Practical Health State Transition Model 2B - Solutions

X. Pouwels

2021-09-27

```
rm(list = ls())
options(scipen = 999)
library(knitr)
```

The method described in this practical is more extensively described in Alarid-Escudero et al. (2020).

Questions and answers

Question 1 & 2 are already performed in the `Assignment_2B_start.R` file, please read the instruction and complete the asssignement. 1. In the previous exercise we have used one average mortality rate for all ages. Since this is not realistic, we will now make the mortality rate age-dependent.
1.a. Once loaded, have a look at the inputs. You can see that there is a dataframe `df_mort` added to the model inputs. This dataframe contains the probability of death of individuals for each age. Inspect this dataframe, it contains 2 columns: `Age` = age of an individual (between 20 and 115) and `p_mort` = mortality probability for each age (the data in this assignment is fake, in reality, we would use probabilities based on national statistics estimates!).
1.b. To include these age-dependent transition probabilities in the model, we have to use 3-dimentional (3D) arrays instead of fixed transition matrices. The use of these 3D arrays allow to select a different transition matrix (containing different transition probabilities) for each cycle of the model. Arrays are defined through the `array()` function.
1.b.i. Define empty arrays for the "No aspirin" group (called `a_tp_comp`) with `n_hs` columns, `n_hs` rows, and a depth of `n_cycles`. Inspect this array. As you can see, there are now 10 empty transition matrix when you inspect that object.
1.b.ii. Define the start age of the cohort (`n_start_age`), which is 45 years old. 1.b.iii. Define a vector (of 10 values) of mortality probabilities containing the transition probabilities from `n_start_age` to `n_start_age` + 9.
1.b.iv. Fill in the array with the (time-dependent) transition probabilities, using the transition probabilities from assignment 2A, and the age-dependent mortality probability `v_p_mort` for the "No aspirin" group.
NOTE: elements in a array are called by the number of dimension of the array. Thus, a single element of a 3D array is called via a x, y, and z dimension. If only the x and y coordinates are provided, the vector of elements with the coordinates x and y is returned. In this case, the third dimension (the 'z') represents the cycle number while the rows and columns represent the same as in the transition matrix (transition from - to).

```
#----------------------------------#
#### 0. Define model parameters ####
#----------------------------------#

# Setting parameters
n_cycles <- 10 # number of cycles
```

```r
r_d_effects <- 0.015 # annual discount rate, health effects
r_d_costs <- 0.04 # annual discount rate, costs
v_names_hs <- c("Well", "Post-minor_stroke", "Post-major_stroke", "Post-MI", "Death_stroke", "Death_MI"
n_hs <- length(v_names_hs) # number of health states
n_ind <- 100000 # number of individuals to simulate
v_start_hs <- c(n_ind, 0, 0, 0, 0, 0, 0) # vector of starting position in the model

# Input parameters

## Rates & probabilities
r_fatal_mi  <- 0.25 # rate fatal MI
r_fatal_stroke  <- 0.3 # rate fatal stroke
r_inc_mi <- 400 / 100000 # yearly incidence rate MI
r_inc_stroke <- 50 / 100000 # yearly incidence rate stroke
r_mort <- 650 / 100000 # yearly rate of death

r_mort_age_dependent <- r_mort_age_dependent <-  0.95^(c(115:20)-19) # mortality rate (age dependent) -

# Determine mortality probability for each age
df_mort <- data.frame(cbind(age = c(20:115),
                            p_mort = 1 - exp(-r_mort_age_dependent)
                            )
)
df_mort[nrow(df_mort), 2] <- 1 # Assumption that everybody dies at age 115
#plot(df_mort[, 1], df_mort[,2], type = 'l')

## Treatment effectiveness
eff_mi  <- 0.6 # Treatment effectiveness of Aspirin on the probability of experiencing a MI
eff_stroke  <- 1.2 # Treatment effectiveness of Aspirin on the probability of experiencing a stroke

## Utility values
u_healthy   <- 1 # utility value health state: Well
u_post_mi   <- 0.85 # utility value health state: Post-MI
u_post_minor_stroke <- 0.75 # utility value health state: Post-minor stroke
u_post_major_stroke <- 0.5 # utility value health state: Post-major stroke
u_aspirin_use   <- 0.999 # utility value health state: Well when using aspiring

## Costs
c_aspirin_use   <- 100 # yearly costs of using aspiring
c_post_mi   <- 8000 # yearly costs after having experienced a NON-FATAL MI
c_post_minor_stroke <- 2000 # yearly costs after having experienced a NON-FATAL minor stroke
c_post_major_stroke <- 20000 # yearly costs after having experienced a NON_FATAL major stroke

# Parameters which needed to be calculated in the previous assignment
p_post_major_stroke <- 235 / 1000 # probability to transit to "Post-major stroke" after a NON-FATAL str
p_post_minor_stroke <- 1 - p_post_major_stroke # probability to transit to "Post-minor stroke" after a

#-------------------------------------------#
#### 1. Define 3D array "No aspirin" group ####
#-------------------------------------------#

# Define start age cohort
n_start_age <- 45
```

```r
# Create a vector containing the mortality probability values for the age 'n_start_age' to 'n_start_age
v_p_mort <- df_mort[c(which(df_mort$age == n_start_age):
                        which(df_mort$age == n_start_age + 9)
                      ), "p_mort"]

# Initialise array's
a_tp_comp <- array(0, dim = c(n_hs, n_hs, n_cycles),
                              dimnames = list(v_names_hs, v_names_hs, 1:n_cycles)
                              )

# Fill in the arrays using the background mortality for each age
a_tp_comp["Well", "Well", ]   <- 1 - v_p_mort - r_inc_mi - r_inc_stroke  # EXAMPLE! Calculate the remai
##Notice, that only using the first 2 elements of the array will fill these transition probabilities fo

a_tp_comp["Well", "Post-minor_stroke", ] <- r_inc_stroke * (1 - r_fatal_stroke) * p_post_minor_stroke
a_tp_comp["Well", "Post-major_stroke", ] <- r_inc_stroke * (1 - r_fatal_stroke) * p_post_major_stroke
a_tp_comp["Well", "Post-MI", ] <- r_inc_mi * (1 - r_fatal_mi)
a_tp_comp["Well", "Death_stroke", ] <- r_inc_stroke * r_fatal_stroke
a_tp_comp["Well", "Death_MI", ] <- r_inc_mi * r_fatal_mi
a_tp_comp["Well", "Death_other", ] <- v_p_mort

a_tp_comp["Post-minor_stroke", "Post-minor_stroke", ] <- 1 - v_p_mort
a_tp_comp["Post-minor_stroke", "Death_other", ] <- v_p_mort

a_tp_comp["Post-major_stroke", "Post-major_stroke", ] <- 1 - v_p_mort
a_tp_comp["Post-major_stroke", "Death_other", ] <- v_p_mort

a_tp_comp["Post-MI", "Post-MI", ] <- 1 - v_p_mort
a_tp_comp["Post-MI", "Death_other", ] <- v_p_mort

a_tp_comp["Death_stroke", "Death_stroke",] <- 1
a_tp_comp["Death_MI", "Death_MI",] <- 1
a_tp_comp["Death_other", "Death_other",] <- 1

# Inspect whether the sum of all TP's for all health states over all cycles = 1
m_check_comp <- matrix(NA,
                  nrow = n_cycles,
                  ncol = n_hs)

for (i in 1:n_cycles){
  v_res <- round(rowSums(a_tp_comp[ , ,i]), 5) == 1
  m_check_comp[i,] <- paste(v_res)

}

kable(a_tp_comp[, , 1],
      caption = "First element of the 3D transition array - 'No aspirin' group")
```

