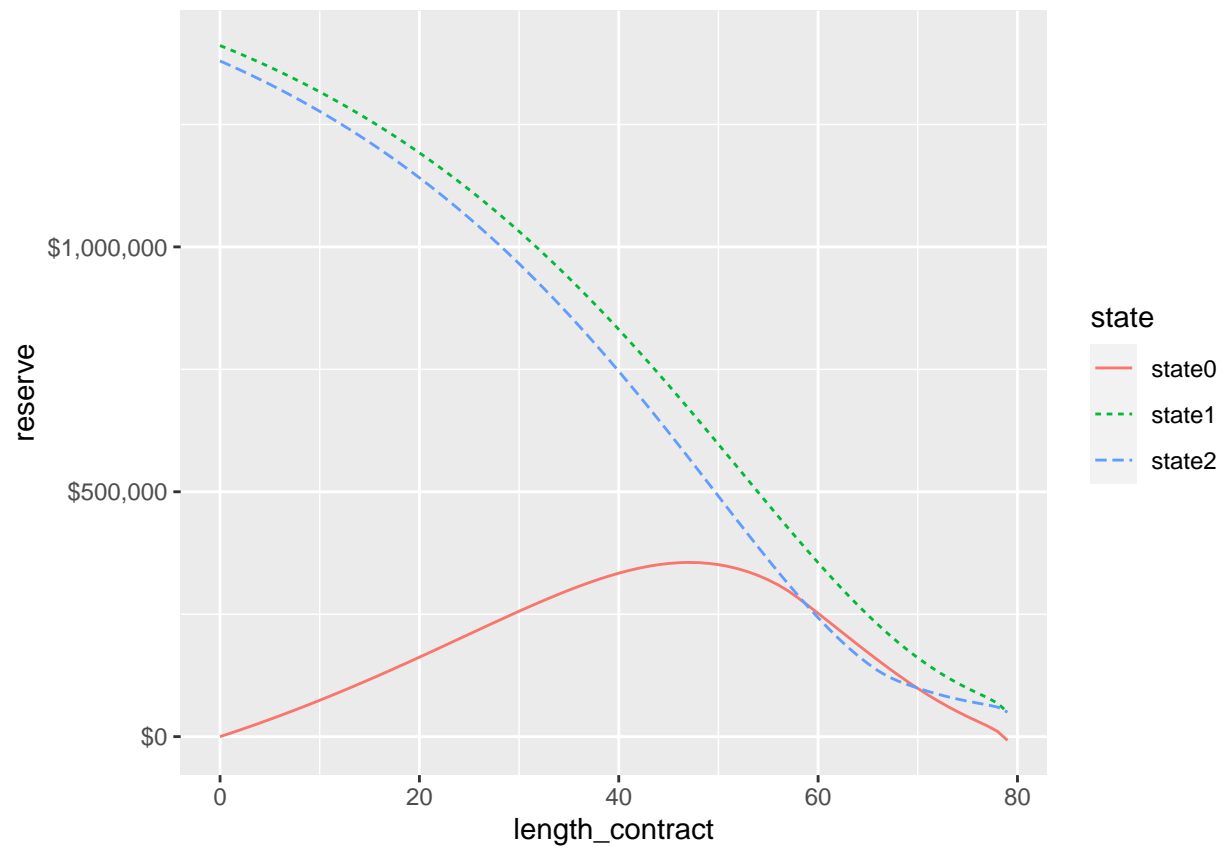
  return(ans)
}

```

```
length_contract <- 0:(T-1)
state0 <- map_dbl(length_contract, V_0)

df <- data.frame(length_contract, state0) %>%
  mutate(state1 = map_dbl(length_contract, V_1)) %>%
  mutate(state2 = map_dbl(length_contract, V_2)) %>%
  pivot_longer(!length_contract, names_to = "state", values_to = "reserve")

df %>%
  ggplot(aes(x = length_contract, y = reserve)) +
  geom_line(aes(color = state, linetype = state)) +
  scale_y_continuous(labels = dollar)
```



Some motivation for speeding up the shiny app:

```
dt_lazy <- dtplyr::lazy_dt(data.frame(length_contract, state0))
```

```
dt_query <- dt_lazy %>%  
  mutate(state1 = map_dbl(length_contract, V_1)) %>%  
  mutate(state2 = map_dbl(length_contract, V_2)) %>%  
  dplyr::show_query()
```

```
dt_query
```

```
## copy('_DT1')[, `:=`(state1 = map_dbl(length_contract, ..V_1))][,  
##      `:=`(state2 = map_dbl(length_contract, ..V_2))]
```

```
mydt <- as.data.table(data.frame(length_contract, state0))
```

```
#the query from dt_query:
```

```
dt <- mydt[, `:=`(state1 = map_dbl(length_contract, ..V_1))][,  
  `:=`(state2 = map_dbl(length_contract, ..V_2))]
```

```
head(dt, 10)
```

```
##      length_contract    state0 state1 state2  
## 1:                0      0.000 1411117 1379624  
## 2:                1  6732.476 1402820 1370612  
## 3:                2 13619.086 1394282 1361336  
## 4:                3 20659.997 1385497 1351788  
## 5:                4 27855.192 1376459 1341964  
## 6:                5 35204.267 1367162 1331855  
## 7:                6 42706.138 1357599 1321454  
## 8:                7 50360.311 1347763 1310755  
## 9:                8 58164.547 1337649 1299750  
## 10:               9 66116.966 1327249 1288431
```