Survival Analysis - Sick-Sicker model

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. The Hospital for Sick Children, Toronto and 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. doi:10.1177/0272989X19893973

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

01 Load packages

```r
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", "gems", "flexsurv", "survHE", "ggplot2", "msm", "igraph", "mstate",
# load (install if required) packages from GitHub
# install_github("DARTH-git/dampack", force = TRUE) Uncomment if there is a newer version
# install_github("DARTH-git/darthtools", force = TRUE) Uncomment if there is a newer version
p_load_gh("DARTH-git/dampack", "DARTH-git/darthtools")
```

02 Load functions

```r
# No function needed
```

03 Input model parameters

```r
set.seed(1)                          # set the seed
v_names_states   <- c("H", "S1", "S2", "D")  # the model states names

# Model structure
n_t   <- 30                          # time horizon, 30 cycles
c_l   <- 1
n_i   <- 100000                      # number of simulated individuals
n_states   <- length(v_names_states) # the number of health states
d_r   <- 0.03                        # discount rate of 3% per cycle
v_dw  <- 1 / ((1 + d_r) ^ (0:n_t))   # discount weight
v_names_str <- c("no treatment", "treatment")  # strategy names

# Event probabilities (per cycle)
# Annual transition probabilities
p_HS1 <- 0.15  # probability of becoming sick when healthy

# Annual probabilities of death
# load age dependent probability
# or use "../data/" if you have a datafolder
p_mort   <- read.csv("mortProb_age.csv")
# load age distribution
dist_Age <- read.csv("MyPopulation-AgeDistribution.csv")

# 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(er) and being treated

times    <- seq(0, n_t, c_l)  # the cycles in years
```

04 Sample individual level characteristicS

04.1 Static characteristics

```
set.seed(2019)    # set the seed for the simulation of individual characteristics
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
v_age0   <- sample(x = dist_Age$age, prob = dist_Age$prop, size = n_i, replace = TRUE)

# store the information at baseline into a data frame.
df_X     <- data.frame(ID = 1:n_i, x = v_x, Age = v_age0)
```

Survival analysis component

```
# load the Sicker data
data_long <- read.csv("data_long_Sicker.csv", row.names = 1)
head(data_long)

# Multi-state models can be fitted independently for each transition. This is more flexible!

# Create subsets for each transition
data_S1H  <- subset(data_long, trans == 1)
data_S1S2 <- subset(data_long, trans == 2)
data_S1D  <- subset(data_long, trans == 3)
data_S2D  <- subset(data_long, trans == 4)

# fit independent models for each transition and pick the one with the lowest AIC
fit_S1H  <- fit.fun(time = "time", status = "status", data = data_S1H, times = times,
                    extrapolate = F)
fit_S1S2 <- fit.fun(time = "time", status = "status", data = data_S1S2, times = times,
                    extrapolate = F)
fit_S1D  <- fit.fun(time = "time", status = "status", data = data_S1D, times = times,
                    extrapolate = F)
fit_S2D  <- fit.fun(time = "time", status = "status", data = data_S2D, times = times,
                    extrapolate = F)

best.fit_S1H  <- fit_S1H$model.objects$models[["Royston-Parmar"]]
best.fit_S1S2 <- fit_S1S2$model.objects$models[["log-Logistic"]]
best.fit_S1D  <- fit_S1D$model.objects$models[["Exponential"]]
best.fit_S2D  <- fit_S2D$model.objects$models[["Weibull (AFT)"]]
```

```r
# Extract transition probabilities from the best fitting models
p_S1H  <- trans_prob(surv_prob(best.fit_S1H))
p_S1S2 <- trans_prob(surv_prob(best.fit_S1S2))
p_S1D  <- trans_prob(surv_prob(best.fit_S1D))
p_S2D  <- trans_prob(surv_prob(best.fit_S2D))
```

04.2 Dynamic characteristics

```r
# Specify the initial health state of the individuals
# everyone begins in the healthy state (in this example)
v_M_init   <- rep("H", n_i)  # a vector with the initial health state for all individuals
v_Ts1_init <- rep(0, n_i)    # a vector with the time of being sick at the start of the model
v_Ts2_init <- rep(0, n_i)    # a vector with the time of being sick at the start of the model
```

