PSA: Markov Sick-Sicker model in R

with age-specific mortality

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 Making. 2020 Online first. 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) # use this package to conveniently
# load (install if required) packages from CRAN
p_load("here", "dplyr", "devtools", "scales", "ellipse", "ggplot2", "lazyeval", "igraph", "truncnorm",
# 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

```
# No functions needed
```

03 Input model parameters

```
# Strategy names
v_names_str <- c("No Treatment", "Treatment")

# Number of strategies
n_str <- length(v_names_str)

# Markov model parameters
age     <- 25                       # age at baseline
max_age <- 55                       # maximum age of follow up
n_t     <- max_age - age            # time horizon, number of cycles
v_n     <- c("H", "S1", "S2", "D")  # the 4 states of the model: Healthy (H), Sick (S1),
                                    # Sicker (S2), Dead (D)
n_s     <- length(v_n)              # number of health states

# Tunnels
tunnel_size <- n_t
# Sick state
Sick_tunnel <- paste("S1_", seq(1, tunnel_size), "Yr", sep = "")
### Create variables for time-dependent model
v_n_td      <- c("H", Sick_tunnel, "S2", "D")   # state names
n_s_td      <- length(v_n_td)                   # number of states

# Transition probabilities (per cycle) and hazard ratios
# Read age-specific mortality rates from csv file
lt_usa_2005 <- read.csv("HMD_USA_Mx_2015.csv")
```

```r
v_r_HD <- lt_usa_2005 %>%
  filter(Age >= age & Age <= (max_age-1)) %>%
  select(Total) %>%
  as.matrix()

p_HD    <- 1 - exp(- v_r_HD)         # probability to die when healthy
p_HS1   <- 0.15                        # probability to become sick when healthy
p_S1H   <- 0.5                         # probability to become healthy when sick

# Weibull parameters
l       <- 0.08 # scale
g       <- 1.1  # shape
# Weibull function
p_S1S2  <- l*g*(1:tunnel_size)^{g-1} # probability to become sicker when sick
                                     # (time-dependent)

hr_S1   <- 3                           # hazard ratio of death in sick vs healthy
hr_S2   <- 10                          # hazard ratio of death in sicker vs healthy
r_HD    <- - log(1 - p_HD)           # rate of death in healthy
r_S1D   <- hr_S1 * r_HD                # rate of death in sick
r_S2D   <- hr_S2 * r_HD                # rate of death in sicker
p_S1D   <- 1 - exp(-r_S1D)           # probability to die in sick
p_S2D   <- 1 - exp(-r_S2D)           # probability to die in sicker

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

# Discounting factor
d_r     <- 0.03                      # equal discount of costs and QALYs by 3%
# calculate discount weights for costs for each cycle based on discount rate d_c
v_dwc   <- 1 / (1 + d_r) ^ (0:n_t)
# calculate discount weights for effectiveness for each cycle based on discount rate d_e
v_dwe   <- 1 / (1 + d_r) ^ (0:n_t)
```

04 Define and initialize matrices and vectors

04.1 Cohort trace

```r
# create the markov trace matrix M capturing the proportion of the cohort in each state
# at each cycle
m_M_notrt <- m_M_trt <- matrix(NA,
                               nrow = n_t + 1, ncol = n_s_td,
```

```
                                dimnames = list(paste("cycle", 0:n_t, sep = " "), v_n_td))

head(m_M_notrt) # show first 6 rows of the matrix

# The cohort starts as healthy
# initialize first cycle of Markov trace accounting for the tunnels
m_M_notrt[1, ] <- m_M_trt[1, ] <- c(1, rep(0, tunnel_size), 0, 0)
```

04.2 Transition probability array

```
# create the transition probability array for NO treatment
a_P_notrt <- array(0,                                          # Create 3-D array
                   dim = c(n_s_td, n_s_td, n_t),
                   dimnames = list(v_n_td, v_n_td, 0:(n_t-1))) # name dimensions
```

