Markov Sick-Sicker model in R

With simulation-time dependency

Wang Ye

Authors:

- Wang Ye 1589936809@qq.com
- Wang Hao

In collaboration of:

1. Nanjing Drum Tower Hospital, China Pharmaceutical University, Nanjing, China
2. Department of Pharmacy, Drum Tower Hospital Affiliated to Medical School of Nanjing University, Nanjing, China

This Code adaptation Developed by the Decision Analysis in R for Technologies in Health (DARTH) workgroup.

This code implements a simulation-time-dependent Sick-Sicker cSTM model to conduct a CEA of two 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 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.

```r
rm(list = ls())      # clear memory (removes all the variables from the workspace)
```

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("dplyr", "tidyr", "reshape2", "devtools", "scales", "ellipse", "ggplot2", "ggrepel", "gridExtra"
# 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

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

03 Model input

```r
## General setup
cycle_length <- 1   # cycle length equal to one year (use 1/12 for monthly)
n_age_init   <- 25  # age at baseline
n_age_max    <- 100 # maximum age of follow up
n_cycles     <- (n_age_max - n_age_init)/cycle_length # time horizon, number of cycles
# Age labels
v_age_names  <- paste(rep(n_age_init:(n_age_max-1), each = 1/cycle_length),
                      1:(1/cycle_length),
                      sep = ".")
# the 4 health states of the model:
v_names_states <- c("H",  # Healthy (H)
                    "S1", # Sick (S1)
                    "S2", # Sicker (S2)
                    "D")  # Dead (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

## Within-cycle correction (WCC) using half-cycle rule
v_wcc  <- gen_wcc(n_cycles = n_cycles, method = "half-cycle")

### Transition rates (annual), and hazard ratios (HRs)
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

## Age-dependent mortality rates
lt_usa_2015 <- read.csv("HMD_USA_Mx_2015.csv")
# Extract age-specific all-cause mortality for ages in model time horizon
v_r_mort_by_age <- lt_usa_2015 %>%
  dplyr::filter(Age >= n_age_init & Age < n_age_max) %>%
  dplyr::select(Total) %>%
  as.matrix()

### 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

### Transition rewards
du_HS1  <- 0  # disutility when transitioning from Healthy to Sick
ic_HS1  <- 0  # increase in cost when transitioning from Healthy to Sick
ic_D    <- 0  # increase in cost when dying

### 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))

# Process model inputs
## Age-specific transition rates to the Dead state for all cycles
v_r_HDage  <- rep(v_r_mort_by_age, each = 1/cycle_length)
# Name age-specific mortality vector
names(v_r_HDage) <- v_age_names

# compute mortality rates
v_r_S1Dage  <- v_r_HDage * hr_S1 # Age-specific mortality rate in the Sick state
v_r_S2Dage  <- v_r_HDage * hr_S2 # Age-specific mortality rate in the Sicker state
v_r_S2Dage
```

```
##    25.1    26.1    27.1    28.1    29.1    30.1    31.1    32.1    33.1    34.1
## 0.01014 0.00999 0.01070 0.01087 0.01162 0.01167 0.01213 0.01289 0.01331 0.01375
##    35.1    36.1    37.1    38.1    39.1    40.1    41.1    42.1    43.1    44.1
## 0.01420 0.01490 0.01550 0.01616 0.01657 0.01747 0.01902 0.02052 0.02173 0.02395
##    45.1    46.1    47.1    48.1    49.1    50.1    51.1    52.1    53.1    54.1
## 0.02559 0.02807 0.03023 0.03349 0.03712 0.04085 0.04490 0.04905 0.05364 0.05806
##    55.1    56.1    57.1    58.1    59.1    60.1    61.1    62.1    63.1    64.1
## 0.06253 0.06775 0.07395 