3-state cohort state transition model in R

with simulation time and state-residence time dependencies

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. Stanford University, Stanford, CA, USA
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 Ottawa, Ottawa, ON, Canada
6. University of Toronto, Toronto ON, Canada
7. The Hospital for Sick Children, Toronto ON, Canada

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 (Epub). https://doi.org/10.1177/0272989X221103163

- Alarid-Escudero F, Krijkamp EM, Enns EA, Yang A, Hunink MGM Pechlivanoglou P, Jalal H. A Tutorial on Time-Dependent Cohort State-Transition Models in R using a Cost-Effectiveness Analysis Example. Medical Decision Making, 2022 (Epub). https://doi.org/10.1177/0272989X221121747

- 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. 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

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

```
## Loading required package: 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
n_time_horizon_yr <- 60                                # time horizon (in years)
cycle_length      <- 1                                 # cycle length in years (use 1/12 for monthly)
n_cycles          <- n_time_horizon_yr / cycle_length  # number of cycles
v_names_cycles    <- paste("cycle", 0:n_cycles)        # cycle names
v_names_states    <- c("Healthy", "Sick", "Dead")      # state names
n_states          <- length(v_names_states)            # number of health states

### Discounting rate
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
                     "Treatment A",
                     "Treatment B")
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 probabilities

## Constant transition probabilities
# Annual probability of becoming sick from Healthy, conditional on surviving cycle
p_HS_yr_SoC  <- 0.05  # under standard of care
p_HS_yr_trtA <- 0.04  # under treatment A
p_HS_yr_trtB <- 0.02  # under treatment B

## Simulation time-dependent transition probabilities
# Annual probability of dying when healthy (increases over time)
p_HD_yr_init <- 0.001    # Initial annual probability of death
p_HD_yr_inc  <- 0.0005   # Annual increase in baseline mortality
v_p_HD_yr    <- seq(from = p_HD_yr_init,
                      by = p_HD_yr_inc,
                    length.out = n_time_horizon_yr)

## State-residence time-dependent transition probabilities
# Annual probability of dying when Sick, increases the longer one is sick, up to 10 years
n_tunnel_size_yr <- 10
# Probability of dying follows a Weibull distribution on Sick state residence time
# Weibull parameters
p_SD_scale <- 0.08
p_SD_shape <- 1.1
# Weibull function
v_p_SD_tunnels_yr <- ((1:n_tunnel_size_yr)     * p_SD_scale) ^ p_SD_shape -
                     ((1:n_tunnel_size_yr - 1) * p_SD_scale) ^ p_SD_shape

### State rewards
#### Costs
c_H_yr     <- 400   # cost of one year in healthy state
c_S_yr     <- 1000  # cost of one year in sick state
c_D_yr     <- 0     # cost of one year in dead state
c_trtA_yr  <- 800   # cost of treatment A (per year) in healthy state
c_trtB_yr  <- 1500  # cost of treatment B (per year) in healthy state
#### Utilities
u_H       <- 1     # utility when healthy
u_S       <- 0.5   # utility when sick
u_D       <- 0     # utility when dead
```

