Cohort State-Transition Sick-Sicker model in R

The DARTH workgroup

Please cite our publications when using this code:

- Alarid-Escudero F, Krijkamp EM, Enns EA, Yang A, Hunink MGM Pechlivanoglou P, Jalal H. An Introductory Tutorial on Cohort State-Transition Models in R Using a Cost-Effectiveness Analysis Example. Medical Decision Making, 2022 (Online First):1-18. https://doi.org/10.1177/0272989X221103163

Authors: - Fernando Alarid-Escudero falarid@stanford.edu - Eline Krijkamp - Eva A. Enns - Alan Yang - M.G. Myriam Hunink - Petros Pechlivanoglou - Hawre Jalal

Please cite the article when using this code

To program this tutorial we used: R version 4.0.5 (2021-03-31) Platform: 64-bit operating system, x64-based processor Running under: Mac OS 12.2.1 RStudio: Version 1.4.1717 2009-2021 RStudio, Inc

Implements a time-independent Sick-Sicker cSTM model that evaluates four strategies: - Standard of Care (SoC): best available care for the patients with the disease. This scenario reflects the natural history of the disease
progression. - Strategy A: treatment A is given to patients in the Sick and Sicker states, but does only improves the quality of life of those in the Sick state. - Strategy B: treatment B is given to all sick patients and reduces disease progression from the Sick to Sicker state. - Strategy AB: This strategy combines treatment A and treatment B. The disease progression is reduced and individuals in the Sick state have an improved quality of life.

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

```
# use this package to conveniently install other packages
if (!require('pacman')) install.packages('pacman'); library(pacman)

# load (install if required) packages from CRAN
p_load("devtools","diagram","dampack","scales")

# 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

```
# all functions are in the darthtools package
```

03 Model parameters

03.1 Define model input parameters

```
## General setup
cycle_length <- 1         # cycle length in years (use 1/12 for monthly)
n_age_init   <- 25        # age at baseline
n_age_max    <- 100       # maximum age of follow up
n_time_horizon_yr <- n_age_max - n_age_init    # time horizon (in years)
n_cycles     <- n_time_horizon_yr/cycle_length # time horizon, number of cycles
v_names_states <- c("H",  # the 4 health states of the model:
                    "S1", # Healthy (H), Sick (S1), Sicker (S2), Dead (D)
                    "S2",
                    "D")
n_states <- length(v_names_states)    # number of health states

### Discounting factors
d_c <- 0.03 # annual discount rate for costs
d_e <- 0.03 # annual discount rate for QALYs

### Strategies
v_names_str <- c("Standard of Care",  # store the strategy names
                 "Strategy AB")
n_str       <- length(v_names_str)    # number of strategies
```

```r
### Within-cycle correction (WCC) using Simpson's 1/3 rule
v_wcc <- gen_wcc(n_cycles = n_cycles,  method = "Simpson1/3")

### Transition rates (annual), and hazard ratios (HRs)
r_HD    <- 0.002 # constant annual rate of dying when Healthy (all-cause mortality)
r_HS1   <- 0.15  # constant annual rate of becoming Sick when Healthy
r_S1H   <- 0.5   # constant annual rate of becoming Healthy when Sick
r_S1S2  <- 0.105 # constant annual rate of becoming Sicker when Sick
hr_S1   <- 3     # hazard ratio of death in Sick vs Healthy
hr_S2   <- 10    # hazard ratio of death in Sicker vs Healthy

### Effectiveness of treatment AB
hr_S1S2_trtAB <- 0.6  # hazard ratio of becoming Sicker when Sick under treatment AB

### State rewards
#### Costs
c_H     <- 2000  # annual cost of being Healthy
c_S1    <- 4000  # annual cost of being Sick
c_S2    <- 15000 # annual cost of being Sicker
c_D     <- 0     # annual cost of being dead
c_trtAB <- 25000 # annual cost of receiving treatment AB
#### Utilities
u_H     <- 1     # annual utility of being Healthy
u_S1    <- 0.75  # annual utility of being Sick
u_S2    <- 0.5   # annual utility of being Sicker
u_D     <- 0     # annual utility of being dead
u_trtAB <- 0.95  # annual utility when receiving treatment AB
```

03.2 Calculate internal model parameters