Fill in the transition probability array:

```
# from Healthy
a_P_notrt["H", "H", ]            <- 1 - (p_HS1 + p_HD)
a_P_notrt["H", Sick_tunnel[1], ] <- p_HS1
a_P_notrt["H", "D", ]            <- p_HD

# from Sick
for(i in 1:(tunnel_size - 1)){
  a_P_notrt[Sick_tunnel[i], "H", ]  <- p_S1H
  a_P_notrt[Sick_tunnel[i], Sick_tunnel[i + 1], ] <- 1 - (p_S1H + p_S1S2[i] + p_S1D)
  a_P_notrt[Sick_tunnel[i], "S2", ] <- p_S1S2[i]
  a_P_notrt[Sick_tunnel[i], "D", ]  <- p_S1D
}
a_P_notrt[Sick_tunnel[tunnel_size], "H", ]  <- p_S1H
a_P_notrt[Sick_tunnel[tunnel_size], Sick_tunnel[tunnel_size], ] <- 1 -
  (p_S1H + p_S1S2[tunnel_size] + p_S1D)
a_P_notrt[Sick_tunnel[tunnel_size], "S2", ] <- p_S1S2[tunnel_size]
a_P_notrt[Sick_tunnel[tunnel_size], "D", ] <- p_S1D

# from Sicker
a_P_notrt["S2", "S2", ] <- 1 - p_S2D
a_P_notrt["S2", "D", ]  <- p_S2D

# from Dead
a_P_notrt["D", "D", ] <- 1

# Check if transition matrix is valid (i_e_, each row should add up to 1)
valid <- apply(a_P_notrt, 3, function(x) sum(rowSums(x))==n_s_td)
if (!isTRUE(all.equal(as.numeric(sum(valid)), as.numeric(n_t)))) {
  stop("This is not a valid transition Matrix")
}

# create transition probability matrix for treatment same as NO treatment
a_P_trt <- a_P_notrt
```

05 Run Markov model

```
for (t in 1:n_t){   # loop through the number of cycles
  m_M_notrt[t + 1, ] <- t(m_M_notrt[t, ]) %*% a_P_notrt[ , , t]      # estimate the Markov
                                                                     # trace for cycle the
                                                                     # next cycle (t + 1)

  m_M_trt[t + 1, ]    <- t(m_M_trt[t, ])    %*% a_P_trt[, , t]       # estimate the Markov
                                                                     # trace for cycle the
                                                                     # next cycle (t + 1)
} # close the loop

head(m_M_notrt)  # show the first 6 lines of the matrix

# create aggregated traces
m_M_td_notrt <- cbind(H = m_M_notrt[, "H"],
                      S1 = rowSums(m_M_notrt[, 2:(tunnel_size +1)]),
                      S2 = m_M_notrt[, "S2"],
                      D = m_M_notrt[, "D"])
head(m_M_td_notrt)

m_M_td_trt    <- cbind(H = m_M_trt[, "H"],
                      S1 = rowSums(m_M_trt[, 2:(tunnel_size +1)]),
                      S2 = m_M_trt[, "S2"],
                      D = m_M_trt[, "D"])
head(m_M_td_trt)
```

06 Compute and Plot Epidemiological Outcomes

06.1 Cohort trace

```
# create a plot of the data
matplot(m_M_td_notrt, type = 'l',
        ylab = "Probability of state occupancy",
        xlab = "Cycle",
        main = "Cohort Trace")
# add a legend to the graph
legend("topright", v_n, col = 1:n_s,lty = 1:n_s, bty = "n")
```

06.2 Overall Survival (OS)

```
# calculate the overall survival (OS) probability for no treatment
v_os_notrt_tunnels <- 1 - m_M_notrt[, "D"]
# alternative way of calculating the OS probability
v_os_notrt_tunnels <- rowSums(m_M_notrt[, 1:3])
# create a simple plot showing the OS
plot(age:max_age, v_os_notrt_tunnels, type = 'l',
```

```
     ylim = c(0, 1),
     ylab = "Survival probability",
     xlab = "Age",
     main = "Overall Survival Age-dependent with tunnels")
# add grid
grid(nx = n_t, ny = 10, col = "lightgray", lty = "dotted", lwd = par("lwd"),
     equilogs = TRUE)
```

06.2.1 Life Expectancy (LE)

```
v_le_tunnels <- sum(v_os_notrt_tunnels)  # summing probablity of OS over time
                                         # (i_e_ life expectancy)
```

06.3 Disease prevalence

```
v_prev_tunnels <- rowSums(m_M_td_notrt[, c("S1", "S2")]) / v_os_notrt_tunnels
plot(v_prev_tunnels,
     ylim = c(0, 1),
     ylab = "Prevalence",
     xlab = "Cycle",
     main = "Disease prevalence")
```