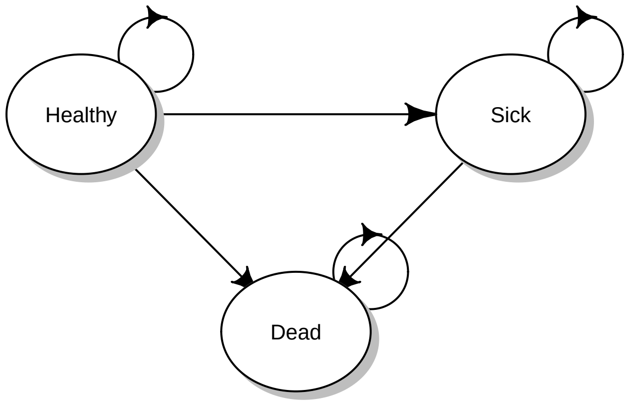
03.2 Calculate internal model parameters

```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))

### Calculate cycle-specific transition probabilities

## Constant transition probabilities
```

```
# probability of becoming sick from Healthy
p_HS_SoC  <- 1 - exp(-(-log(1 - p_HS_yr_SoC)  * cycle_length)) # Standard of Care
p_HS_trtA <- 1 - exp(-(-log(1 - p_HS_yr_trtA) * cycle_length)) # Treatment A
p_HS_trtB <- 1 - exp(-(-log(1 - p_HS_yr_trtB) * cycle_length)) # Treatment B

## Simulation time-dependent transition probabilities
# Probability of dying from Healthy state
# Expand the vector by number of cycles per year (1/cycle_length):
v_p_HD_yr_expanded <- rep(v_p_HD_yr, each = 1 / cycle_length)
# Convert to cycle-specific transition probabilities
v_p_HD <- 1 - exp(-(-log(1 - v_p_HD_yr_expanded) * cycle_length))

## State-residence time-dependent transition probabilities
# Set up cycle-specific tunnel to track time in Sick state
v_cycles_tunnel     <- 1:(n_tunnel_size_yr / cycle_length)          # tunnel state enumeration
v_names_Sick_tunnel <- paste0("Sick_", v_cycles_tunnel, "Cycle")     # names of tunnel states
n_tunnel_size       <- length(v_cycles_tunnel)                      # number of tunnel states
# Adjust model health states to account for tunnels
v_names_states_tunnels  <- c("Healthy", v_names_Sick_tunnel, "Dead") # state names with tunnels
n_states_tunnels  <- length(v_names_states_tunnels)                 # number of states with tunnels

# Probability of dying from Sick state
# Expand the vector by number of cycles per year (1/cycle_length):
v_p_SD_tunnels_yr_expanded <- rep(v_p_SD_tunnels_yr, each = 1 / cycle_length)
# Convert to cycle-specific transition probabilities
v_p_SD_tunnels <- 1 - exp(-(-log(1 - v_p_SD_tunnels_yr_expanded) * cycle_length))

### Calculate cycle-specific state rewards
#### Costs
v_c_SoC  <- c(c_H_yr, rep(c_S_yr,n_tunnel_size), c_D_yr) * cycle_length             # Standard of Care
v_c_trtA <- c(c_H_yr + c_trtA_yr, rep(c_S_yr,n_tunnel_size), c_D_yr) * cycle_length # Treatment A
v_c_trtB <- c(c_H_yr + c_trtB_yr, rep(c_S_yr,n_tunnel_size), c_D_yr) * cycle_length # Treatment B
#### QALYs
v_q_SoC  <- c(u_H, rep(u_S, n_tunnel_size), u_D) * cycle_length # Standard of Care
v_q_trtA <- v_q_trtB <- v_q_SoC             # Treatments A and B have same utilities as SoC
```

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["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)
```



04.1 Initial state vector

```r
# All starting healthy
v_m_init_tunnels <- c(1, rep(0, n_tunnel_size), 0)
```

04.2 Initialize cohort traces with tunnels

```r
### Initialize cohort trace for state-residece dependent cSTM under SoC
m_M_tunnels_SoC <- matrix(0,
                          nrow     = (n_cycles + 1), ncol = n_states_tunnels,
                          dimnames = list(0:n_cycles, v_names_states_tunnels))
# Store the initial state vector in the first row of the cohort trace
m_M_tunnels_SoC[1, ] <- v_m_init_tunnels

### Initialize cohort trace for strategies A and B
# Structure and initial states are the same as for SoC
m_M_tunnels_trtA <- m_M_tunnels_SoC # Strategy A
m_M_tunnels_trtB <- m_M_tunnels_SoC # Strategy B
```

04.3 Create transition probability arrays

```r
## Create transition probability arrays for strategy SoC
### Initialize transition probability array for strategy SoC
# All transitions to a non-death state are assumed to be conditional on survival
a_P_tunnels_SoC <- array(0,  # Create 3-D array
                       dim = c(n_states_tunnels, n_states_tunnels, n_cycles),
                       dimnames = list(v_names_states_tunnels, v_names_states_tunnels,
                                       v_names_cycles[-length(v_names_cycles)])) # name the dimensions

