SA: Simple 3-state Markov model in R

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. Division of Public Administration, Center for Research and Teaching in Economics (CIDE), 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
- Alarid-Escudero F, Krijkamp EM, Enns EA, Yang A, Hunink MGM Pechlivanoglou P, Jalal H. Cohort State-Transition Models in R: A Tutorial. arXiv:200107824v2. 2020:1-48. http://arxiv.org/abs/2001.07824
- 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. 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",
# 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 function needed
```

03 Input model parameters

```
# Strategy names
v_names_str <- c("Standard of Care", "Treatment")

# Markov model parameters
v_n  <- c("Healthy", "Sick", "Dead")  # state names
n_t  <- 60                            # number of cycles

v_init <- c("Healthy" = 1,
            "Sick"    = 0,
            "Dead"    = 0)            # initial cohort distribution (everyone allocated to the
                                      # "healthy" state)

# Transition probabilities
p_HD      <- 0.02                     # probability of dying when healthy
p_HS      <- 0.05                     # probability of becoming sick when healthy, under standard of ca
p_HS_trt  <- 0.03                     # probability of becoming sick when healthy, under treatment
p_SD      <- 0.1                      # probability of dying when sick

# Costs and utilities
c_H       <- 400                      # cost of one cycle in healthy state
c_S       <- 1000                     # cost of one cycle in sick state
c_D       <- 0                        # cost of one cycle in dead state
c_trt     <- 800                      # cost of treatment (per cycle)
u_H       <- 0.8                      # utility when healthy
u_S       <- 0.5                      # utility when sick
u_D       <- 0                        # utility when dead
```

```r
d_e       <- d_c <- 0.03              # discount rate per cycle equal discount of costs and QALYs by 3%

n_str     <- length(v_names_str)      # Number of strategies
n_states  <- length(v_n)              # number of states

# Discount weights for costs and effects
v_dwc <- 1 / (1 + d_c) ^ (0:n_t)
v_dwe <- 1 / (1 + d_e) ^ (0:n_t)
```

Draw the state-transition cohort model

```r
m_P_diag <- matrix(0, nrow = n_states, ncol = n_states, dimnames = list(v_n, v_n))
m_P_diag["Healthy", "Sick" ]     = ""
m_P_diag["Healthy", "Dead" ]     = ""
m_P_diag["Healthy", "Healthy" ]  = ""
m_P_diag["Sick"   , "Dead" ]     = ""
m_P_diag["Sick"   , "Sick" ]     = ""
m_P_diag["Dead"   , "Dead" ]     = ""
layout.fig <- c(2, 1)
plotmat(t(m_P_diag), t(layout.fig), self.cex = 0.5, curve = 0, arr.pos = 0.8,
        latex = T, arr.type = "curved", relsize = 0.85, box.prop = 0.8,
        cex = 0.8, box.cex = 0.7, lwd = 1)
```

04 Define and initialize matrices and vectors

04.1 Cohort trace

```r
# create the cohort trace
m_M <- m_M_trt <-  matrix(NA,
                          nrow = n_t + 1 ,  # create Markov trace (n.t + 1 because R doesn't
                                            # understand Cycle 0)
                          ncol = n_states,
                         dimnames = list(0:n_t, v_n))

m_M[1, ] <- m_M_trt[1, ] <- v_init         # initialize first cycle of Markov trace
```

04.2 Transition probability matrix

```r
# create the transition probability matrices
m_P  <- m_P_trt <- matrix(0,
                    nrow = n_states, ncol = n_states,
                    dimnames = list(v_n, v_n))    # name the columns and rows of the transition
                                                  # probability matrices
m_P
```

Fill in the transition probability matrix:

```r
# from Healthy
m_P["Healthy", "Healthy"] <- (1 - p_HD) * (1 - p_HS)
m_P["Healthy", "Sick"]    <- (1 - p_HD) * p_HS
m_P["Healthy", "Dead"]    <- p_HD

# from Sick
m_P["Sick", "Sick"] <- 1 - p_SD
m_P["Sick", "Dead"] <- p_SD

# from Dead
m_P["Dead", "Dead"] <- 1

# Under treatment
m_P_trt <- m_P
m_P_trt["Healthy", "Healthy"] <- (1 - p_HD) * (1 - p_HS_trt)
m_P_trt["Healthy", "Sick"]    <- (1 - p_HD) * p_HS_trt

# Check that transition probabilities are in [0, 1]
check_transition_probability(m_P, verbose = TRUE)
check_transition_probability(m_P_trt, verbose = TRUE)
# Check that all rows sum to 1
check_sum_of_transition_array(m_P, n_states = n_states, n_cycles = n_t, verbose = TRUE)
check_transition_probability(m_P_trt, verbose = TRUE)
```

05 Run Markov model