06.4 ratio of sick(S1) vs sicker(S2)

```
v_ratio_S1S2_tunnels <- m_M_td_notrt[, "S1"] / m_M_td_notrt[, "S2"]
plot(0:n_t, v_ratio_S1S2_tunnels,
     xlab = "Cycle",
     ylab = "Ratio S1 vs S2",
     main = "Ratio of sick and sicker",
     col  = "black", type = "l")
```

07 Compute Cost-Effectiveness Outcomes

```
# Vectors with costs and utilities by treatment
v_u_notrt   <- c(u_H, u_S1, u_S2, u_D)
v_u_trt     <- c(u_H, u_trt, u_S2, u_D)

v_c_notrt   <- c(c_H, c_S1, c_S2, c_D)
v_c_trt     <- c(c_H, c_S1 + c_trt, c_S2 + c_trt, c_D)
```

07.1 Mean Costs and QALYs for Treatment and NO Treatment

```
v_tu_notrt  <- m_M_td_notrt   %*%  v_u_notrt
v_tu_trt    <- m_M_td_trt     %*%  v_u_trt

v_tc_notrt  <- m_M_td_notrt   %*%  v_c_notrt
v_tc_trt    <- m_M_td_trt     %*%  v_c_trt
```

07.2 Discounted Mean Costs and QALYs

```
tu_d_notrt  <- t(v_tu_notrt)  %*%  v_dwe
tu_d_trt    <- t(v_tu_trt)    %*%  v_dwe

tc_d_notrt  <- t(v_tc_notrt)  %*%  v_dwc
tc_d_trt    <- t(v_tc_trt)    %*%  v_dwc


# store them into a vector
v_tc_d      <- c(tc_d_notrt, tc_d_trt)
v_tu_d      <- c(tu_d_notrt, tu_d_trt)

# Dataframe with discounted costs and effectiveness
df_ce       <- data.frame(Strategy = v_names_str,
                          Cost     = v_tc_d,
                          Effect   = v_tu_d)
df_ce
```

07.3 Compute ICERs of the Markov model

```
df_cea <- calculate_icers(cost       = df_ce$Cost,
                          effect     = df_ce$Effect,
                          strategies = df_ce$Strategy)
df_cea
```

07.4 Plot frontier of the Markov model

```
plot(df_cea, effect_units = "Quality of Life", xlim=c(17,18))
```

08 Deterministic Sensitivity Analysis

08.1 List of input parameters

Create list “l_params_all” with all input probabilities, cost and utilities.

```r
l_params_all <- as.list(data.frame(
  p_HS1   = 0.15,                # probability to become sick when healthy
  p_S1H   = 0.5,               # probability to become healthy when sick
  hr_S1   = 3,                 # hazard ratio of death in sick vs healthy
  hr_S2   = 10,                # hazard ratio of death in sicker vs healthy
  c_H     = 2000,              # cost of remaining one cycle in the healthy state
  c_S1    = 4000,              # cost of remaining one cycle in the sick state
  c_S2    = 15000,             # cost of remaining one cycle in the sicker state
  c_trt   = 12000,             # cost of treatment(per cycle)
  c_D     = 0,                 # cost of being in the death state
  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 treated
  d_e    = 0.03,               # discount factor for effectiveness
  d_c    = 0.03                # discount factor for costs
))

# store the parameter names into a vector
v_names_params <- c('p_HS1', 'p_S1H', 'hr_S1', 'hr_S2', 'c_H', 'c_S1', 'c_S2', 'c_trt', 'c_D', 'u_H', 'u
                    'u_S2', 'u_D', 'u_trt', 'd_e', 'd_c')
```

08.2 Load Sick-Sicker Markov model function

```r
source("Functions_markov_sick-sicker_tunnels.R")
# Test function
calculate_ce_out(l_params_all)
```