Table 1: First element of the 3D transition array - 'No aspirin' group

	Well	Post-minor_stroke	Post-major_stroke	Post-MI	Death_stroke	Death_MI	Death_other
Well	0.9696359	0.0002677	0.0000822	0.0030000	0.00015	0.001	0.0258641
Post-minor_stroke	0.0000000	0.9741359	0.0000000	0.0000000	0.00000	0.000	0.0258641
Post-major_stroke	0.0000000	0.0000000	0.9741359	0.0000000	0.00000	0.000	0.0258641
Post-MI	0.0000000	0.0000000	0.0000000	0.9741359	0.00000	0.000	0.0258641
Death_stroke	0.0000000	0.0000000	0.0000000	0.0000000	1.00000	0.000	0.0000000
Death_MI	0.0000000	0.0000000	0.0000000	0.0000000	0.00000	1.000	0.0000000
Death_other	0.0000000	0.0000000	0.0000000	0.0000000	0.00000	0.000	1.0000000

```
kable(a_tp_comp[, , 2],
      caption = "Second element of the 3D transition array - 'No aspirin' group")
```

Table 2: Second element of the 3D transition array - 'No aspirin' group

	Well	Post-minor_stroke	Post-major_stroke	Post-MI	Death_stroke	Death_MI	Death_other
Well	0.9682933	0.0002677	0.0000822	0.0030000	0.00015	0.001	0.0272067
Post-minor_stroke	0.0000000	0.9727933	0.0000000	0.0000000	0.00000	0.000	0.0272067
Post-major_stroke	0.0000000	0.0000000	0.9727933	0.0000000	0.00000	0.000	0.0272067
Post-MI	0.0000000	0.0000000	0.0000000	0.9727933	0.00000	0.000	0.0272067
Death_stroke	0.0000000	0.0000000	0.0000000	0.0000000	1.00000	0.000	0.0000000
Death_MI	0.0000000	0.0000000	0.0000000	0.0000000	0.00000	1.000	0.0000000
Death_other	0.0000000	0.0000000	0.0000000	0.0000000	0.00000	0.000	1.0000000

```
kable(a_tp_comp[, , 10],
      caption = "Tenth element of the 3D transition array - 'No aspirin' group")
```

Table 3: Tenth element of the 3D transition array - 'No aspirin' group

	Well	Post-minor_stroke	Post-major_stroke	Post-MI	Death_stroke	Death_MI	Death_other
Well	0.9547745	0.0002677	0.0000822	0.0030000	0.00015	0.001	0.0407255
Post-minor_stroke	0.0000000	0.9592745	0.0000000	0.0000000	0.00000	0.000	0.0407255
Post-major_stroke	0.0000000	0.0000000	0.9592745	0.0000000	0.00000	0.000	0.0407255
Post-MI	0.0000000	0.0000000	0.0000000	0.9592745	0.00000	0.000	0.0407255
Death_stroke	0.0000000	0.0000000	0.0000000	0.0000000	1.00000	0.000	0.0000000
Death_MI	0.0000000	0.0000000	0.0000000	0.0000000	0.00000	1.000	0.0000000
Death_other	0.0000000	0.0000000	0.0000000	0.0000000	0.00000	0.000	1.0000000

```r
kable(m_check_comp,
      caption = "Check whether sum of transition probabilities = 1"
      )# visualise the check
```

```
## Warning in kable_pipe(x = structure(c("TRUE", "TRUE", "TRUE", "TRUE", "TRUE", :
## The table should have a header (column names)
```

Table 4: Check whether sum of transition probabilities = 1

TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE
TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE
TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE
TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE
TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE
TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE
TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE
TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE
TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE
TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE

2. Fill the cohort simulation for the "No aspirin" group using the 3D array. To do so, create a matrix to store the cohort simulation (`m_hs_comp`) of 7 columns (number of health states) and 11 rows (`n_cycles` + 1), define the start position of individuals (all in "well", `v_start_hs`), and perform the matrix multiplication using the 3D array to fill the `m_hs_comp` matrix. You have to loop over the elements of the array to ensure that the time dependent mortality probability are used.

```r
#----------------------------------------------------------------#
#### 2. Fill the cohort simulation for the "No aspirin" group ####
#----------------------------------------------------------------#

# Create a matrix to store the cohort simulation ('m_hs_comp')
m_hs_comp <- matrix(0,
                    ncol = length(v_names_hs),
                    nrow = n_cycles + 1,
                    dimnames = list(c(0:n_cycles),
                                    v_names_hs)
                    )



# Define the start position of individuals (all in "well")
m_hs_comp[1,] <- v_start_hs

# Perform the matrix multiplication using the 3D array.
for(cycle in 1:n_cycles){
  m_hs_comp[cycle + 1,] <- m_hs_comp[cycle,] %*% a_tp_comp[, , cycle]

}
```

3. Fill the 3D array for the "Aspirin" group.
   3.a. Make a copy of `a_tp_comp` and call it `a_tp_int`. This will be the transition array we will use for

the "Aspirin" group.

3.b. Aspirin has an effect on the probability of experiencing an MI (object `eff_mi`) and stroke (object `eff_stroke`). Use these two input parameters to adjust the probabilities to experience these (non-)fatal cardiovascular events in the `a_tp_int` array. Again you may use the rates as probabilities when calculating the transitions. Assume that once individuals had an event they will no longer use aspirin.

```
# Make a copy of 'a_tp_comp' and call it 'a_tp_int'
a_tp_int <- a_tp_comp

# Modify 'a_tp_int' since Aspirin has an
a_tp_int["Well", "Well", ]   <- 1 - v_p_mort - r_inc_mi * eff_mi - r_inc_stroke * eff_stroke
a_tp_int["Well", "Post-minor_stroke", ] <- r_inc_stroke * eff_stroke * (1 - r_fatal_stroke) * p_post_mi
a_tp_int["Well", "Post-major_stroke", ] <- r_inc_stroke * eff_stroke * (1 - r_fatal_stroke) * p_post_ma
a_tp_int["Well", "Post-MI", ] <- r_inc_mi * eff_mi * (1 - r_fatal_mi)
a_tp_int["Well", "Death_stroke", ] <- r_inc_stroke * eff_stroke * r_fatal_stroke
a_tp_int["Well", "Death_MI", ] <- r_inc_mi * eff_mi * r_fatal_mi

# Inspect whether the sum of all TP's for all health states over all cycles = 1
m_check_int <- matrix(NA,
                      nrow = n_cycles,
                      ncol = n_hs)

for (i in 1:n_cycles){
  v_res <- round(rowSums(a_tp_int[ , ,i]), 5) == 1
  m_check_int[i,] <- paste(v_res)

}
kable(m_check_int,
      caption = "Check sum of transition probabilities for each health state = 1") # visualise the chec
```

```
## Warning in kable_pipe(x = structure(c("TRUE", "TRUE", "TRUE", "TRUE", "TRUE", :
## The table should have a header (column names)
```

Table 5: Check sum of transition probabilities for each health state = 1

TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE
TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE
TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE
TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE
TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE
TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE
TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE
TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE
TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE
TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE

3.b.i. What is the probability of an individual on aspirin treatment to have a myocardial infarction?

```
p_mi <-  r_inc_mi * eff_mi
```

Answer: The probability to have a myocardial infarction when using aspirin is 0 %.