```r
### Cycle-specific discount weight for costs and effects
v_dwc   <- 1 / ((1 + (d_e * cycle_length)) ^ (0:n_cycles))
v_dwe   <- 1 / ((1 + (d_c * cycle_length)) ^ (0:n_cycles))

### Cycle-specific transition probabilities
# compute mortality rates
r_S1D   <- r_HD * hr_S1 # annual mortality rate in the Sick state
r_S2D   <- r_HD * hr_S2 # annual mortality rate in the Sicker state
# transform rates to probabilities
p_HS1   <- rate_to_prob(r = r_HS1,  t = cycle_length) # constant annual probability of becoming Sick wh
p_S1H   <- rate_to_prob(r = r_S1H,  t = cycle_length) # constant annual probability of becoming Healthy
p_S1S2  <- rate_to_prob(r = r_S1S2, t = cycle_length) # constant annual probability of becoming Sicker
p_HD    <- rate_to_prob(r = r_HD,   t = cycle_length) # annual mortality risk in the Healthy state
p_S1D   <- rate_to_prob(r = r_S1D,  t = cycle_length) # annual mortality risk in the Sick state
p_S2D   <- rate_to_prob(r = r_S2D,  t = cycle_length) # annual mortality risk in the Sicker state

## Annual transition probability of becoming Sicker when Sick for treatment AB
# Apply hazard ratio to rate to obtain transition rate of becoming Sicker when
# Sick for treatment AB
r_S1S2_trtAB <- r_S1S2 * hr_S1S2_trtAB
# Transform rate to probability to become Sicker when Sick under treatment AB conditional on surviving
```

```
p_S1S2_trtAB <- rate_to_prob(r = r_S1S2_trtAB, t = cycle_length)

### Calculate cycle-specific state rewards
#### Costs
v_c_SoC  <- c(c_H, c_S1, c_S2, c_D) * cycle_length                      # Standard of Care
v_c_strAB <- c(c_H, c_S1 + c_trtAB, c_S2 + c_trtAB, c_D) * cycle_length # Treatment AB
#### QALYs
v_q_SoC  <-  c(u_H, u_S1, u_S2, u_D) * cycle_length    # Standord of Care
v_q_strAB <- c(u_H, u_trtAB, u_S2, u_D) * cycle_length # Treatment AB
```

04 Construct state-transition models

```
m_P_diag <- matrix(0, nrow = n_states, ncol = n_states, dimnames = list(v_names_states, v_names_states)
m_P_diag["H" , "S1"] = ""
m_P_diag["H" , "D" ] = ""
m_P_diag["H" , "H" ] = ""
m_P_diag["S1", "H" ] = ""
m_P_diag["S1", "S2"] = ""
m_P_diag["S1", "D" ] = ""
m_P_diag["S1", "S1"] = ""
m_P_diag["S2", "D" ] = ""
m_P_diag["S2", "S2"] = ""
m_P_diag["D", "D"  ] = ""
layout.fig <- c(3, 1)

plotmat(t(m_P_diag), t(layout.fig), self.cex = 0.5, curve = 0, arr.pos = 0.7,
        latex = T, arr.type = "curved", relsize = 0.9, box.prop = 0.8,
        cex = 0.8, box.cex = 0.9, lwd = 1)
```

04.1 Initial state vector

```
# All starting healthy
v_m_init <- c(H = 1, S1 = 0, S2 = 0, D = 0) # initial state vector
v_m_init
```

04.2 Initialize cohort traces

```
### Initialize cohort trace for SoC
m_M_SoC <- matrix(NA,
              nrow = (n_cycles + 1), ncol = n_states,
              dimnames = list(0:n_cycles, v_names_states))
# Store the initial state vector in the first row of the cohort trace
m_M_SoC[1, ] <- v_m_init

### Initialize cohort trace for strategies  AB
# Structure and initial states are the same as for SoC
m_M_strAB <- m_M_SoC # Strategy AB
```

04.3 Create transition probability matrices