08.3 One-way sensitivity analysis (OWSA)

```r
v_params_owsa <-c("p_HS1", "c_trt", "u_S1", "u_trt") # vector of names of parameters of interest

# dataframe containing all parameters, their basecase values, and the min and max values of the paramet
params_all_owsa  <- data.frame(pars = v_names_params, basecase = as.numeric(l_params_all), min = rep(NA
params_all_owsa$min[params_all_owsa$pars %in% v_params_owsa] <-  c(0.05 ,  6000 , 0.65, 0.80)  # min pa
params_all_owsa$max[params_all_owsa$pars %in% v_params_owsa] <-  c(0.155, 18000 , 0.85, 0.98)  # max pa

# list of all parameters with their basecase values
params_basecase_owsa <- as.list(params_all_owsa$basecase)
names(params_basecase_owsa) <- as.character(params_all_owsa$pars)

# dataframe containing name, min and max of parameters of interest
df_params_owsa <- params_all_owsa[params_all_owsa$pars %in% v_params_owsa, !colnames(params_all_owsa) %

owsa_nmb  <- run_owsa_det(params_range     = df_params_owsa,        # parameters of interest
                          params_basecase  = params_basecase_owsa,  # dataframe containing all paramete
                          nsamp            = 100,                   # number of parameter values
                          FUN              = calculate_ce_out,      # function to compute outputs
```

```
                          outcomes         = c("NMB"),              # output to do the OWSA on
                          strategies       = v_names_str,           # names of the strategies
                          n_wtp            = 120000)                # extra argument to pass to FUN
```

08.3.1 Plot OWSA

```
plot(owsa_nmb, txtsize = 16, n_x_ticks = 5,
     facet_scales = "free") +
  theme(legend.position = "bottom")
```

08.3.2 Optimal strategy with OWSA

```
owsa_opt_strat(owsa = owsa_nmb)
```

08.3.3 Tornado plot

```
owsa_tornado(owsa = owsa_nmb)
```

08.4 Two-way sensitivity analysis (TWSA)

```
v_params_twsa     <- c("c_trt", "u_trt") # parameters of interest for TWSA
# dataframe containing all parameters, their basecase values, and the min and max values of the paramet
params_all_twsa <- data.frame(pars = v_names_params,
                              basecase = as.numeric(l_params_all),
                              min = rep(NA, length(v_names_params)),
                              max  = rep(NA, length(v_names_params)))
params_all_twsa$min[params_all_twsa$pars %in% v_params_twsa] <- c( 6000, 0.80)  # min parameter values
params_all_twsa$max[params_all_twsa$pars %in% v_params_twsa] <- c(18000, 0.98)  # max parameter values

# list of all parameters with their basecase values
params_basecase_twsa <- as.list(params_all_twsa$basecase)
names(params_basecase_twsa) <- as.character(params_all_twsa$pars)

# dataframe containing name, min and max of parameters of interest
df_params_twsa <- params_all_twsa[params_all_twsa$pars %in% v_params_twsa, !colnames(params_all_twsa) %

twsa_nmb <- run_twsa_det(params_range    = df_params_twsa,       # parameters of interest
                         params_basecase = params_basecase_twsa, # dataframe containing all parameter b
                         nsamp           = 40,                   # number of parameter values
                         FUN             = calculate_ce_out,     # function to compute outputs
                         outcomes        = c("NMB"),             # output to do the twsa on
                         strategies      = v_names_str,          # names of the strategies
                         n_wtp           = 120000)               # extra argument to pass to FUN
```

08.4.1 Plot TWSA

```
plot(twsa_nmb)
```

09 Probabilistic Sensitivity Analysis (PSA)

```
# Function to generate PSA input dataset
generate_psa_params <- function(n_sim = 1000, seed = 071818){
  set.seed(seed) # set a seed to be able to reproduce the same results
  df_psa <- data.frame(
    # Transition probabilities (per cycle)
    p_HS1   = rbeta(n_sim, 30, 170),  # probability to become sick when healthy
    p_S1H   = rbeta(n_sim, 60, 60) ,  # probability to become healthy when sick
    hr_S1   = rlnorm(n_sim, log(3),  0.01), # rate ratio of death in S1 vs healthy
    hr_S2   = rlnorm(n_sim, log(10), 0.02), # rate ratio of death in S2 vs healthy

    # State rewards
    # Costs
    c_H   = rgamma(n_sim, shape = 100, scale = 20)    , # cost of remaining one cycle in state H
    c_S1  = rgamma(n_sim, shape = 177.8, scale = 22.5), # cost of remaining one cycle in state S1
    c_S2  = rgamma(n_sim, shape = 225, scale = 66.7)  , # cost of remaining one cycle in state S2
    c_Trt = rgamma(n_sim, shape = 73.5, scale = 163.3), # cost of treatment (per cycle)
    c_D   = 0                                         , # cost of being in the death state

    # Utilities
    u_H   = rtruncnorm(n_sim, mean =    1, sd = 0.01, b = 1), # utility when healthy
    u_S1  = rtruncnorm(n_sim, mean = 0.75, sd = 0.02, b = 1), # utility when sick
    u_S2  = rtruncnorm(n_sim, mean = 0.50, sd = 0.03, b = 1), # utility when sicker
    u_D   = 0                                               , # utility when dead
    u_Trt = rtruncnorm(n_sim, mean = 0.95, sd = 0.02, b = 1), # utility when being treated
    d_e   = 0.03,  # discount factor for effectiveness
    d_c   = 0.03   # discount factor for costs
  )
  return(df_psa)
}
# Try it
generate_psa_params(10)

# Number of simulations
n_sim <- 1000

# Generate PSA input dataset
df_psa_input <- generate_psa_params(n_sim = n_sim)
# First six observations
head(df_psa_input)

# Histogram of parameters
ggplot(melt(df_psa_input, variable.name = "Parameter"), aes(x = value)) +
  facet_wrap(~Parameter, scales = "free") +
  geom_histogram(aes(y = ..density..)) +
```

```
  theme_bw(base_size = 16)

# Initialize matrices with PSA output
# Dataframe of costs
df_c <- as.data.frame(matrix(0,
                             nrow = n_sim,
                             ncol = n_str))
colnames(df_c) <- v_names_str
# Dataframe of effectiveness
df_e <- as.data.frame(matrix(0,
                             nrow = n_sim,
                             ncol = n_str))
colnames(df_e) <- v_names_str
```