0.07895 0.08418 0.08974 0.09666 0.10456 0.11384 0.11838
##    65.1    66.1    67.1    68.1    69.1    70.1    71.1    72.1    73.1    74.1
## 0.12667 0.13593 0.14700 0.15732 0.17340 0.18758 0.20967 0.22917 0.24913 0.26767
##    75.1    76.1    77.1    78.1    79.1    80.1    81.1    82.1    83.1    84.1
## 0.29707 0.32412 0.35982 0.39238 0.43595 0.48727 0.53735 0.59911 0.66618 0.74051
##    85.1    86.1    87.1    88.1    89.1    90.1    91.1    92.1    93.1    94.1
## 0.82190 0.90754 1.03968 1.15093 1.24341 1.37872 1.54177 1.72393 1.94100 2.12654
##    95.1    96.1    97.1    98.1    99.1
## 2.43752 2.59087 2.87781 3.16429 3.39149
```

```
# transform rates to probabilities adjusting by cycle length
p_HS1       <- rate_to_prob(r = r_HS1,  t = cycle_length) # constant annual probability of becoming Sic
p_S1H       <- rate_to_prob(r = r_S1H,  t = cycle_length) # constant annual probability of becoming Hea
p_S1S2      <- rate_to_prob(r = r_S1S2, t = cycle_length) # constant annual probability of becoming Sic
v_p_HDage   <- rate_to_prob(v_r_HDage,  t = cycle_length) # Age-specific mortality risk in the Healthy 
v_p_S1Dage  <- rate_to_prob(v_r_S1Dage, t = cycle_length) # Age-specific mortality risk in the Sick sta
v_p_S2Dage  <- rate_to_prob(v_r_S2Dage, t = cycle_length) # Age-specific mortality risk in the Sicker s
v_p_S2Dage
```

```
##        25.1        26.1        27.1        28.1        29.1        30.1
## 0.010088764 0.009940266 0.010642959 0.010811135 0.011552749 0.011602170
##        31.1        32.1        33.1        34.1        35.1        36.1
## 0.012056728 0.012807280 0.013221814 0.013655901 0.014099656 0.014789544
##        37.1        38.1        39.1        40.1        41.1        42.1
## 0.015380493 0.016030128 0.016433473 0.017318284 0.018840261 0.020310898
##        43.1        44.1        45.1        46.1        47.1        48.1
## 0.021495604 0.023665475 0.025265351 0.027679698 0.029777643 0.032935418
##        49.1        50.1        51.1        52.1        53.1        54.1
## 0.036439499 0.040026885 0.043906914 0.047866478 0.052226757 0.056406670
##        55.1        56.1        57.1        58.1        59.1        60.1
## 0.060615119 0.065505932 0.071281871 0.075913872 0.080734227 0.085831162
##        61.1        62.1        63.1        64.1        65.1        66.1
## 0.092135372 0.099279247 0.107599271 0.111641588 0.118975637 0.127096266
##        67.1        68.1        69.1        70.1        71.1        72.1
## 0.136706023 0.145569403 0.159198773 0.171037201 0.189148217 0.204806661
##        73.1        74.1        75.1        76.1        77.1        78.1
## 0.220521365 0.234839758 0.257007999 0.276836542 0.302198081 0.324552605
##        79.1        80.1        81.1        82.1        83.1        84.1
## 0.353349942 0.385698851 0.415705417 0.450699704 0.486332957 0.523129351
##        85.1        86.1        87.1        88.1        89.1        90.1
## 0.560404371 0.596484347 0.646432194 0.683657566 0.711600902 0.748099220
##        91.1        92.1        93.1        94.1        95.1        96.1
## 0.785998018 0.821636201 0.856439682 0.880750817 0.912622721 0.925045199
##        97.1        98.1        99.1
## 0.943742167 0.957755876 0.966341512
```

```
## 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
# adjusting by cycle length conditional on surviving
p_S1S2_trtAB <- rate_to_prob(r = r_S1S2_trtAB, t = cycle_length)
```

04 Construct state-transition models

04.1 Initial state vector

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

```
##  H S1 S2  D
##  1  0  0  0
```