3.b.ii. And what is the probability of an individual on aspirin treatment to have a stroke?

```
p_stroke <-  r_inc_stroke * eff_stroke
```

Answer: The probability to have a myocardial infarction when using aspirin is 0 %.
4. Fill the cohort simulation for the "No aspirin" group (`m_hs_int`) using the 3D array `m_tp_int` and the same method has shown in step 2.

```
#-------------------------------------------------------------#
#### 4. Fill the cohort simulation for the "Aspirin" group ####
#-------------------------------------------------------------#

# Create a matrix to store the cohort simulation (‘m_hs_int‘)
m_hs_int <- matrix(0,
                   ncol = length(v_names_hs),
                   nrow = n_cycles + 1,
                   dimnames = list(c(0:n_cycles),
                                   v_names_hs)
)



# Define the start position of individuals (all in "well")
m_hs_int[1,] <- v_start_hs

# Perform the matrix multiplication using the 3D array.
for(cycle in 1:n_cycles){
  m_hs_int[cycle + 1,] <- m_hs_int[cycle,] %*% a_tp_int[, , cycle]

}
```

5. Now that we know the life course of the hypothetical individuals we can calculate the (undiscounted) costs and effects over time. For convenience, account for state membership at the end of the year (i.e., not all 100,000 individuals are considered to be "Well" at the start of year 1 and you do not have to use a half-cycle correction). Practically, it means that you do not account for the state membership at cycle 0, the starting position.
   5.a. Calculate the life years of both strategies, i.e. the number of individuals alive each year, and the cumulative life years over the 10 years of the model. To do so, define two vectors (`v_ly_comp` & `v_ly_int`) of 7 values which determine the number of life years gained by an individual in each health state during a single cycle. These vectors are called the "rewards" vectors, and the vector of rewards should be ordered as the health states in the `m_hs_comp` & `m_hs_int` cohort simulation. For health state were individuals are alive, the reward should thus be 1, and 0 for the "Death" health state. Use matrix multiplication to multiply these health state rewards by the membership of individuals over the cycles (from row 2 onwards!). Store these results in vectors called `v_t_ly_comp` & `v_t_ly_int`, calculate the cumulative number of life years over the cycles (`v_cum_ly_comp` & `v_cum_ly_int`, using the cumsum () function) and calculate the total number of life years for each strategy as the sum of these vectors (`n_t_ly_comp` & `n_t_ly_int`). The calculations for the life years are provided in the `Assignment_2B_start.R` file, use this example for performing 4.b. and 4.c.
   5.b. Using the utilities defined in the `Assignment_2B_start.R` file, calculate the total quality adjusted life-years (QALYs) in each year for the 100,000 hypothetical individuals in each group. Use the same approach as for the life years calculations, expect that the state rewards are different. Calculate also the cumulative and total QALY gained.
   Assmuptions: Individuals in the "Well" health state have a utility value equal to`u_healthy` in the

"No Aspirin" group, while in the "Aspirin" group, they have the utility associated with use of Aspirin. Assume that after a MI or stroke, individuals stop with aspirin. Individuals in the "Death" health states do not accrue QALYs. 5.c. Using the annual costs for health state defined in the `Assignment_start.R` file, calculate the total costs in each year for the 100,000 hypothetical individuals. Use the same approach as for the life years calculation, expect that the state rewards are different. Calculate also the cumulative and total costs.

Assmuptions: Individuals in the "Well" health state do not incur any costs in the "No aspirin" group. However, in the "Apsirin" group, individuals in the "Well" health state incur the costs associated with aspirin use. Assume that after a MI or stroke, individuals stop with aspirin.Individuals in the "Death" health states do not incur any costs.

5.d. Calculate the mean outcomes (life years, QALYs, costs) per individual for each strategy. Calculate the incremental QALYs and costs of the "Aspirin" versus "No aspirin" group and calculate the incremental cost-effectiveness ratio (incremental costs/incremental QALYs).

```r
#-------------------------------------------------------------------------------------------------#
#### 5. Calculate life years, quality-adjusted life years, and costs for both strategy ####
#-------------------------------------------------------------------------------------------------#

# Life years: EXAMPLE! DO THE SAME FOR QALY's AND COSTS
## Determine the number of life year won by 1 individual during 1 cycle
v_ly_comp <- c("Well" = 1,
               "Post-minor_stroke" = 1,
               "Post-major_stroke" = 1,
               "Post-MI" = 1,
               "Death_stroke" = 0,
               "Death_MI" = 0,
               "Death_other" = 0)
v_ly_int <- c("Well" = 1,
               "Post-minor_stroke" = 1,
               "Post-major_stroke" = 1,
               "Post-MI" = 1,
               "Death_stroke" = 0,
               "Death_MI" = 0,
               "Death_other" = 0)

## Determine the number of life year gained over the cycles (reward at the end of the cycle!)
v_t_ly_comp <- m_hs_comp[2:nrow(m_hs_comp),] %*% v_ly_comp
v_t_ly_int  <- m_hs_int[2:nrow(m_hs_int),] %*% v_ly_int

## Determine the cumulative number of life year gained over the cycles (reward at the end of the cycle!)
v_cum_ly_comp <- cumsum(v_t_ly_comp)
v_cum_ly_int  <- cumsum(v_t_ly_int)

## Determine the total number of life year gained (sum of all cycles; reward at the end of the cycle!)
n_t_ly_comp <- sum(v_t_ly_comp)
n_t_ly_int <- sum(v_t_ly_int)

# QALY's
## Determine the number of QALYs won by 1 individual during 1 cycle
v_qaly_comp <- c("Well" = u_healthy,
                 "Post-minor_stroke" = u_post_minor_stroke,
                 "Post-major_stroke" = u_post_major_stroke,
                 "Post-MI" = u_post_mi,
                 "Death_stroke" = 0,
```

```
                 "Death_MI" = 0,
                 "Death_other" = 0)

v_qaly_int <- c("Well" = u_aspirin_use,
                "Post-minor_stroke" = u_post_minor_stroke,
                "Post-major_stroke" = u_post_major_stroke,
                "Post-MI" = u_post_mi,
                "Death_stroke" = 0,
                "Death_MI" = 0,
                "Death_other" = 0)

## Determine the number of QALYs gained over the cycles (reward at the end of the cycle!)
v_t_qaly_comp <- m_hs_comp[2:nrow(m_hs_comp),] %*% v_qaly_comp
v_t_qaly_int  <- m_hs_int[2:nrow(m_hs_int),] %*% v_qaly_int

## Determine the cumulative number of QALYsr gained over the cycles (reward at the end of the cycle!)
v_cum_qaly_comp <- cumsum(v_t_qaly_comp)
v_cum_qaly_int <- cumsum(v_t_qaly_int)

## Determine the total number of QALYs gained (sum of all cycles; reward at the end of the cycle!)
n_t_qaly_comp <- sum(v_t_qaly_comp)
n_t_qaly_int  <- sum(v_t_qaly_int)

# Costs
## Determine the costs accrued by 1 individual during 1 cycle
v_c_comp <- c("Well" = 0,
              "Post-minor_stroke" = c_post_minor_stroke,
              "Post-major_stroke" = c_post_major_stroke,
              "Post-MI" = c_post_mi,
              "Death_stroke" = 0,
              "Death_MI" = 0,
              "Death_other" = 0)

v_c_int <- c("Well" = c_aspirin_use,
             "Post-minor_stroke" = c_post_minor_stroke,
             "Post-major_stroke" = c_post_major_stroke,
             "Post-MI" = c_post_mi,
             "Death_stroke" = 0,
             "Death_MI" = 0,
             "Death_other" = 0)

## Determine the costs accrued over the cycles (reward at the end of the cycle!)
v_t_c_comp <- m_hs_comp[2:nrow(m_hs_comp),] %*% v_c_comp
v_t_c_int  <- m_hs_int[2:nrow(m_hs_int),] %*% v_c_int

## Determine the costs accrued over the cycles (reward at the end of the cycle!)
v_cum_c_comp <- cumsum(v_t_c_comp)
v_cum_c_int  <- cumsum(v_t_c_int)

## Determine the total costs accrued (sum of all cycles; reward at the end of the cycle!)
n_t_c_comp <- sum(v_t_c_comp)
n_t_c_int  <- sum(v_t_c_int)
```

```r
# Mean outcomes per individual
## incrementals and ICER
kable(cbind(Strategy = c("No aspirin", "Aspirin"),
      LY = round( c(n_t_ly_comp, n_t_ly_int) / n_ind, 2),
      QALY = round( c(n_t_qaly_comp, n_t_qaly_int) / n_ind, 2),
      COSTS = round( c(n_t_c_comp, n_t_c_int) / n_ind, 0),
      INC_QALY = c("-", round(((n_t_qaly_int - n_t_qaly_comp) / n_ind), 2)),
      INC_COSTS = c("-", round(((n_t_c_int - n_t_c_comp) / n_ind), 0)),
      ICER = c("-", round( ((n_t_c_int - n_t_c_comp) / n_ind) / ((n_t_qaly_int - n_t_qaly_comp) / n_ind
      ),
      caption = "Mean undiscounted outcome per individual"
)
```