09.1 Conduct probabilistic sensitivity analysis

```
# Run Markov model on each parameter set of PSA input dataset
for(i in 1:n_sim){
  l_out_temp <- calculate_ce_out(df_psa_input[i, ])
  df_c[i, ]  <- l_out_temp$Cost
  df_e[i, ]  <- l_out_temp$Effect
  # Display simulation progress
  if(i/(n_sim/10) == round(i/(n_sim/10),0)) { # display progress every 10%
    cat('\r', paste(i/n_sim * 100, "% done", sep = " "))
  }
}
```

09.2 Create PSA object for dampack

```
l_psa <- make_psa_obj(cost          = df_c,
                      effectiveness = df_e,
                      parameters    = df_psa_input,
                      strategies    = v_names_str)
```

09.2.1 Save PSA objects

```
save(df_psa_input, df_c, df_e, v_names_str, n_str,
     l_psa,
     file = "markov_sick-sicker_tunnels_PSA_dataset.RData")
```

09.3 Create probabilistic analysis graphs

```
load(file = "markov_sick-sicker_tunnels_PSA_dataset.RData")
```

Vector with willingness-to-pay (WTP) thresholds.

```
v_wtp <- seq(0, 200000, by = 10000)
```

09.3.1 Cost-Effectiveness Scatter plot

```
plot(l_psa)
```

09.4 Conduct CEA with probabilistic output

```
# Compute expected costs and effects for each strategy from the PSA
df_out_ce_psa <- summary(l_psa)

# Calculate incremental cost-effectiveness ratios (ICERs)
df_cea_psa <- calculate_icers(cost       = df_out_ce_psa$meanCost,
                              effect     = df_out_ce_psa$meanEffect,
                              strategies = df_out_ce_psa$Strategy)
df_cea_psa

# Save CEA table with ICERs
# As .RData
save(df_cea_psa,
     file = "markov_sick-sicker_tunnels_probabilistic_CEA_results.RData")
# As .csv
write.csv(df_cea_psa,
          file = "markov_sick-sicker_tunnels_probabilistic_CEA_results.csv")
```

09.4.1 Plot cost-effectiveness frontier

```
plot(df_cea_psa)
```

09.4.2 Cost-effectiveness acceptability curves (CEACs) and frontier (CEAF)

```
ceac_obj <- ceac(wtp = v_wtp, psa = l_psa)
# Regions of highest probability of cost-effectiveness for each strategy
summary(ceac_obj)
# CEAC & CEAF plot
plot(ceac_obj)
```

09.4.3 Expected Loss Curves (ELCs)

```
elc_obj <- calc_exp_loss(wtp = v_wtp, psa = l_psa)
elc_obj
# ELC plot
plot(elc_obj, log_y = FALSE)
```

09.4.4 Expected value of perfect information (EVPI)

```
evpi <- calc_evpi(wtp = v_wtp, psa = l_psa)
# EVPI plot
plot(evpi, effect_units = "QALY")
```