### Fill in array
## from Healthy
a_P_tunnels_SoC["Healthy", "Healthy", ]     <- (1 - v_p_HD) * (1 - p_HS_SoC)
a_P_tunnels_SoC["Healthy", "Sick_1Cycle", ] <- (1 - v_p_HD) *      p_HS_SoC
a_P_tunnels_SoC["Healthy", "Dead", ]        <-      v_p_HD
```

```
## from Sick
for(i in 1:(n_tunnel_size - 1)){
  a_P_tunnels_SoC[v_names_Sick_tunnel[i],
                  v_names_Sick_tunnel[i + 1], ] <- 1 - v_p_SD_tunnels[i]

  a_P_tunnels_SoC[v_names_Sick_tunnel[i],
                  "Dead", ]                     <-     v_p_SD_tunnels[i]
}
# end of tunnel, stay in final Sick state
a_P_tunnels_SoC[v_names_Sick_tunnel[n_tunnel_size],
                v_names_Sick_tunnel[n_tunnel_size], ] <- 1 - v_p_SD_tunnels[n_tunnel_size]

a_P_tunnels_SoC[v_names_Sick_tunnel[n_tunnel_size],
                "Dead", ]                             <-     v_p_SD_tunnels[n_tunnel_size]

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

## Treatment A
a_P_tunnels_trtA <- a_P_tunnels_SoC
a_P_tunnels_trtA["Healthy", "Healthy", ]     <- (1 - v_p_HD) * (1 - p_HS_trtA)
a_P_tunnels_trtA["Healthy", "Sick_1Cycle", ] <- (1 - v_p_HD) *      p_HS_trtA

## Treatment B
a_P_tunnels_trtB <- a_P_tunnels_SoC
a_P_tunnels_trtB["Healthy", "Healthy", ]     <- (1 - v_p_HD) * (1 - p_HS_trtB)
a_P_tunnels_trtB["Healthy", "Sick_1Cycle", ] <- (1 - v_p_HD) *      p_HS_trtB

# Check if transition array and probabilities are valid
# Check that transition probabilities are in [0, 1]
check_transition_probability(a_P_tunnels_SoC,  verbose = TRUE)
```

```
## [1] "Valid transition probabilities"
```

```
check_transition_probability(a_P_tunnels_trtA, verbose = TRUE)
```

```
## [1] "Valid transition probabilities"
```

```
check_transition_probability(a_P_tunnels_trtB, verbose = TRUE)
```

```
## [1] "Valid transition probabilities"
```

```
# Check that all rows sum to 1
check_sum_of_transition_array(a_P_tunnels_SoC,  n_states = n_states_tunnels, n_cycles = n_cycles, verbo
```

```
## [1] "This is a valid transition array"
```

```
check_sum_of_transition_array(a_P_tunnels_trtA, n_states = n_states_tunnels, n_cycles = n_cycles, verbo
```

```
## [1] "This is a valid transition array"
```

```
check_sum_of_transition_array(a_P_tunnels_trtB, n_states = n_states_tunnels, n_cycles = n_cycles, verbo
```