```r
for (t in 1:n_t){                                # loop through the number of cycles
  m_M[t + 1, ]     <- m_M[t, ]     %*% m_P      # estimate the state vector for the next cycle (t + 1)
  m_M_trt[t + 1, ] <- m_M_trt[t, ] %*% m_P_trt  # for treatment
}
```

06 Compute and Plot Epidemiological Outcomes

06.1 Cohort trace

Standard of Care:

```r
matplot(m_M, type = 'l',
        ylab = "Probability of state occupancy",
        xlab = "Cycle",
        main = "Cohort Trace - standard of care", lwd = 3)  # create a plot of the data
legend("right", v_n, col = c("black", "red", "green"),
       lty = 1:3, bty = "n")                                # add a legend to the graph

abline(v = which.max(m_M[, "Sick"]), col = "gray")          # plot a vertical line that helps identifyi
```

Treatment:

```
matplot(m_M_trt, type = 'l',
        ylab = "Probability of state occupancy",
        xlab = "Cycle",
        main = "Cohort Trace - treatment", lwd = 3)     # create a plot of the data
legend("right", v_n, col = c("black", "red", "green"),
       lty = 1:3, bty = "n")                            # add a legend to the graph

abline(v = which.max(m_M[, "Sick"]), col = "gray")      # plot a vertical line that helps identifying a
```

06.2 Overall Survival (OS)

Standard of Care:

```
v_os <- 1 - m_M[, "Dead"]             # calculate the overall survival (OS) probability
v_os <- rowSums(m_M[, 1:2])           # alternative way of calculating the OS probability

plot(v_os, type = 'l',
     ylim = c(0, 1),
     ylab = "Survival probability",
     xlab = "Cycle",
     main = "Overall Survival")       # create a simple plot showing the OS

# add grid
grid(nx = n_t, ny = 10, col = "lightgray", lty = "dotted", lwd = par("lwd"),
     equilogs = TRUE)
```

Treatment:

```
v_os_trt <- 1 - m_M_trt[, "Dead"]     # calculate the overall survival (OS) probability
v_os_trt <- rowSums(m_M_trt[, 1:2])   # alternative way of calculating the OS probability

plot(v_os_trt, type = 'l',
     ylim = c(0, 1),
     ylab = "Survival probability",
     xlab = "Cycle",
     main = "Overall Survival")       # create a simple plot showing the OS

# 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     <- sum(v_os)      # summing probability of OS over time (i.e. life expectancy)
v_le_trt <- sum(v_os_trt)  # summing probability of OS over time (i.e. life expectancy), treatment
```

06.3 Disease prevalence

Standard of Care:

```r
v_prev <- m_M[, "Sick"]/v_os
plot(v_prev,
     ylim = c(0, 1),
     ylab = "Prevalence",
     xlab = "Cycle",
     main = "Disease prevalence")
```

Treatment:

```r
v_prev_trt <- m_M_trt[, "Sick"]/v_os_trt
plot(v_prev_trt,
     ylim = c(0, 1),
     ylab = "Prevalence",
     xlab = "Cycle",
     main = "Disease prevalence")
```

07 Compute Cost-Effectiveness Outcomes

07.1 Mean Costs and QALYs

```r
# per cycle
# calculate expected costs by multiplying m_M with the cost vector for the different
# health states
v_tc     <- m_M     %*% c(c_H, c_S, c_D)          # Standard of Care
v_tc_trt <- m_M_trt %*% c(c_H, c_S + c_trt, c_D)  # Treatment
# calculate expected QALYs  by multiplying m_M with the utilities for the different
# health states
v_tu     <- m_M     %*% c(u_H, u_S, u_D)          # Standard of Care
v_tu_trt <- m_M_trt %*% c(u_H, u_S, u_D)          # Treatment
```

07.2 Discounted Mean Costs and QALYs

```r
# Discount costs by multiplying the cost vector with discount weights
tc_d     <-  t(v_tc)     %*% v_dwc      # Standard of Care
tc_d_trt <-  t(v_tc_trt) %*% v_dwc      # Treatment
# Discount QALYS by multiplying the QALYs vector with discount weights
tu_d     <-  t(v_tu)     %*% v_dwe      # Standard of Care
tu_d_trt <-  t(v_tu_trt) %*% v_dwe      # Treatment

# store them into a vector
v_tc_d   <- c(tc_d, tc_d_trt)
v_tu_d   <- c(tu_d, 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 = "QALYs", xlim = c(10, 12))
```

08 Deterministic Sensitivity Analysis

08.1 List of input parameters

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

