# Microsimulation Sick-Sicker model with time dependency

Includes individual characteristics: age, age dependent mortality, individual treatment effect modifier, state-residency for the sick (S1) state, increasing change of death in the first 6 year of sickness

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)

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 EA, et al. A Multidimensional Array Representation of State-Transition Model Dynamics. Med Decis Mak. https://doi.org/10.1177/0272989X19893973

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

```
rm(list = ls())      # clear memory (removes all the variables from the workspace)
```

## 01 Load packages

```
if (!require('pacman')) install.packages('pacman'); library(pacman)
# load (install if required) packages from CRAN
p_load("here", "devtools", "dplyr", "scales", "ellipse", "ggplot2", "lazyeval", "igraph", "truncnorm",
# 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

# Strategy names
v_names_str <- c("no treatment", "treatment")  # strategy names
n_str        <- length(v_names_str)            # number of strategies

# Model structure
v_names_states <- c("H", "S1", "S2", "D")  # the model states names
n_states <- length(v_names_states)         # the number of health states

# Model parameters
n_t      <- 30                      # time horizon, 30 cycles
n_i      <- 100000                  # number of simulated individuals
d_r      <- 0.03                    # discount rate of 3% per cycle
v_dwe    <- v_dwc <- 1 / ((1 + d_r) ^ (0:n_t)) # discount weight

# Event probabilities (per cycle)
# Annual transition probabilities
# (all non-probabilities are conditional on survival)
p_HS1   <- 0.15   # probability of becoming sick when healthy
p_S1H   <- 0.5    # probability of recovering to healthy when sick
p_S1S2  <- 0.105  # probability of becoming sicker when sick

# Annual probabilities of death
# load age dependent probability
# NOTE: if you don't have a data folder but the files in the same folder as your markdown you need
```

```r
# read.csv("mortProb_age.csv") and
# read.csv("MyPopulation-AgeDistribution.csv")
p_mort   <- read.csv("mortProb_age.csv")
# load age distribution
dist_Age <- read.csv("MyPopulation-AgeDistribution.csv")

# probability to die in S1 by cycle (is increasing)
p_S1D    <- c(0.0149, 0.018, 0.021, 0.026, 0.031, rep(0.037, n_t - 5))
p_S2D    <- 0.048  # probability to die in S2

# Cost inputs
c_H      <- 2000    # cost of one cycle in the healthy state
c_S1     <- 4000    # cost of one cycle in the sick state
c_S2     <- 15000   # cost of one cycle in the sicker state
c_D      <- 0       # cost of one cycle in the dead state
c_trt    <- 12000   # cost of treatment (per cycle)

# Utility inputs
u_H      <- 1       # utility when healthy
u_S1     <- 0.75    # utility when sick
u_S2     <- 0.5     # utility when sicker
u_D      <- 0       # utility when dead
u_trt    <- 0.95    # utility when sick and being treated
```

# 04 Sample individual level characteristics

## 04.1 Static characteristics

```r
v_x      <- runif(n_i, min = 0.95, max = 1.05) # treatment effect modifier at baseline

# sample from age distribution an initial age for every individual
# your turn
```

## 04.2 Dynamic characteristics

```r
# Specify the initial health state of the individuals
# everyone begins in the healthy state (in this example)

# your turn
```

## 04.3 Create a dataframe with the individual characteristics

```r
# your turn
# HINT: df_X <- # data.frame(# ADD ALL CHARACTERISTICS)
```

# 05 Define Simulation Functions

*HINT*: There is no need to make two functions for each strategy. We recommend to make one `Probs()`, one `Costs()` and one `Effs()` function and have a function argument `Trt` which you "switch" on and off (i.e TRUE/FALSE - or 0/1) for the strategy of interest.

## 05.1 Probability function

The function that updates the transition probabilities. Please make sure you incorporate the time dependency

```
# your turn
# HINT: In this function you have to incorporate age specific mortality
# and incorporate the change in probability of the years spend in the sick state
```

## 05.2 Cost function

The `Costs` function estimates the costs at every cycle.

```
# your turn
# Make sure you incorporate the cost of the treatment in the treatment strategy for both sick and sicke
```

## 05.3 Health outcome function

The `Effs` function to update the utilities at every cycle.

```
# your turn
# HINT: Make sure you incorporate the treatment effect modifier
# HINT: Remember treatment improves the quality of life for those in the Sick (S1) state but not for th
```

## 05.4 The Microsimulation function

You need to develop the main function `MicroSim()` that runs the microsimulation.

```
# your turn
# HINT: Build your own `MicroSim` function here that calls the Probs(),  Effs() and Costs() functions a
```

# 06 Run Microsimulation

You have to run the `Microsim()` function twice. Once for the treatment strategy and once of the no-treatment strategy, as follows:

- `outcomes_no_trt  <- MicroSim(n_i, df_X, Trt = FALSE, seed = 1)`
- `outcomes_trt     <- MicroSim(n_i, df_X, Trt = TRUE,  seed = 1)`

# 07 Visualize results

```r
# your turn
# HINT: use functions from the example code
```

## 08 Cost Effectiveness Analysis

```r
# store the mean costs of each strategy in a new variable v_C (vector of costs)
# remove # below
# v_C <- c(outcomes_no_trt$tc_hat, outcomes_trt$tc_hat)

# store the mean QALYs of each strategy in a new variable v_E (vector of effects)
# remove # below
# v_E <- c(outcomes_no_trt$te_hat, outcomes_trt$te_hat)


# remove # below
# use dampack to calculate the ICER
# calculate_icers(cost       = v_C,
#                 effect     = v_E,
#                 strategies = v_names_str)
```