

# Simple 3-state microsimulation model

Includes age and sex specific probability of dying when Healthy

The DARTH workgroup

Developed by the Decision Analysis in R for Technologies in Health (DARTH) workgroup:

Fernando Alarid-Escudero, PhD (1)

Eva A. Enns, MS, PhD (2)

M.G. Myriam Hunink, MD, PhD (3,4)

Hawre J. Jalal, MD, PhD (5)

Eline M. Krijkamp, MSc (3)

Petros Pechlivanoglou, PhD (6,7)

Alan Yang, MSc (7)

In collaboration of:

1. Drug Policy Program, Center for Research and Teaching in Economics (CIDE) - CONACyT, Aguascalientes, Mexico
2. University of Minnesota School of Public Health, Minneapolis, MN, USA
3. Erasmus MC, Rotterdam, The Netherlands
4. Harvard T.H. Chan School of Public Health, Boston, USA
5. University of Pittsburgh Graduate School of Public Health, Pittsburgh, PA, USA
6. University of Toronto, Toronto ON, Canada
7. The Hospital for Sick Children, Toronto ON, Canada

Please cite our publications when using this code:

- Jalal H, Pechlivanoglou P, Krijkamp E, Alarid-Escudero F, Enns E, Hunink MG. An Overview of R in Health Decision Sciences. *Med Decis Making*. 2017; 37(3): 735-746. <https://journals.sagepub.com/doi/abs/10.1177/0272989X16686559>
- Krijkamp EM, Alarid-Escudero F, Enns EA, Jalal HJ, Hunink MGM, Pechlivanoglou P. Microsimulation modeling for health decision sciences using R: A tutorial. *Med Decis Making*. 2018;38(3):400-22. <https://journals.sagepub.com/doi/abs/10.1177/0272989X18754513>
- Krijkamp EM, Alarid-Escudero F, Enns E, Pechlivanoglou P, Hunink MM, Jalal H. A Multidimensional Array Representation of State-Transition Model Dynamics. *Med Decis Mak*. 2020;40(2):242-248. <https://doi.org/10.1177/0272989X19893973>

Copyright 2017, THE HOSPITAL FOR Sick CHILDREN AND THE COLLABORATING INSTITUTIONS. All rights reserved in Canada, the United States and worldwide. Copyright, trademarks, trade names and any and all associated intellectual property are exclusively owned by THE HOSPITAL FOR Sick CHILDREN and the collaborating institutions. These materials may be used, reproduced, modified, distributed and adapted with proper attribution.

Change eval to TRUE if you want to knit this document.

## 01 Load packages

```
if (!require('pacman')) install.packages('pacman'); library(pacman) # use this package to conveniently
# load (install if required) packages from CRAN
p_load("here", "dplyr", "devtools", "scales", "ellipse", "ggplot2", "lazyeval", "igraph", "ggraph", "re
# load (install if required) packages from GitHub
# install_github("DARTH-git/darthtools", force = TRUE) # Uncomment if there is a newer version
p_load_gh("DARTH-git/darthtools")
```

## 02 Load functions

```
# No functions needed
```

## 03 Input model parameters

```
set.seed(1) # set the seed

# Model structure
v_names_states <- c("Healthy", "Sick", "Dead") # vector with state names
n_states      <- length(v_names_states)      # number of states
n_t           <- 60                          # number of cycles
n_i           <- 10000                       # number of individuals
d_e           <- d_c <- 0.03                 # equal discount of costs and QALYs

#### Deterministic analysis ####

# Transition probabilities
p_HS          <- 0.05      # probability Healthy -> Sick
p_HD_female   <- 0.0382    # probability health -> Dead when female
p_HD_male     <- 0.0463    # probability health -> Dead when male
p_SD          <- 0.1       # probability Sick -> Dead

# Costs inputs
c_H           <- 1500      # cost of one cycle in Healthy state
c_S           <- 5000      # cost of one cycle in Sick state
c_D           <- 0         # cost of one cycle in Dead state

# utility inputs
u_H           <- 1         # utility when Healthy
u_S           <- 0.85      # utility when Sick
u_D           <- 0         # utility when Dead

# calculate discount weights for costs for each cycle based on discount rate d_c
```

```

v_dwc      <- 1 / (1 + d_e) ^ (0:n_t)
# calculate discount weights for effectiveness for each cycle based on discount rate d_e
v_dwe      <- 1 / (1 + d_c) ^ (0:n_t)

m_p_HD      <- data.frame(Sex = c("Female", "Male"), p_HD = c(p_HD_female, p_HD_male))

```

## 04 Sample individual level characteristics

### 04.1 Static characteristics

```

# randomly sample the sex of an individual (50% female)
v_sex <- sample(x = c("Female", "Male"), prob = c(0.5, 0.5), size = n_i, replace = TRUE)
df_X  <- data.frame(ID = 1:n_i, Sex = v_sex)

```

### 04.2 Dynamic characteristics

```

# Specify the initial health state of the individuals
# everyone begins in the Healthy state (in this example)
v_M_init <- rep("Healthy", times = n_i)

```

## 05 Define Simulation Functions

### 05.1 Probability function

The function that updates the transition probabilities of every cycle is shown below.

```

Probs <- function(M_t, df_X) {
  # Arguments:
  # M_t: health state occupied at cycle t (character variable)
  # df_X: data frame with individual characteristics data
  # Returns:
  # transition probabilities for that cycle

  # create matrix of state transition probabilities
  m_p_t <- matrix(0, nrow = n_states, ncol = n_i)
  rownames(m_p_t) <- v_names_states # give the state names to the rows

  # lookup baseline probability and rate of dying based on individual characteristics
  p_HD_all <- inner_join(df_X, m_p_HD, by = c("Sex"))
  p_HD <- p_HD_all[M_t == "Healthy", "p_HD"]

  # update m_p_t with the appropriate probabilities
  # transition probabilities when Healthy
  m_p_t[, M_t == "Healthy"] <- rbind((1 - p_HD) * (1 - p_HS),
                                     (1 - p_HD) * p_HS,
                                     p_HD)
}