Table 6: Mean undiscounted outcome per individual

Strategy	LY	QALY	COSTS	INC_QALY	INC_COSTS	ICER
No aspirin	8.43	8.4	1179	-	-	-
Aspirin	8.45	8.42	1605	0.02	425	27932

6. Calculate the discounted results. To do so, define a vector of length `n_cycles`, which contain the discount weights for each cycle, and use matrix multiplication of the vector of total health effects (or costs) by the vector of discount weights. Name the discount weights vector for health effects `v_dw_e` and for costs `v_dw_c`. The yearly discount rates for health effects and costs are provided under the objects `r_d_effects` and `r_d_costs`. The calculations are performed for discounted life years, use that example to perform discounting of QALY's and costs. Calculate the mean discounted outcomes (life years, QALYs, costs) per individual for each strategy. Calculate the incremental QALYs and costs of the "Aspirin" versus "No aspirin" group and calculate the incremental cost-effectiveness ratio (incremental costs/incremental QALYs).

```r
#---------------------------------------#
#### 6. Calculate discounted results ####
#---------------------------------------#

# Life years
## Define discount weights per cycle (years in this case)
v_dw_e <- 1 / (1 + r_d_effects) ^ c(1:n_cycles)

## Total discounted life years, using matrix multiplication
n_t_ly_comp_d <- t(v_t_ly_comp) %*% v_dw_e
n_t_ly_int_d  <- t(v_t_ly_int) %*% v_dw_e

##[ALTERNATIVE]##
### Multiply the number of life year gained in each cycle by the discount weight of each cycle
v_t_ly_comp_d <- v_t_ly_comp * v_dw_e
v_t_ly_int_d  <- v_t_ly_int * v_dw_e

### Sum to obtain total
n_t_ly_comp_d2 <- sum(v_t_ly_comp_d)
n_t_ly_int_d2  <- sum(v_t_ly_int_d)

### Check whether results are the same
round(n_t_ly_comp_d, 5) == round(n_t_ly_comp_d2, 5) # TRUE
```

```
##      [,1]
## [1,] TRUE
```

```
round(n_t_ly_int_d, 5) == round(n_t_ly_int_d2, 5) # TRUE
```

```
##      [,1]
## [1,] TRUE
```

```
# QALYs
## Define discount weights per cycle (years in this case)
#v_dw_e <- 1 / (1 + r_d_effects) ^ c(1:n_cycles)

## Total discounted life years, using matrix multiplication
n_t_qaly_comp_d <- t(v_t_qaly_comp) %*% v_dw_e
n_t_qaly_int_d  <- t(v_t_qaly_int) %*% v_dw_e

# Costs
## Define discount weights per cycle (years in this case)
v_dw_c <- 1 / (1 + r_d_costs) ^ c(1:n_cycles)

## Total discounted life years, using matrix multiplication
n_t_c_comp_d <- t(v_t_c_comp) %*% v_dw_c
n_t_c_int_d  <- t(v_t_c_int) %*% v_dw_c

# Mean discounted outcomes per individual
## incrementals and ICER
kable(cbind(Strategy = c("No aspirin", "Aspirin"),
      LY = round( c(n_t_ly_comp_d, n_t_ly_int_d) / n_ind, 2),
      QALY = round( c(n_t_qaly_comp_d, n_t_qaly_int_d) / n_ind, 2),
      COSTS = round( c(n_t_c_comp_d, n_t_c_int_d) / n_ind, 0),
      INC_QALY = c("-", round(((n_t_qaly_int_d - n_t_qaly_comp_d) / n_ind), 2)),
      INC_COSTS = c("-", round(((n_t_c_int_d - n_t_c_comp_d) / n_ind), 0)),
      ICER = c("-", round( ((n_t_c_int_d - n_t_c_comp_d) / n_ind) / ((n_t_qaly_int_d - n_t_qaly_comp_d)
),
caption = "Mean discounted outcome per individual"
)
```

Table 7: Mean discounted outcome per individual

Strategy	LY	QALY	COSTS	INC_QALY	INC_COSTS	ICER
No aspirin	7.81	7.78	909	-	-	-
Aspirin	7.82	7.8	1278	0.01	369	27178

7. Have a look at the expected life-years, costs, and effects of both strategies.
   7.a. Which strategy is cheapest?
   Answer: The "No aspirin" strategy is cheaper than the "Aspirin" strategy.
   7.b. Which strategy gives most effects in terms of life years? And which one in terms of QALYs?
   Answer: For both outcomes, the "Aspirin" strategy provides the most effects.
   7.c. What is your conclusion when looking at the discounted ICER?
   7.d. What is the difference between the undiscounted and discounted results? Do you understand this difference?

QUESTION 7.e. - 8.b. can be performed using the R shiny app, you can also modify your own model to answer these questions.
To access the shiny app for the following assignment, use the following command in your R session and select the tab "Assignment HSTM 2B".

```
runGitHub("Teaching", "Xa4P", subdir = "Basics/shiny_app_cea/", ref = "main")
```