```
## [1] "This is a valid transition array"
```

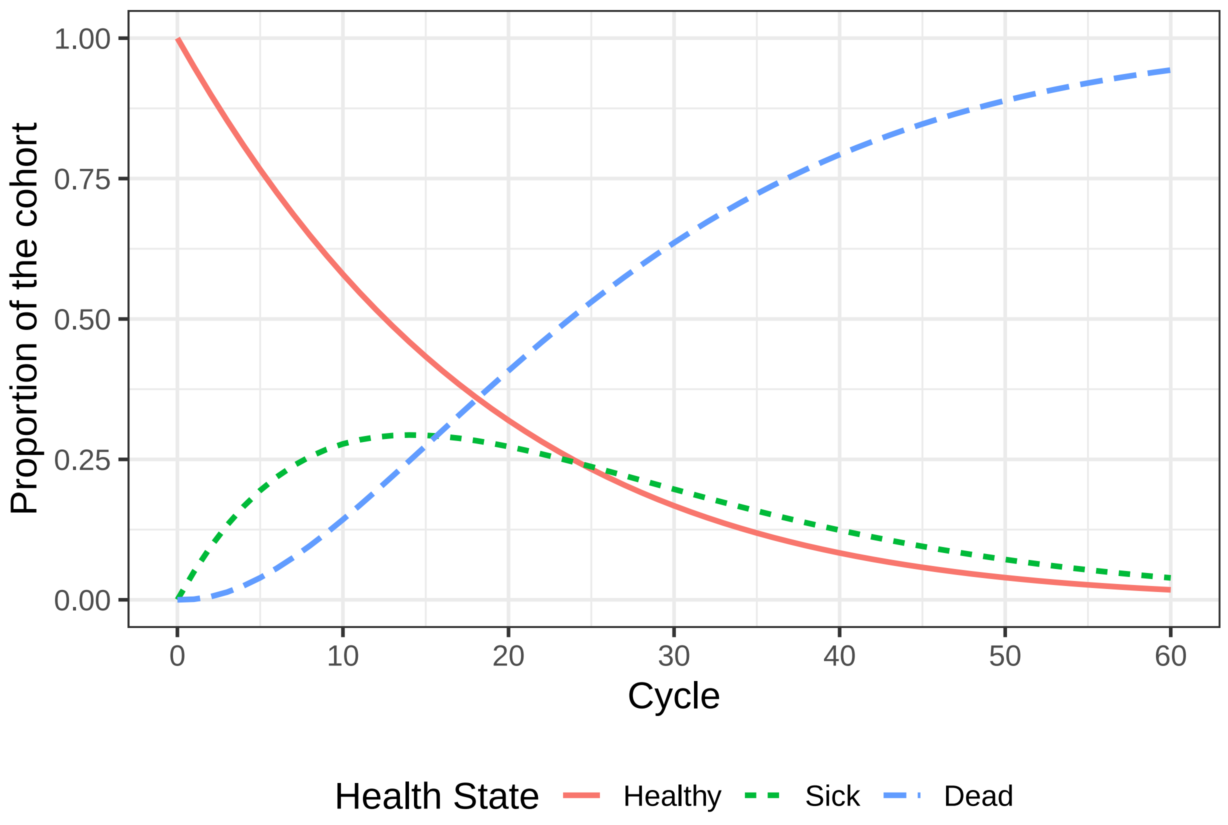
05 Run cSTM

```
## Loop over time
# Calculate cohort state based on previous state and appropriate time-slice of transition array
for (t in 1:n_cycles){
  ## Fill in cohort trace
  # For SoC
  m_M_tunnels_SoC [t + 1, ] <- m_M_tunnels_SoC [t, ] %*% a_P_tunnels_SoC [, , t]
  # For Strategy A
  m_M_tunnels_trtA[t + 1, ] <- m_M_tunnels_trtA[t, ] %*% a_P_tunnels_trtA[, , t]
  # For Strategy B
  m_M_tunnels_trtB[t + 1, ] <- m_M_tunnels_trtB[t, ] %*% a_P_tunnels_trtB[, , t]
}