04.2 Initialize cohort traces

```r
### Initialize cohort trace under 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 strategy 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 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_SoC <- array(0,
                 dim  = c(n_states, n_states, n_cycles),
                 dimnames = list(v_names_states,
                                 v_names_states,
                                 0:(n_cycles - 1)))
### Fill in array
## From H
a_P_SoC["H", "H", ]   <- (1 - v_p_HDage) * (1 - p_HS1)
a_P_SoC["H", "S1", ]  <- (1 - v_p_HDage) *      p_HS1
a_P_SoC["H", "D", ]   <-      v_p_HDage
## From S1
a_P_SoC["S1", "H", ]  <- (1 - v_p_S1Dage) *       p_S1H
a_P_SoC["S1", "S1", ] <- (1 - v_p_S1Dage) * (1 - (p_S1H + p_S1S2))
a_P_SoC["S1", "S2", ] <- (1 - v_p_S1Dage) *               p_S1S2
a_P_SoC["S1", "D", ]  <-      v_p_S1Dage
## From S2
a_P_SoC["S2", "S2", ] <- 1 - v_p_S2Dage
a_P_SoC["S2", "D", ]  <-     v_p_S2Dage
## From D
a_P_SoC["D", "D", ]   <- 1

### Initialize transition probability array for strategy AB
a_P_strAB <- a_P_SoC
# Update only transition probabilities from S1 involving p_S1S2
```

```r
a_P_strAB["S1", "S1", ] <- (1 - v_p_S1Dage) * (1 - (p_S1H + p_S1S2_trtAB))
a_P_strAB["S1", "S2", ] <- (1 - v_p_S1Dage) *               p_S1S2_trtAB

## Check if transition probability arrays are valid
### Check that transition probabilities are [0, 1]
check_transition_probability(a_P_SoC,   verbose = TRUE)
check_transition_probability(a_P_strAB, verbose = TRUE)
### Check that all rows for each slice of the array sum to 1
check_sum_of_transition_array(a_P_SoC,   n_states = n_states, n_cycles = n_cycles, verbose = TRUE)
check_sum_of_transition_array(a_P_strAB, n_states = n_states, n_cycles = n_cycles, verbose = TRUE)
```

04.4 Create transition dynamics arrays

```r
# These arrays will capture transitions from each state to another over time
### Initialize transition dynamics array for strategy SoC
a_A_SoC <- array(0,
             dim      = c(n_states, n_states, n_cycles + 1),
             dimnames = list(v_names_states, v_names_states, 0:n_cycles))
# Set first slice of a_A_SoC with the initial state vector in its diagonal
diag(a_A_SoC[, , 1]) <- v_m_init
### Initialize transition-dynamics array for strategy AB
# Structure and initial states are the same as for SoC
a_A_strAB <- a_A_SoC
```