7.e. What happens to the costs and effects over the time horizon when the discounting rate of both costs and effects is increased to 6%? And what if it is only 0.5%? How does this affect the ICER of aspirin?

```
#-----------------------------------------------------------------#
#### 7. Impact of changing the discount rates on the results ####
#-----------------------------------------------------------------#

# 7.a. Both discount rates to 6%
# Change discount rate
r_d_effects <- 0.06
r_d_costs <- 0.06

# Life years
## Define discount weights per cycle (years in this case)
v_dw_e <- 1 / (1 + r_d_effects) ^ c(1:n_cycles)

## Total discounted life years, using matrix multiplication
n_t_ly_comp_d <- t(v_t_ly_comp) %*% v_dw_e
n_t_ly_int_d  <- t(v_t_ly_int) %*% v_dw_e

##[ALTERNATIVE]##
### Multiply the number of life year gained in each cycle by the discount weight of each cycle
v_t_ly_comp_d <- v_t_ly_comp * v_dw_e
v_t_ly_int_d  <- v_t_ly_int * v_dw_e

### Sum to obtain total
n_t_ly_comp_d2 <- sum(v_t_ly_comp_d)
n_t_ly_int_d2  <- sum(v_t_ly_int_d)

### Check whether results are the same
round(n_t_ly_comp_d, 5) == round(n_t_ly_comp_d2, 5) # TRUE
```

```
##      [,1]
## [1,] TRUE
```

```
round(n_t_ly_int_d, 5) == round(n_t_ly_int_d2, 5) # TRUE
```

```
##      [,1]
## [1,] TRUE
```

```
# QALYs
## Define discount weights per cycle (years in this case)
#v_dw_e <- 1 / (1 + r_d_effects) ^ c(1:n_cycles)

## Total discounted life years, using matrix multiplication
```

```
n_t_qaly_comp_d <- t(v_t_qaly_comp) %*% v_dw_e
n_t_qaly_int_d  <- t(v_t_qaly_int) %*% v_dw_e

# Costs
## Define discount weights per cycle (years in this case)
v_dw_c <- 1 / (1 + r_d_costs) ^ c(1:n_cycles)

## Total discounted life years, using matrix multiplication
n_t_c_comp_d <- t(v_t_c_comp) %*% v_dw_c
n_t_c_int_d  <- t(v_t_c_int) %*% v_dw_c

# Mean discounted outcomes per individual
## incrementals and ICER
kable(cbind(Strategy = c("No aspirin", "Aspirin"),
      LY = round( c(n_t_ly_comp_d, n_t_ly_int_d) / n_ind, 3),
      QALY = round( c(n_t_qaly_comp_d, n_t_qaly_int_d) / n_ind, 3),
      COSTS = round( c(n_t_c_comp_d, n_t_c_int_d) / n_ind, 0),
      INC_QALY = c("-", round(((n_t_qaly_int_d - n_t_qaly_comp_d) / n_ind), 3)),
      INC_COSTS = c("-", round(((n_t_c_int_d - n_t_c_comp_d) / n_ind), 0)),
      ICER = c("-", round( ((n_t_c_int_d - n_t_c_comp_d) / n_ind) / ((n_t_qaly_int_d - n_t_qaly_comp_d)
),
caption = "Mean results per individual when both discount rates are 6%"
)
```

Table 8: Mean results per individual when both discount rates are 6%

Strategy	LY	QALY	COSTS	INC_QALY	INC_COSTS	ICER
No aspirin	6.308	6.291	803	-	-	-
Aspirin	6.319	6.301	1150	0.01	346	35257

```
# 7.b. Both discount rates to 1.5%
# Change discount rate
r_d_effects <- 0.015
r_d_costs <- 0.015

# Life years
## Define discount weights per cycle (years in this case)
v_dw_e <- 1 / (1 + r_d_effects) ^ c(1:n_cycles)

## Total discounted life years, using matrix multiplication
n_t_ly_comp_d <- t(v_t_ly_comp) %*% v_dw_e
n_t_ly_int_d  <- t(v_t_ly_int) %*% v_dw_e

##[ALTERNATIVE]##
### Multiply the number of life year gained in each cycle by the discount weight of each cycle
v_t_ly_comp_d <- v_t_ly_comp * v_dw_e
v_t_ly_int_d  <- v_t_ly_int * v_dw_e

### Sum to obtain total
n_t_ly_comp_d2 <- sum(v_t_ly_comp_d)
n_t_ly_int_d2  <- sum(v_t_ly_int_d)
```

```r
### Check whether results are the same
round(n_t_ly_comp_d, 5) == round(n_t_ly_comp_d2, 5) # TRUE
```

```
##      [,1]
## [1,] TRUE
```

```r
round(n_t_ly_int_d, 5) == round(n_t_ly_int_d2, 5) # TRUE
```

```
##      [,1]
## [1,] TRUE
```

```r
# QALYs
## Define discount weights per cycle (years in this case)
#v_dw_e <- 1 / (1 + r_d_effects) ^ c(1:n_cycles)

## Total discounted life years, using matrix multiplication
n_t_qaly_comp_d <- t(v_t_qaly_comp) %*% v_dw_e
n_t_qaly_int_d  <- t(v_t_qaly_int) %*% v_dw_e

# Costs
## Define discount weights per cycle (years in this case)
v_dw_c <- 1 / (1 + r_d_costs) ^ c(1:n_cycles)

## Total discounted life years, using matrix multiplication
n_t_c_comp_d <- t(v_t_c_comp) %*% v_dw_c
n_t_c_int_d  <- t(v_t_c_int) %*% v_dw_c

# Mean discounted outcomes per individual
## incrementals and ICER
kable(cbind(Strategy = c("No aspirin", "Aspirin"),
      LY = round( c(n_t_ly_comp_d, n_t_ly_int_d) / n_ind, 3),
      QALY = round( c(n_t_qaly_comp_d, n_t_qaly_int_d) / n_ind, 3),
      COSTS = round( c(n_t_c_comp_d, n_t_c_int_d) / n_ind, 0),
      INC_QALY = c("-", round(((n_t_qaly_int_d - n_t_qaly_comp_d) / n_ind), 3)),
      INC_COSTS = c("-", round(((n_t_c_int_d - n_t_c_comp_d) / n_ind), 0)),
      ICER = c("-", round( ((n_t_c_int_d - n_t_c_comp_d) / n_ind) / ((n_t_qaly_int_d - n_t_qaly_comp_d)
),
caption = "Mean results per individual when both discount rates are 1.5%"
)
```

Table 9: Mean results per individual when both discount rates are 1.5%

Strategy	LY	QALY	COSTS	INC_QALY	INC_COSTS	ICER
No aspirin	7.808	7.785	1067	-	-	-
Aspirin	7.822	7.798	1470	0.014	403	29626

Answer: Increasing the discount rates to 6% causes the costs and effects to decrease faster over time. It gives relatively less weight to costs and effects in the future. The ICER increases in this case, because effects were only discounted by 1.5% originally and costs by 4%. Hence, the effects of increasing the discount rates