# Create aggregated trace
m_M_tunnels_SoC_sum  <- cbind(Healthy = m_M_tunnels_SoC[, "Healthy"],
                              Sick    = rowSums(m_M_tunnels_SoC[, 2:(n_tunnel_size + 1)]),
                              Dead    = m_M_tunnels_SoC[, "Dead"])
m_M_tunnels_trtA_sum <- cbind(Healthy = m_M_tunnels_trtA[, "Healthy"],
                              Sick    = rowSums(m_M_tunnels_trtA[, 2:(n_tunnel_size + 1)]),
                              Dead    = m_M_tunnels_trtA[, "Dead"])
m_M_tunnels_trtB_sum <- cbind(Healthy = m_M_tunnels_trtB[, "Healthy"],
                              Sick    = rowSums(m_M_tunnels_trtB[, 2:(n_tunnel_size + 1)]),
                              Dead    = m_M_tunnels_trtB[, "Dead"])
```

06 Plot Outputs

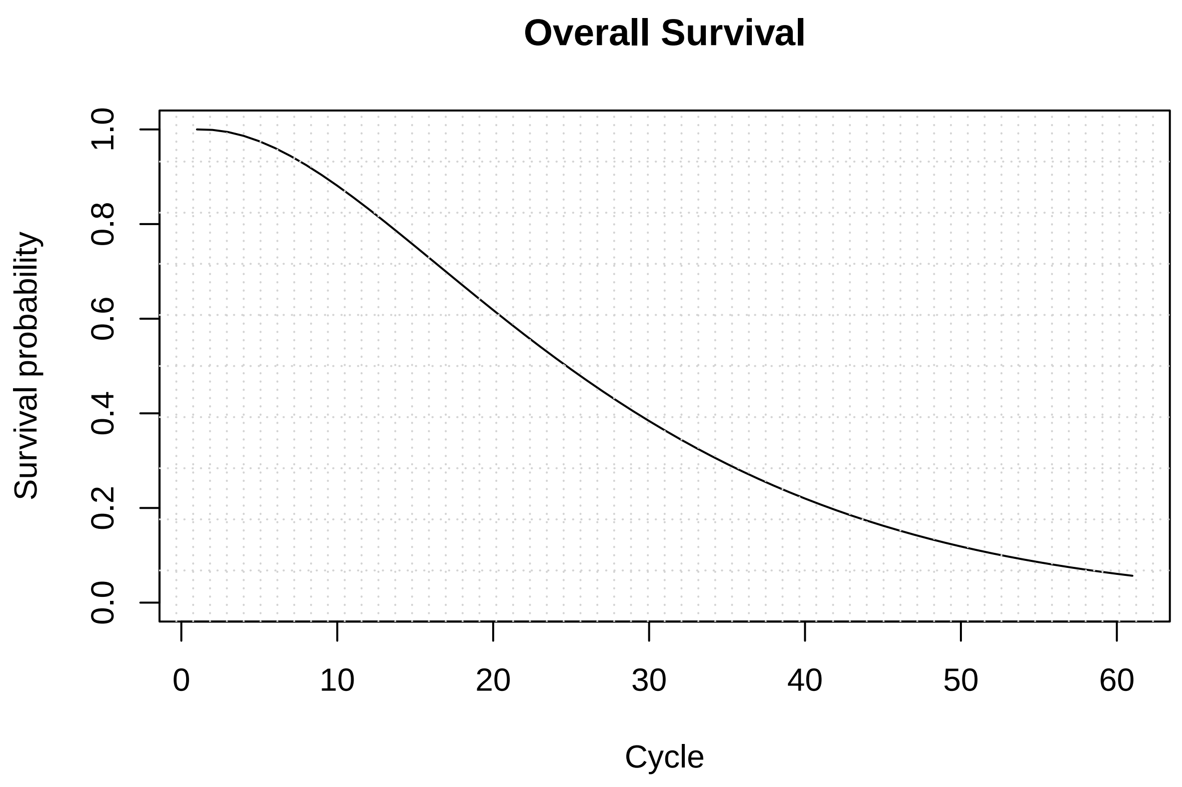
06.1 Plot the cohort trace for strategy SoC

```
## Plot the cohort trace for strategy SoC
plot_trace(m_M_tunnels_SoC_sum)
```



06.2 Overall Survival (OS)

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

# create a simple plot showing the OS
plot(v_os, type = 'l',
     ylim = c(0, 1),
     ylab = "Survival probability",
     xlab = "Cycle",
     main = "Overall Survival")
# add grid
grid(nx = n_cycles, ny = 10, col = "lightgray", lty = "dotted", lwd = par("lwd"), equilogs = TRUE)
```