```

```

# transition probabilities when Sick
m_p_t[, M_t == "Sick"] <- rbind(
  0,
  1 - p_SD,
  p_SD)

# transition probabilities when Dead
m_p_t[, M_t == "Dead"] <- rbind(0,
  0,
  1)

return(t(m_p_t))
}

```

## 05.2 Cost function

The Costs function estimates the costs at every cycle.

```

Costs <- function (M_t) {
  # Arguments:
  # M_t: health state occupied at cycle t (character variable)
  # Returns:
  # costs accrued in this cycle

  c_t <- c()
  c_t[M_t == "Dead"] <- c_D # costs at Dead state
  c_t[M_t == "Healthy"] <- c_H # costs accrued by being Healthy this cycle
  c_t[M_t == "Sick"] <- c_S # costs accrued by being Sick this cycle

  return(c_t) # return costs accrued this cycle
}

```

## 05.3 Health outcome function

The Effs function to update the utilities at every cycle.

```

Effs <- function (M_t) {
  # Arguments:
  # M_t: health state occupied at cycle t (character variable)
  # Returns:
  # QALYs accrued this cycle

  q_t <- c()
  q_t[M_t == "Dead"] <- u_D # QALYs at Dead state
  q_t[M_t == "Healthy"] <- u_H # QALYs accrued by being Healthy this cycle
  q_t[M_t == "Sick"] <- u_S # QALYs accrued by being Sick this cycle

  return(q_t) # return the QALYs accrued this cycle
}

```

## 06 Run Microsimulation

```
MicroSim <- function(n_i, df_X, seed = 1) {  
  # Arguments:  
  # n_i:      number of individuals  
  # df_X:     data frame with individual data  
  # seed:     defaults is 1  
  # Returns  
  # a list with information about the individuals transitions, associated costs and  
  # effects and total costs and rewards  
  
  set.seed(seed) # set the seed  
  
  # create three matrices called m_M, m_C and m_E  
  # number of rows is equal to the n_i, the number of columns is equal to n_t  
  # (the initial state and all the n_t cycles)  
  # m_M is used to store the health state information over time for every individual  
  # m_C is used to store the costs information over time for every individual  
  # m_E is used to store the effects information over time for every individual  
  
  m_M <- m_C <- m_E <- matrix(nrow = n_i, ncol = n_t + 1,  
                                dimnames = list(paste("ind" , 1:n_i, sep = " "),  
                                                  paste("cycle", 0:n_t, sep = " ")))  
  
  m_M[, 1] <- v_M_init      # initial health state  
  m_C[, 1] <- Costs(m_M[, 1]) # costs accrued during cycle 0  
  m_E[, 1] <- Effe(m_M[, 1]) # QALYs accrued during cycle 0  
  
  # open a loop for time running cycles 1 to n_t  
  for (t in 1:n_t) {  
    # calculate the transition probabilities for the cycle based on health state t  
    m_P <- Probs(m_M[, t], df_X)  
    # check if transition probabilities are between 0 and 1  
    check_transition_probability(m_P, verbose = TRUE)  
    # check if each of the rows of the transition probabilities matrix sum to one  
    check_sum_of_transition_array(m_P, n_rows = n_i, n_cycles = n_t, verbose = TRUE)  
    # sample the current health state and store that state in matrix m_M  
    m_M[, t + 1] <- samplev(m_P, 1)  
    m_C[, t + 1] <- Costs(m_M[, t + 1]) # calculate costs  
    m_E[, t + 1] <- Effe(m_M[, t + 1]) # calculate QALYs  
  
    # Display simulation progress  
    if(t/(n_t/10) == round(t/(n_t/10), 0)) { # display progress every 10%  
      cat('\r', paste(t/n_t * 100, "% done", sep = " "))  
    }  
  } # close the loop for the time points  
  
  # calculate  
  tc <- m_C %*% v_dwc # total (discounted) cost per individual  
  te <- m_E %*% v_dwe # total (discounted) QALYs per individual  
  tc_hat <- mean(tc)  # average (discounted) cost  
  te_hat <- mean(te)  # average (discounted) QALYs
```

```

# store the results from the simulation in a list
results <- list(m_M = m_M, m_C = m_C, m_E = m_E, tc = tc , te = te, tc_hat = tc_hat,
               te_hat = te_hat)

return(results) # return the results

} # end of the MicroSim function

# By specifying all the arguments in the `MicroSim()` the simulation can be started

# Run the simulation model
outcomes <- MicroSim(n_i = n_i, df_X = df_X, seed = 1)

# Show results
results <- data.frame("Total Cost" = outcomes$tc_hat, "Total QALYs" = outcomes$te_hat)
results

```

## 07 Visualize results

```

plot(density(outcomes$tc), main = paste("Total cost per person"), xlab = "Cost ($)")
plot(density(outcomes$te), main = paste("Total QALYs per person"), xlab = "QALYs")
plot_trace_microsim(outcomes$m_M) # health state trace

```

Small note: The difference between `paste()` and `paste0()` is that the argument sep by default is " " (`paste`) and "" (`paste0`). In conclusion, `paste0()` is faster than `paste()` if our objective is concatenate strings without spaces because we don't have to specify the argument sep.