is greater on the health effects. Decreasing the discount rate of the costs to 1.5% causes the value of the costs to decrease slower over time. It gives relatively more weight to costs in the future. As the costs of aspirin use occur throughout the time horizon and the benefits (MIs and strokes prevented) occur in later years increasing the discount rates will result in an ICER that is higher, that is, less favorable, compared to a lower discount rate.
8. Set the discounting rate back to 4% for costs and 1.5% for effects.
8.a. Change the starting age to 65. What do you expect will happen to the costs and effects? And how does it affect the ICER?
Answer: At 65 the competing risk of dying from other causes is higher than at 45. Therefore the expected gain in life-years and QALYs will be probably be less. Since the costs are still similar the ICER will increase slightly. See below for the table with the mean results per individual.

```
# 8.a. Starting age = 65
# Change discount rate
r_d_effects <- 0.015
r_d_costs <- 0.04
n_start_age <- 65

# Create a vector containing the mortality probability values for the age 'n_start_age' to 'n_start_age
v_p_mort <- df_mort[c(which(df_mort$age == n_start_age):
                      which(df_mort$age == n_start_age + 9)
), "p_mort"]

# Initialise array's
a_tp_comp <- array(0, dim = c(n_hs, n_hs, n_cycles),
                   dimnames = list(v_names_hs, v_names_hs, 1:n_cycles)
)

# Fill in the arrays using the background mortality for each age
a_tp_comp["Well", "Well", ]   <- 1 - v_p_mort - r_inc_mi - r_inc_stroke  # EXAMPLE! Calculate the remai
##Notice, that only using the first 2 elements of the array will fill these transition probabilities fo

a_tp_comp["Well", "Post-minor_stroke", ] <- r_inc_stroke * (1 - r_fatal_stroke) * p_post_minor_stroke
a_tp_comp["Well", "Post-major_stroke", ] <- r_inc_stroke * (1 - r_fatal_stroke) * p_post_major_stroke
a_tp_comp["Well", "Post-MI", ] <- r_inc_mi * (1 - r_fatal_mi)
a_tp_comp["Well", "Death_stroke", ] <- r_inc_stroke * r_fatal_stroke
a_tp_comp["Well", "Death_MI", ] <- r_inc_mi * r_fatal_mi
a_tp_comp["Well", "Death_other", ] <- v_p_mort

a_tp_comp["Post-minor_stroke", "Post-minor_stroke", ] <- 1 - v_p_mort
a_tp_comp["Post-minor_stroke", "Death_other", ] <- v_p_mort

a_tp_comp["Post-major_stroke", "Post-major_stroke", ] <- 1 - v_p_mort
a_tp_comp["Post-major_stroke", "Death_other", ] <- v_p_mort

a_tp_comp["Post-MI", "Post-MI", ] <- 1 - v_p_mort
a_tp_comp["Post-MI", "Death_other", ] <- v_p_mort

a_tp_comp["Death_stroke", "Death_stroke",] <- 1
a_tp_comp["Death_MI", "Death_MI",] <- 1
a_tp_comp["Death_other", "Death_other",] <- 1

# Inspect whether the sum of all TP's for all health states over all cycles = 1
m_check_comp <- matrix(NA,
```

```
                      nrow = n_cycles,
                      ncol = n_hs)

for (i in 1:n_cycles){
  v_res <- round(rowSums(a_tp_comp[ , ,i]), 5) == 1
  m_check_comp[i,] <- paste(v_res)

}
#m_check_comp # visualise the check

# Create a matrix to store the cohort simulation (‘m_hs_comp‘)
m_hs_comp <- matrix(0,
                    ncol = length(v_names_hs),
                    nrow = n_cycles + 1,
                    dimnames = list(c(0:n_cycles),
                                    v_names_hs)
)



# Define the start position of individuals (all in "well")
m_hs_comp[1,] <- v_start_hs

# Perform the matrix multiplication using the 3D array.
for(cycle in 1:n_cycles){
  m_hs_comp[cycle + 1,] <- m_hs_comp[cycle,] %*% a_tp_comp[, , cycle]

}

# Make a copy of ‘a_tp_comp‘ and call it ‘a_tp_int‘
a_tp_int <- a_tp_comp

# Modify ‘a_tp_int‘ since Aspirin has an
a_tp_int["Well", "Well", ]   <- 1 - v_p_mort - r_inc_mi * eff_mi - r_inc_stroke * eff_stroke
a_tp_int["Well", "Post-minor_stroke", ] <- r_inc_stroke * eff_stroke * (1 - r_fatal_stroke) * p_post_mi
a_tp_int["Well", "Post-major_stroke", ] <- r_inc_stroke * eff_stroke * (1 - r_fatal_stroke) * p_post_ma
a_tp_int["Well", "Post-MI", ] <- r_inc_mi * eff_mi * (1 - r_fatal_mi)
a_tp_int["Well", "Death_stroke", ] <- r_inc_stroke * eff_stroke * r_fatal_stroke
a_tp_int["Well", "Death_MI", ] <- r_inc_mi * eff_mi * r_fatal_mi

# Inspect whether the sum of all TP's for all health states over all cycles = 1
m_check_int <- matrix(NA,
                      nrow = n_cycles,
                      ncol = n_hs)

for (i in 1:n_cycles){
  v_res <- round(rowSums(a_tp_int[ , ,i]), 5) == 1
  m_check_int[i,] <- paste(v_res)

}
#m_check_int # visualise the check

# Create a matrix to store the cohort simulation (‘m_hs_int‘)
```

```r
m_hs_int <- matrix(0,
                   ncol = length(v_names_hs),
                   nrow = n_cycles + 1,
                   dimnames = list(c(0:n_cycles),
                                   v_names_hs)
)



# Define the start position of individuals (all in "well")
m_hs_int[1,] <- v_start_hs

# Perform the matrix multiplication using the 3D array.
for(cycle in 1:n_cycles){
  m_hs_int[cycle + 1,] <- m_hs_int[cycle,] %*% a_tp_int[, , cycle]

}

# Life years: EXAMPLE! DO THE SAME FOR QALY's AND COSTS
## Determine the number of life year won by 1 individual during 1 cycle
v_ly_comp <- c("Well" = 1,
               "Post-minor_stroke" = 1,
               "Post-major_stroke" = 1,
               "Post-MI" = 1,
               "Death_stroke" = 0,
               "Death_MI" = 0,
               "Death_other" = 0)
v_ly_int <- c("Well" = 1,
              "Post-minor_stroke" = 1,
              "Post-major_stroke" = 1,
              "Post-MI" = 1,
              "Death_stroke" = 0,
              "Death_MI" = 0,
              "Death_other" = 0)

## Determine the number of life year gained over the cycles (reward at the end of the cycle!)
v_t_ly_comp <- m_hs_comp[2:nrow(m_hs_comp),] %*% v_ly_comp
v_t_ly_int  <- m_hs_int[2:nrow(m_hs_int),] %*% v_ly_int

## Determine the cumulative number of life year gained over the cycles (reward at the end of the cycle!
v_cum_ly_comp <- cumsum(v_t_ly_comp)
v_cum_ly_int  <- cumsum(v_t_ly_int)

## Determine the total number of life year gained (sum of all cycles; reward at the end of the cycle!)
n_t_ly_comp <- sum(v_t_ly_comp)
n_t_ly_int <- sum(v_t_ly_int)

# QALY's
## Determine the number of QALYs won by 1 individual during 1 cycle
v_qaly_comp <- c("Well" = u_healthy,
                 "Post-minor_stroke" = u_post_minor_stroke,
                 "Post-major_stroke" = u_post_major_stroke,
                 "Post-MI" = u_post_mi,
```

```
                 "Death_stroke" = 0,
                 "Death_MI" = 0,
                 "Death_other" = 0)

v_qaly_int <- c("Well" = u_aspirin_use,
                "Post-minor_stroke" = u_post_minor_stroke,
                "Post-major_stroke" = u_post_major_stroke,
                "Post-MI" = u_post_mi,
                "Death_stroke" = 0,
                "Death_MI" = 0,
                "Death_other" = 0)

## Determine the number of QALYs gained over the cycles (reward at the end of the cycle!)
v_t_qaly_comp <- m_hs_comp[2:nrow(m_hs_comp),] %*% v_qaly_comp
v_t_qaly_int  <- m_hs_int[2:nrow(m_hs_int),] %*% v_qaly_int

## Determine the cumulative number of QALYsr gained over the cycles (reward at the end of the cycle!)
v_cum_qaly_comp <- cumsum(v_t_qaly_comp)
v_cum_qaly_int <- cumsum(v_t_qaly_int)

## Determine the total number of QALYs gained (sum of all cycles; reward at the end of the cycle!)
n_t_qaly_comp <- sum(v_t_qaly_comp)
n_t_qaly_int  <- sum(v_t_qaly_int)

# Costs
## Determine the costs accrued by 1 individual during 1 cycle
v_c_comp <- c("Well" = 0,
              "Post-minor_stroke" = c_post_minor_stroke,
              "Post-major_stroke" = c_post_major_stroke,
              "Post-MI" = c_post_mi,
              "Death_stroke" = 0,
              "Death_MI" = 0,
              "Death_other" = 0)

v_c_int <- c("Well" = c_aspirin_use,
             "Post-minor_stroke" = c_post_minor_stroke,
             "Post-major_stroke" = c_post_major_stroke,
             "Post-MI" = c_post_mi,
             "Death_stroke" = 0,
             "Death_MI" = 0,
             "Death_other" = 0)

## Determine the costs accrued over the cycles (reward at the end of the cycle!)
v_t_c_comp <- m_hs_comp[2:nrow(m_hs_comp),] %*% v_c_comp
v_t_c_int  <- m_hs_int[2:nrow(m_hs_int),] %*% v_c_int

## Determine the costs accrued over the cycles (reward at the end of the cycle!)
v_cum_c_comp <- cumsum(v_t_c_comp)
v_cum_c_int  <- cumsum(v_t_c_int)

## Determine the total costs accrued (sum of all cycles; reward at the end of the cycle!)
n_t_c_comp <- sum(v_t_c_comp)
n_t_c_int  <- sum(v_t_c_int)
```

```r
# Life years
## Define discount weights per cycle (years in this case)
v_dw_e <- 1 / (1 + r_d_effects) ^ c(1:n_cycles)

## Total discounted life years, using matrix multiplication
n_t_ly_comp_d <- t(v_t_ly_comp) %*% v_dw_e
n_t_ly_int_d  <- t(v_t_ly_int) %*% v_dw_e

# QALYs
## Define discount weights per cycle (years in this case)
#v_dw_e <- 1 / (1 + r_d_effects) ^ c(1:n_cycles)

## Total discounted life years, using matrix multiplication
n_t_qaly_comp_d <- t(v_t_qaly_comp) %*% v_dw_e
n_t_qaly_int_d  <- t(v_t_qaly_int) %*% v_dw_e

# Costs
## Define discount weights per cycle (years in this case)
v_dw_c <- 1 / (1 + r_d_costs) ^ c(1:n_cycles)

## Total discounted life years, using matrix multiplication
n_t_c_comp_d <- t(v_t_c_comp) %*% v_dw_c
n_t_c_int_d  <- t(v_t_c_int) %*% v_dw_c

# Mean discounted outcomes per individual
## incrementals and ICER
kable(cbind(Strategy = c("No aspirin", "Aspirin"),
      LY = round( c(n_t_ly_comp_d, n_t_ly_int_d) / n_ind, 3),
      QALY = round( c(n_t_qaly_comp_d, n_t_qaly_int_d) / n_ind, 3),
      COSTS = round( c(n_t_c_comp_d, n_t_c_int_d) / n_ind, 0),
      INC_QALY = c("-", round(((n_t_qaly_int_d - n_t_qaly_comp_d) / n_ind), 3)),
      INC_COSTS = c("-", round(((n_t_c_int_d - n_t_c_comp_d) / n_ind), 0)),
      ICER = c("-", round( ((n_t_c_int_d - n_t_c_comp_d) / n_ind) / ((n_t_qaly_int_d - n_t_qaly_comp_d)
),
caption = "Mean results per individual, with starting age = 65"
)
```