05 Define Simulation Functions

05.1 Probability function

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

```r
Probs <- function(M_t, df_X, v_Ts1, v_Ts2, t) {
  # Arguments:
    # M_t: health state occupie at cycle t (character variable)
    # v_Ts: time an individual is sick
    # t:     current cycle
  # Returns:
    # transition probabilities for that cycle

  # create matrix of state transition probability vectors
  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, p_mort, by = c("Age"))
  p_HD     <- p_HD_all[M_t == "H", "p_HD"]

  # update the v_p with the appropriate probabilities
  # transition probabilities when healthy
  m_p_t[, M_t == "H"]  <- rbind((1 - p_HD) * (1 - p_HS1),
                                (1 - p_HD) *      p_HS1 ,
                                                      0,
                                     p_HD               )
  # transition probabilities when sick
  m_p_t[, M_t == "S1"] <- rbind(p_S1H[v_Ts1], 1 - p_S1H[v_Ts1] - p_S1S2[v_Ts1] -
                                p_S1D[v_Ts1],    p_S1S2[v_Ts1],   p_S1D[v_Ts1])
  # transition probabilities when sicker
  m_p_t[, M_t == "S2"] <- rbind(0, 0, 1 - p_S2D[v_Ts2], p_S2D[v_Ts2])
  # transition probabilities when dead
  m_p_t[, M_t == "D"]  <- rbind(0, 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, Trt = FALSE) {
  # M_t: health state occupied by individual i at cycle t (character variable)
  # Trt:  is the individual being treated? (default is FALSE)

  c_t <- vector("numeric", n_i)              # create the cost variable
  c_t[M_t == "H"]  <- c_H                    # update the cost if healthy
  c_t[M_t == "S1"] <- c_S1 + c_Trt * Trt     # update the cost if sick conditional
                                             # on treatment
  c_t[M_t == "S2"] <- c_S2 + c_Trt * Trt     # update the cost if sicker conditional
                                             # on treatment
  c_t[M_t == "D"]  <- c_D                    # update the cost if dead

  return(c_t)                                        # return the costs
}
```

05.3 Health outcome function

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

```
Effs <- function (M_t, df_X, Trt = FALSE, cl = 1) {
  # M_it: health state occupied by individual i at cycle t (character variable)
  # df_X: individual characteristics including Age, Sex and the effect modifier of
  #       the treatment effect
  # Trt:  is the individual treated? (default is FALSE)
  # cl:   cycle length (default is 1)

  u_t <- 0                                   # by default the utility for everyone is zero
  u_t[M_t == "H"]  <- u_H                    # update the utility if healthy
  u_t[M_t == "S1" & Trt == FALSE] <- u_S1    # update the utility if sick
  # update the utility if sick but on treatment (adjust for individual effect modifier)
  u_t[M_t == "S1" & Trt == TRUE]  <- u_Trt * df_X$x[M_t == "S1"]
  u_t[M_t == "S2"] <- u_S2                   # update the utility if sicker
  u_t[M_t == "D"]  <- u_D                    # update the utility if dead

  QALYs <-  u_t * cl  # calculate the QALYs during cycle t
  return(QALYs)       # return the QALYs
}
```

06 Run Microsimulation

```r
MicroSim <- function(n_i, df_X , Trt = FALSE, seed = 1) {
  # Arguments:
    # n_i:     number of individuals
    # df_X     data frame with individual data
    # Trt:     is this the individual receiving treatment? (default is FALSE)
    # seed:    default is 1

  set.seed(seed)     # set the seed

  n_states <- length(v_names_states) # the number of health states

  # 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 <- m_Ts <-  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 at cycle 0 for individual i
  v_Ts1      <- v_Ts1_init  # initialize time since illnes onset for individual i
  v_Ts2      <- v_Ts2_init  # initialize time since illnes onset for individual i

  m_C[, 1]   <- Costs(m_M[, 1], Trt)        # calculate costs per individual during cycle 0
  m_E[, 1]   <- Effs (m_M[, 1], df_X, Trt)  # calculate QALYs per individual 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, v_Ts1, v_Ts2, t)
    # sample the current health state and store that state in matrix m_M
    m_M[, t + 1]  <- samplev(m_p)
    # calculate costs per individual during cycle t + 1
    m_C[, t + 1]  <- Costs(m_M[, t + 1], Trt)
    # calculate QALYs per individual during cycle t + 1
    m_E[, t + 1]  <- Effs(m_M[, t + 1], df_X, Trt)

    v_Ts1 <- if_else(m_M[, t + 1] == "S1", v_Ts1 + 1, 0)
    v_Ts2 <- if_else(m_M[, t + 1] == "S2", v_Ts2 + 1, 0)
    df_X$Age[m_M[, t + 1] != "D"]  <- df_X$Age[m_M[, t + 1] != "D"] + 1

    # 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_dw  # total (discounted) cost per individual
  te <- m_E %*% v_dw  # 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
# In this example the outcomes are of the simulation are stored in the variables `outcomes_no_tr` and `

# Run the simulation for both no treatment and treatment options
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

```
# No treatment
plot(density(outcomes_no_trt$tc), main = paste("Total cost per person"), xlab = "Cost ($)")
plot(density(outcomes_no_trt$te), main = paste("Total QALYs per person"), xlab = "QALYs")
plot_trace_microsim(outcomes_no_trt$m_M)  # health state trace

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

08 Cost Effectiveness Analysis

```
# store the mean costs of each strategy in a new variable C (vector of costs)
v_C <- c(outcomes_no_trt$tc_hat, outcomes_trt$tc_hat)
# store the mean QALYs of each strategy in a new variable E (vector of effects)
v_E <- c(outcomes_no_trt$te_hat, outcomes_trt$te_hat)

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