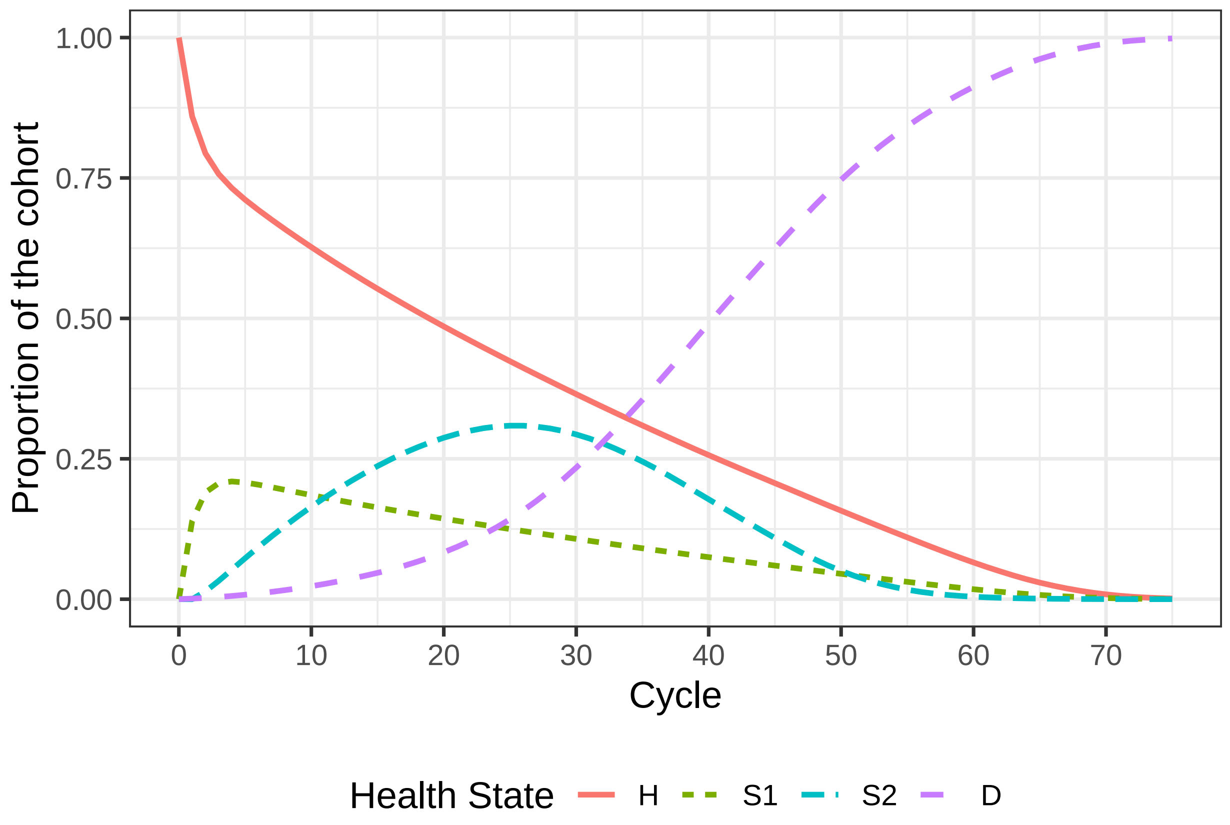
05 Run Markov model

```r
# Iterative solution of age-dependent cSTM
for(t in 1:n_cycles){
  ## Fill in cohort trace
  # For SoC
  m_M_SoC[t + 1, ]   <- m_M_SoC[t, ]   %*% a_P_SoC[, , t]
  # For strategy AB
  m_M_strAB[t + 1, ] <- m_M_strAB[t, ] %*% a_P_strAB[, , t]

  ## Fill in transition-dynamics array
  # For SoC
  a_A_SoC[, , t + 1]   <- diag(m_M_SoC[t, ])   %*% a_P_SoC[, , t]
  # For strategy AB
  a_A_strAB[, , t + 1] <- diag(m_M_strAB[t, ]) %*% a_P_strAB[, , t]
}

## Store the cohort traces in a list
l_m_M <- list(SoC =  m_M_SoC,
              AB  =  m_M_strAB)
names(l_m_M) <- v_names_str

## Store the transition dynamics array for each strategy in a list
l_a_A <- list(SoC =  a_A_SoC,
              AB  =  a_A_strAB)
names(l_a_A) <- v_names_str
```