```r
## Create transition probability matrices for strategy SoC
### Initialize transition probability matrix for strategy SoC
# All transitions to a non-death state are assumed to be conditional on survival
m_P_SoC <- matrix(0,
              nrow = n_states, ncol = n_states,
              dimnames = list(v_names_states,
                              v_names_states)) # define row and column names
### Fill in matrix
# your turn #
# From H
m_P_SoC["H", "H"]   <- (1 - p_HD) * (1 - p_HS1)
m_P_SoC["H", "S1"]  <- (1 - p_HD) *      p_HS1
m_P_SoC["H", "D"]   <-      p_HD
# From S1
m_P_SoC["S1", "H"]  <- (1 - p_S1D) *       p_S1H
m_P_SoC["S1", "S1"] <- (1 - p_S1D) * (1 - (p_S1H + p_S1S2))
m_P_SoC["S1", "S2"] <- (1 - p_S1D) *               p_S1S2
m_P_SoC["S1", "D"]  <-      p_S1D
# From S2
m_P_SoC["S2", "S2"] <- 1 - p_S2D
m_P_SoC["S2", "D"]  <-     p_S2D
# From D
m_P_SoC["D", "D"]   <- 1


### Initialize transition probability matrix for strategy AB
m_P_strAB <- m_P_SoC
# your turn #
# Update only transition probabilities from S1 involving p_S1S2
m_P_strAB["S1", "S1"] <- (1 - p_S1D) * (1 - (p_S1H + p_S1S2_trtAB))
m_P_strAB["S1", "S2"] <- (1 - p_S1D) *               p_S1S2_trtAB


## Check if transition probability matrices are valid
### Check that transition probabilities are [0, 1]
check_transition_probability(m_P_SoC, verbose = TRUE) # m_P >= 0 && m_P <= 1
check_transition_probability(m_P_strAB, verbose = TRUE) # m_P_strAB >= 0 && m_P_strAB <= 1
### Check that all rows sum to 1
check_sum_of_transition_array(m_P_SoC, n_states = n_states, n_cycles = n_cycles, verbose = TRUE) # rowS
check_sum_of_transition_array(m_P_strAB, n_states = n_states, n_cycles = n_cycles, verbose = TRUE) # ro
```

05 Run Markov model

```r
## Loop over time
# Calculating cohort state based on previous state and transition matrix
# your turn #
for(t in 1:n_cycles){
  # For SoC
  m_M_SoC[t + 1, ] <- m_M_SoC[t, ] %*% m_P_SoC
  # For strategy AB
  m_M_strAB[t + 1, ] <- m_M_strAB[t, ] %*% m_P_strAB
}
```

06 Plot Outputs

06.1 Plot the cohort trace for strategies SoC and AB

```
# your turn #
plot_trace(m_M_SoC)
plot_trace(m_M_strAB)
```

07 Compute expected outcomes (costs, QALYs)

```
# Create empty vectors to store total costs and QALYs
v_tot_cost <- v_tot_qaly <- vector(mode = "numeric", length = n_str)
names(v_tot_qaly) <- names(v_tot_cost) <- v_names_str

### Expected discounted cost for each strategy
# your turn #
v_tot_cost["Standard of Care"] <- t(m_M_SoC %*% v_c_SoC) %*% (v_dwe * v_wcc)
v_tot_cost["Strategy AB"] <- t(m_M_strAB %*% v_c_strAB) %*% (v_dwe * v_wcc)

### Expected discounted QALYs for each strategy
# your turn #
v_tot_qaly["Standard of Care"] <- t(m_M_SoC %*% v_q_SoC) %*% (v_dwe * v_wcc)
v_tot_qaly["Strategy AB"] <- t(m_M_strAB %*% v_q_strAB) %*% (v_dwe * v_wcc)
```

08 Cost-effectiveness analysis (CEA)

```
## Incremental cost-effectiveness ratios (ICERs)
# your turn #
df_cea <- calculate_icers(cost       = v_tot_cost,
                          effect     = v_tot_qaly,
                          strategies = v_names_str)
df_cea
```

```
## CEA table in proper format
# your turn #
table_cea <- format_table_cea(df_cea)
table_cea
```

```r
## CEA frontier
# your turn #
plot(df_cea, label = "all", txtsize = 14) +
  expand_limits(x = max(table_cea$QALYs) + 0.1) +
  theme(legend.position = c(0.85, 0.3))
```