```
l_params_all <- as.list(data.frame(
  p_HD     = 0.02,  # probability of dying when healthy
  p_HS     = 0.05,  # probability of becoming sick when healthy, conditioned on not dying
  p_HS_trt = 0.03,  # probability of becoming sick when healthy, conditioned on not dying
  p_SD     = 0.1,   # probability of dying when sick
  c_H      = 400,   # cost of one cycle in healthy state
  c_S      = 1000,  # cost of one cycle in sick state
  c_D      = 0,     # cost of one cycle in dead state
  c_trt    = 800,   # cost of treatment (per cycle)
  u_H      = 0.8,   # utility when healthy
  u_S      = 0.5,   # utility when sick
  u_D      = 0,     # utility when dead
  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 <- names(l_params_all)
```

08.2 Load Sick-Sicker Markov model function

```
source("Functions_markov_3state.R")
# Test function
calculate_ce_out(l_params_all)
```

08.3 One-way sensitivity analysis (OWSA)

```
options(scipen = 999) # disabling scientific notation in R
# dataframe containing all parameters, their base case values, and the min and
# max values of the parameters of interest
df_params_owsa <- data.frame(pars = c("c_trt", "c_S", "u_H"),
                             min  = c(300 , 500, 0.7),   # min parameter values
                             max  = c(1200, 2000, 0.9)   # max parameter values
                             )

owsa_nmb  <- run_owsa_det(params_range     = df_params_owsa,    # dataframe with parameters for OWSA
                          params_basecase  = l_params_all,      # list with all parameters
                          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            = 2000)              # extra argument to pass to FUN
```

08.3.1 Plot OWSA

```
plot(owsa_nmb, txtsize = 10, n_x_ticks = 4,
     facet_scales = "free") +
     theme(legend.position = "bottom")
```

08.3.2 Optimal strategy with OWSA

Only useful if we have more than one strategies to compare.

```
owsa_opt_strat(owsa = owsa_nmb, txtsize = 10)
```

08.3.3 Tornado plot

```
owsa_tornado(owsa = owsa_nmb, txtsize = 11)
```

08.4 Two-way sensitivity analysis (TWSA)

```
# dataframe containing all parameters, their basecase values, and the min and
# max values of the parameters of interest
df_params_twsa <- data.frame(pars = c("c_trt", "u_H"),
                             min  = c(300, 0.7),  # min parameter values
                             max  = c(1200, 0.9)  # max parameter values
                             )

twsa_nmb <- run_twsa_det(params_range    = df_params_twsa,    # dataframe with parameters for TWSA
                         params_basecase = l_params_all,      # list with all parameters
```

```
                          nsamp           = 40,                 # number of parameter values
                          FUN             = calculate_ce_out,   # function to compute outputs
                          outcomes        = "NMB",              # output to do the TWSA on
                          strategies      = v_names_str,        # names of the strategies
                          n_wtp           = 2000)               # extra argument to pass to FUN
```

08.4.1 Plot TWSA

Only useful if we have more than one strategies to compare.

```
plot(twsa_nmb)
```

09 Probabilistic Sensitivity Analysis (PSA)

```
# Function to generate PSA input dataset
gen_psa <- 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)
    # probability to become sick when healthy
    p_HS      = rbeta(n_sim, shape1 = 24, shape2 = 450),
    p_HS_trt = rbeta(n_sim, shape1 = 9,  shape2 = 281),    # under treatment
    # probability of dying when healthy
    p_HD      = rbeta(n_sim, shape1 = 16, shape2 = 767),
    # probability of dying when sick
    p_SD      = rbeta(n_sim, shape1 = 22.4, shape2 = 201.6),

    # Cost vectors with length n_sim
    # cost of remaining one cycle in state H
    c_H       = rgamma(n_sim, shape = 16, scale = 25),
    # cost of remaining one cycle in state S1
    c_S       = rgamma(n_sim, shape = 100, scale = 10),
    # cost of being in the death state
    c_D       = 0,
    # cost of treatment (per cycle)
    c_trt     = rgamma(n_sim, shape = 64, scale = 12.5),

    # Utility vectors with length n_sim
    # utility when healthy
    u_H       = rbeta(n_sim, shape1 =  50.4, shape2 = 12.6),
    # utility when sick
    u_S       = rbeta(n_sim, shape1 = 49.5, shape2 = 49.5),
    # utility when dead
    u_D       = 0
  )
  return(df_psa)
}
# Try it
gen_psa(10)
```

```r
# Number of simulations
n_sim <- 1000

# Generate PSA input dataset
df_psa_input <- gen_psa(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) +
       theme(axis.text = element_text(size=8))

# Initialize dataframes 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

```r
# 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

```r
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_3state_PSA_dataset.RData")
```

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

```
v_wtp <- seq(0, 5000, by = 1000)
```

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_3state_probabilistic_CEA_results.RData")
# As .csv
write.csv(df_cea_psa,
          file = "markov_3state_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)

The expected loss is the the quantification of the foregone benefits when choosing a suboptimal strategy given current evidence.

```
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")
```