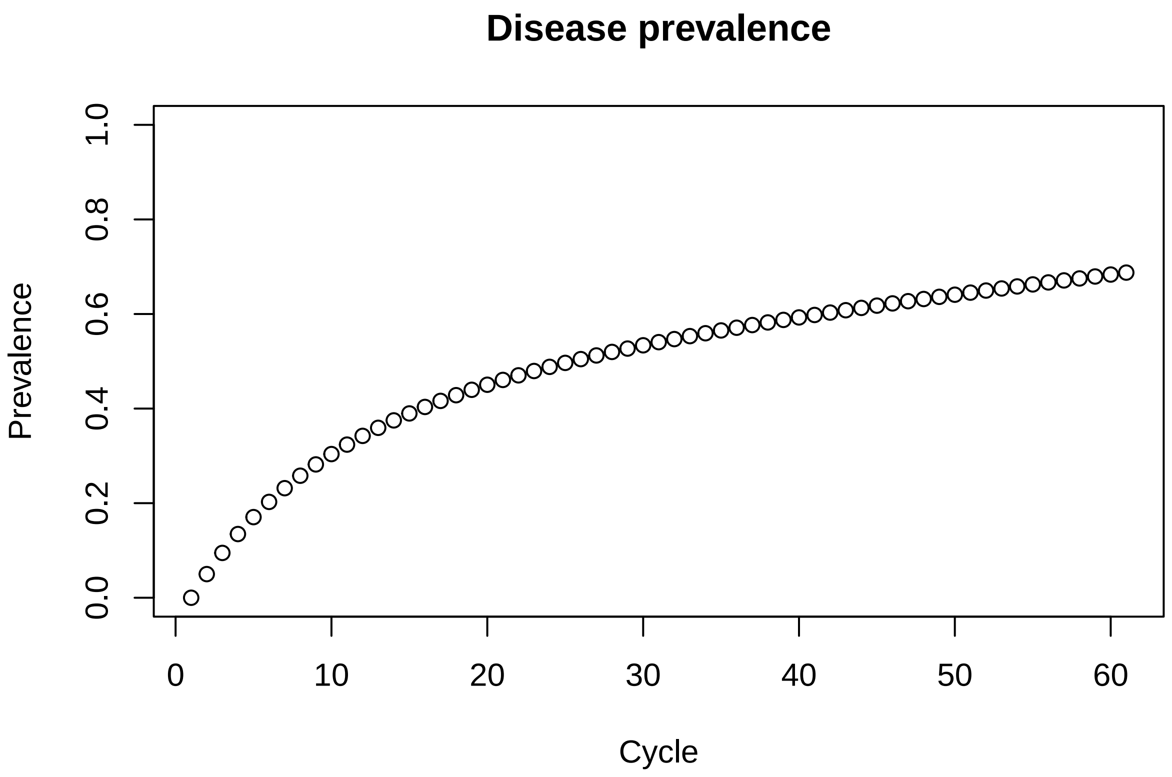


06.2.1 Life Expectancy (LE)

```r
v_le <- sum(v_os) # summing probablity of OS over time  (i.e. life expectancy)
```

06.3 Disease prevalence

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



07 Compute expected outcomes

```
# 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
v_tot_cost["Standard of Care"] <- t(m_M_tunnels_SoC %*% v_c_SoC) %*% (v_dwe * v_wcc)
v_tot_cost["Treatment A"] <- t(m_M_tunnels_trtA %*% v_c_trtA) %*% (v_dwe * v_wcc)
v_tot_cost["Treatment B"] <- t(m_M_tunnels_trtB %*% v_c_trtB) %*% (v_dwe * v_wcc)

### Expected discounted QALYs for each strategy
v_tot_qaly["Standard of Care"] <- t(m_M_tunnels_SoC %*% v_q_SoC) %*% (v_dwe * v_wcc)
v_tot_qaly["Treatment A"] <- t(m_M_tunnels_trtA %*% v_q_trtA) %*% (v_dwe * v_wcc)
v_tot_qaly["Treatment B"] <- t(m_M_tunnels_trtB %*% v_q_trtB) %*% (v_dwe * v_wcc)
```