Table 10: Mean results per individual, with starting age = 65

Strategy	LY	QALY	COSTS	INC_QALY	INC_COSTS	ICER
No aspirin	5.996	5.979	674	-	-	-
Aspirin	6.007	5.989	972	0.01	298	30636

8.b. Change the starting age to 95. What do you think will now happen to the effects? How do you explain these results? How does this affect the ICER?
Answer: At 95 the competing risk of dying from other causes is so high that (almost) no life-years or QALYs are gained with the strategy of prescribing aspirin. The (discounted) ICER has now dramatically increased above the € 50,000 per QALY gained. See below for the mean results per individual.

```r
# 8.a. Starting age = 95
# Change discount rate
```

```r
n_start_age <- 95

# Create a vector containing the mortality probability values for the age `n_start_age` to `n_start_age
v_p_mort <- df_mort[c(which(df_mort$age == n_start_age):
                        which(df_mort$age == n_start_age + 9)
), "p_mort"]

# Initialise array's
a_tp_comp <- array(0, dim = c(n_hs, n_hs, n_cycles),
                   dimnames = list(v_names_hs, v_names_hs, 1:n_cycles)
)

# Fill in the arrays using the background mortality for each age
a_tp_comp["Well", "Well", ]   <- 1 - v_p_mort - r_inc_mi - r_inc_stroke  # EXAMPLE! Calculate the remai
##Notice, that only using the first 2 elements of the array will fill these transition probabilities fo

a_tp_comp["Well", "Post-minor_stroke", ] <- r_inc_stroke * (1 - r_fatal_stroke) * p_post_minor_stroke
a_tp_comp["Well", "Post-major_stroke", ] <- r_inc_stroke * (1 - r_fatal_stroke) * p_post_major_stroke
a_tp_comp["Well", "Post-MI", ] <- r_inc_mi * (1 - r_fatal_mi)
a_tp_comp["Well", "Death_stroke", ] <- r_inc_stroke * r_fatal_stroke
a_tp_comp["Well", "Death_MI", ] <- r_inc_mi * r_fatal_mi
a_tp_comp["Well", "Death_other", ] <- v_p_mort

a_tp_comp["Post-minor_stroke", "Post-minor_stroke", ] <- 1 - v_p_mort
a_tp_comp["Post-minor_stroke", "Death_other", ] <- v_p_mort

a_tp_comp["Post-major_stroke", "Post-major_stroke", ] <- 1 - v_p_mort
a_tp_comp["Post-major_stroke", "Death_other", ] <- v_p_mort

a_tp_comp["Post-MI", "Post-MI", ] <- 1 - v_p_mort
a_tp_comp["Post-MI", "Death_other", ] <- v_p_mort

a_tp_comp["Death_stroke", "Death_stroke",] <- 1
a_tp_comp["Death_MI", "Death_MI",] <- 1
a_tp_comp["Death_other", "Death_other",] <- 1

# Inspect whether the sum of all TP's for all health states over all cycles = 1
m_check_comp <- matrix(NA,
                       nrow = n_cycles,
                       ncol = n_hs)

for (i in 1:n_cycles){
  v_res <- round(rowSums(a_tp_comp[ , ,i]), 5) == 1
  m_check_comp[i,] <- paste(v_res)

}
#m_check_comp # visualise the check

# Create a matrix to store the cohort simulation (`m_hs_comp`)
m_hs_comp <- matrix(0,
                    ncol = length(v_names_hs),
                    nrow = n_cycles + 1,
                    dimnames = list(c(0:n_cycles),
```

```
                                    v_names_hs)
)



# Define the start position of individuals (all in "well")
m_hs_comp[1,] <- v_start_hs

# Perform the matrix multiplication using the 3D array.
for(cycle in 1:n_cycles){
  m_hs_comp[cycle + 1,] <- m_hs_comp[cycle,] %*% a_tp_comp[, , cycle]

}

# Make a copy of 'a_tp_comp' and call it 'a_tp_int'
a_tp_int <- a_tp_comp

# Modify 'a_tp_int' since Aspirin has an
a_tp_int["Well", "Well", ]   <- 1 - v_p_mort - r_inc_mi * eff_mi - r_inc_stroke * eff_stroke
a_tp_int["Well", "Post-minor_stroke", ] <- r_inc_stroke * eff_stroke * (1 - r_fatal_stroke) * p_post_mi
a_tp_int["Well", "Post-major_stroke", ] <- r_inc_stroke * eff_stroke * (1 - r_fatal_stroke) * p_post_ma
a_tp_int["Well", "Post-MI", ] <- r_inc_mi * eff_mi * (1 - r_fatal_mi)
a_tp_int["Well", "Death_stroke", ] <- r_inc_stroke * eff_stroke * r_fatal_stroke
a_tp_int["Well", "Death_MI", ] <- r_inc_mi * eff_mi * r_fatal_mi

# Inspect whether the sum of all TP's for all health states over all cycles = 1
m_check_int <- matrix(NA,
                      nrow = n_cycles,
                      ncol = n_hs)

for (i in 1:n_cycles){
  v_res <- round(rowSums(a_tp_int[ , ,i]), 5) == 1
  m_check_int[i,] <- paste(v_res)

}
#m_check_int # visualise the check

# Create a matrix to store the cohort simulation ('m_hs_int')
m_hs_int <- matrix(0,
                   ncol = length(v_names_hs),
                   nrow = n_cycles + 1,
                   dimnames = list(c(0:n_cycles),
                                   v_names_hs)
)



# Define the start position of individuals (all in "well")
m_hs_int[1,] <- v_start_hs

# Perform the matrix multiplication using the 3D array.
for(cycle in 1:n_cycles){
  m_hs_int[cycle + 1,] <- m_hs_int[cycle,] %*% a_tp_int[, , cycle]
```

```
}

# Life years: EXAMPLE! DO THE SAME FOR QALY's AND COSTS
## Determine the number of life year won by 1 individual during 1 cycle
v_ly_comp <- c("Well" = 1,
               "Post-minor_stroke" = 1,
               "Post-major_stroke" = 1,
               "Post-MI" = 1,
               "Death_stroke" = 0,
               "Death_MI" = 0,
               "Death_other" = 0)
v_ly_int <- c("Well" = 1,
              "Post-minor_stroke" = 1,
              "Post-major_stroke" = 1,
              "Post-MI" = 1,
              "Death_stroke" = 0,
              "Death_MI" = 0,
              "Death_other" = 0)

## Determine the number of life year gained over the cycles (reward at the end of the cycle!)
v_t_ly_comp <- m_hs_comp[2:nrow(m_hs_comp),] %*% v_ly_comp
v_t_ly_int  <- m_hs_int[2:nrow(m_hs_int),] %*% v_ly_int

## Determine the cumulative number of life year gained over the cycles (reward at the end of the cycle!)
v_cum_ly_comp <- cumsum(v_t_ly_comp)
v_cum_ly_int  <- cumsum(v_t_ly_int)

## Determine the total number of life year gained (sum of all cycles; reward at the end of the cycle!)
n_t_ly_comp <- sum(v_t_ly_comp)
n_t_ly_int <- sum(v_t_ly_int)

# QALY's
## Determine the number of QALYs won by 1 individual during 1 cycle
v_qaly_comp <- c("Well" = u_healthy,
                 "Post-minor_stroke" = u_post_minor_stroke,
                 "Post-major_stroke" = u_post_major_stroke,
                 "Post-MI" = u_post_mi,
                 "Death_stroke" = 0,
                 "Death_MI" = 0,
                 "Death_other" = 0)

v_qaly_int <- c("Well" = u_aspirin_use,
                "Post-minor_stroke" = u_post_minor_stroke,
                "Post-major_stroke" = u_post_major_stroke,
                "Post-MI" = u_post_mi,
                "Death_stroke" = 0,
                "Death_MI" = 0,
                "Death_other" = 0)

## Determine the number of QALYs gained over the cycles (reward at the end of the cycle!)
v_t_qaly_comp <- m_hs_comp[2:nrow(m_hs_comp),] %*% v_qaly_comp
v_t_qaly_int  <- m_hs_int[2:nrow(m_hs_int),] %*% v_qaly_int
```

```r
## Determine the cumulative number of QALYsr gained over the cycles (reward at the end of the cycle!)
v_cum_qaly_comp <- cumsum(v_t_qaly_comp)
v_cum_qaly_int <- cumsum(v_t_qaly_int)

## Determine the total number of QALYs gained (sum of all cycles; reward at the end of the cycle!)
n_t_qaly_comp <- sum(v_t_qaly_comp)
n_t_qaly_int  <- sum(v_t_qaly_int)

# Costs
## Determine the costs accrued by 1 individual during 1 cycle
v_c_comp <- c("Well" = 0,
              "Post-minor_stroke" = c_post_minor_stroke,
              "Post-major_stroke" = c_post_major_stroke,
              "Post-MI" = c_post_mi,
              "Death_stroke" = 0,
              "Death_MI" = 0,
              "Death_other" = 0)

v_c_int <- c("Well" = c_aspirin_use,
             "Post-minor_stroke" = c_post_minor_stroke,
             "Post-major_stroke" = c_post_major_stroke,
             "Post-MI" = c_post_mi,
             "Death_stroke" = 0,
             "Death_MI" = 0,
             "Death_other" = 0)

## Determine the costs accrued over the cycles (reward at the end of the cycle!)
v_t_c_comp <- m_hs_comp[2:nrow(m_hs_comp),] %*% v_c_comp
v_t_c_int  <- m_hs_int[2:nrow(m_hs_int),] %*% v_c_int

## Determine the costs accrued over the cycles (reward at the end of the cycle!)
v_cum_c_comp <- cumsum(v_t_c_comp)
v_cum_c_int  <- cumsum(v_t_c_int)

## Determine the total costs accrued (sum of all cycles; reward at the end of the cycle!)
n_t_c_comp <- sum(v_t_c_comp)
n_t_c_int  <- sum(v_t_c_int)

# Life years
## Define discount weights per cycle (years in this case)
v_dw_e <- 1 / (1 + r_d_effects) ^ c(1:n_cycles)

## Total discounted life years, using matrix multiplication
n_t_ly_comp_d <- t(v_t_ly_comp) %*% v_dw_e
n_t_ly_int_d  <- t(v_t_ly_int) %*% v_dw_e

# QALYs
## Define discount weights per cycle (years in this case)
#v_dw_e <- 1 / (1 + r_d_effects) ^ c(1:n_cycles)

## Total discounted life years, using matrix multiplication
n_t_qaly_comp_d <- t(v_t_qaly_comp) %*% v_dw_e
n_t_qaly_int_d  <- t(v_t_qaly_int) %*% v_dw_e
```

```r
# Costs
## Define discount weights per cycle (years in this case)
v_dw_c <- 1 / (1 + r_d_costs) ^ c(1:n_cycles)

## Total discounted life years, using matrix multiplication
n_t_c_comp_d <- t(v_t_c_comp) %*% v_dw_c
n_t_c_int_d  <- t(v_t_c_int) %*% v_dw_c

# Mean discounted outcomes per individual
## incrementals and ICER
kable(cbind(Strategy = c("No aspirin", "Aspirin"),
      LY = round( c(n_t_ly_comp_d, n_t_ly_int_d) / n_ind, 3),
      QALY = round( c(n_t_qaly_comp_d, n_t_qaly_int_d) / n_ind, 3),
      COSTS = round( c(n_t_c_comp_d, n_t_c_int_d) / n_ind, 0),
      INC_QALY = c("-", round(((n_t_qaly_int_d - n_t_qaly_comp_d) / n_ind), 3)),
      INC_COSTS = c("-", round(((n_t_c_int_d - n_t_c_comp_d) / n_ind), 0)),
      ICER = c("-", round( ((n_t_c_int_d - n_t_c_comp_d) / n_ind) / ((n_t_qaly_int_d - n_t_qaly_comp_d)
),
caption = "Mean results per individual, with starting age = 95"
)
```

Table 11: Mean results per individual, with starting age = 95

Strategy	LY	QALY	COSTS	INC_QALY	INC_COSTS	ICER
No aspirin	2.064	2.059	193	-	-	-
Aspirin	2.067	2.062	318	0.002	125	56115

Reference

Alarid-Escudero, Fernando, Eline M. Krijkamp, Eva A. Enns, Alan Yang, M. G. Myriam Hunink, Petros Pechlivanoglou, and Hawre Jalal. 2020. "Cohort State-Transition Models in R: A Tutorial." http://arxiv.org/abs/2001.07824.