09 Cost-effectiveness analysis (CEA)

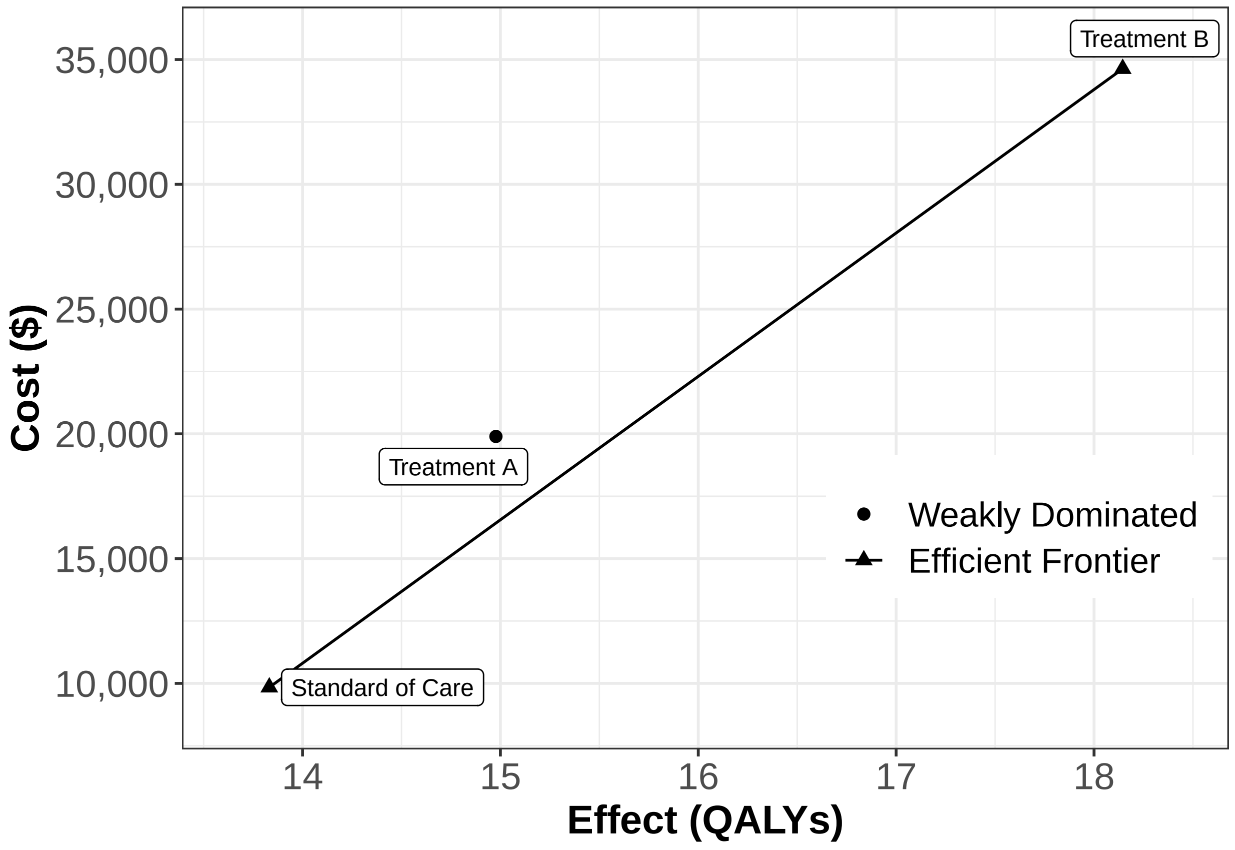
```
## Incremental cost-effectiveness ratios (ICERs)
df_cea <- calculate_icers(cost       = v_tot_cost,
                          effect     = v_tot_qaly,
                          strategies = v_names_str)
df_cea
```

```
##                                  Strategy      Cost    Effect Inc_Cost Inc_Effect
## Standard of Care Standard of Care   9841.86 13.83189       NA         NA
## Treatment B            Treatment B 34629.73 18.14492 24787.87   4.313034
## Treatment A            Treatment A 19895.47 14.97712       NA         NA
##                       ICER Status
## Standard of Care        NA     ND
## Treatment B       5747.201     ND
## Treatment A             NA     ED
```

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

```
##                                  Strategy Costs ($) QALYs Incremental Costs ($)
## Standard of Care Standard of Care     9,842 13.83                  <NA>
## Treatment B            Treatment B    34,630 18.14                24,788
## Treatment A            Treatment A    19,895 14.98                  <NA>
##                   Incremental QALYs ICER ($/QALY) Status
## Standard of Care                 NA          <NA>     ND
## Treatment B                    4.31         5,747     ND
## Treatment A                      NA          <NA>     ED
```

```
## CEA frontier
plot_icers(df_cea, label = "all", txtsize = 16) +
  expand_limits(x = max(table_cea$QALYs) + 0.1) +
  theme(legend.position = c(0.8, 0.3))
```


Treatment B
Treatment A
Standard of Care
● Weakly Dominated
▲ Efficient Frontier
Cost ($)
Effect (QALYs)