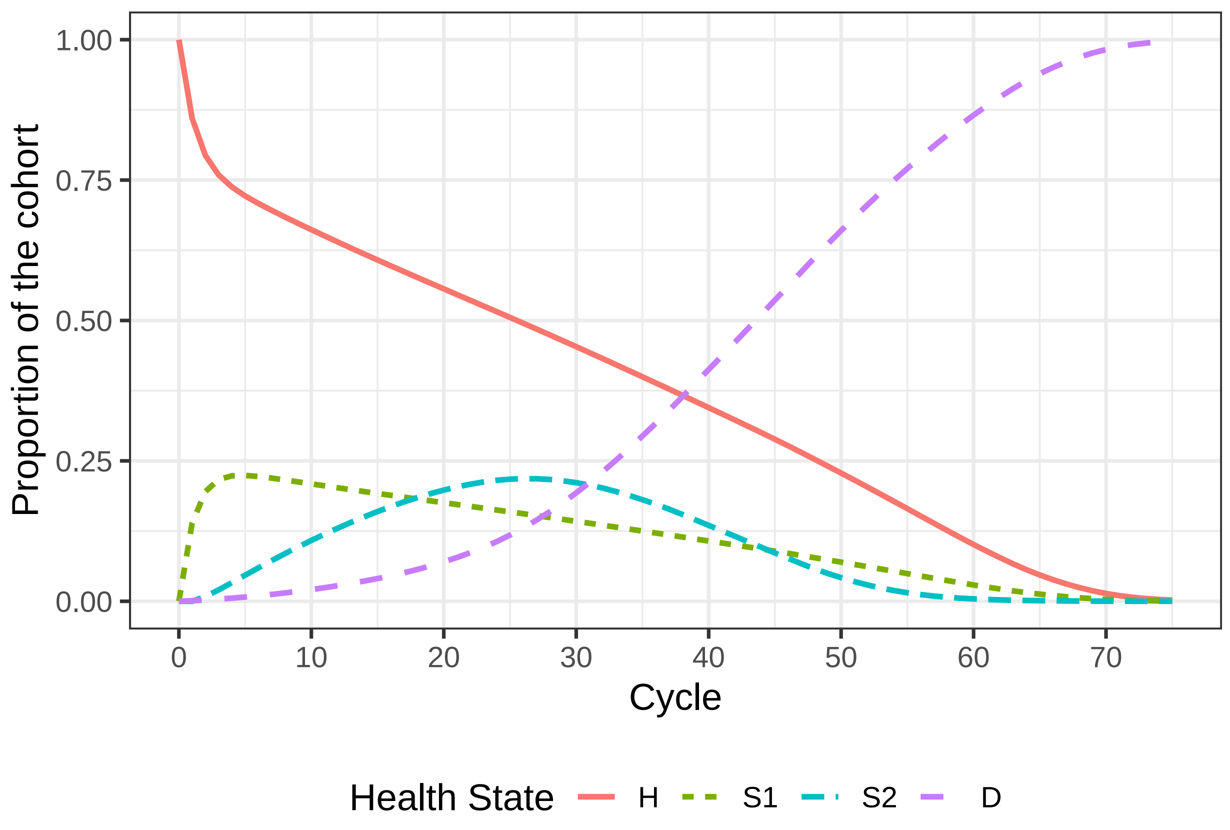
06 Plot Outputs

06.1 Plot the cohort trace for strategies SoC and AB

```
plot_trace(m_M_SoC)
```



```
plot_trace(m_M_strAB)
```



07 State Rewards

```
## Scale by the cycle length
# Vector of state utilities under strategy SoC
v_u_SoC    <- c(H  = u_H,
                S1 = u_S1,
                S2 = u_S2,
                D  = u_D) * cycle_length
# Vector of state costs under strategy SoC
v_c_SoC    <- c(H  = c_H,
                S1 = c_S1,
                S2 = c_S2,
                D  = c_D) * cycle_length
# Vector of state utilities under strategy AB
v_u_strAB  <- c(H  = u_H,
                S1 = u_trtAB,
                S2 = u_S2,
                D  = u_D) * cycle_length
# Vector of state costs under strategy AB
v_c_strAB  <- c(H  = c_H,
                S1 = c_S1 + c_trtAB,
                S2 = c_S2 + c_trtAB,
                D  = c_D) * cycle_length

## Store state rewards
```

```r
# Store the vectors of state utilities for each strategy in a list
l_u <- list(SoC = v_u_SoC,
            AB  = v_u_strAB)
# Store the vectors of state cost for each strategy in a list
l_c <- list(SoC =  v_c_SoC,
            AB  =  v_c_strAB)

# assign strategy names to matching items in the lists
names(l_u) <- names(l_c) <- v_names_str
```

08 Compute expected outcomes

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

## Loop through each strategy and calculate total utilities and costs
for (i in 1:n_str) { # i <- 1
  v_u_str <- l_u[[i]]   # select the vector of state utilities for the i-th strategy
  v_c_str <- l_c[[i]]   # select the vector of state costs for the i-th strategy
  a_A_str <- l_a_A[[i]] # select the transition array for the i-th strategy

  ## Array of state rewards
  # Create transition matrices of state utilities and state costs for the i-th strategy
  m_u_str   <- matrix(v_u_str, nrow = n_states, ncol = n_states, byrow = T)
  m_c_str   <- matrix(v_c_str, nrow = n_states, ncol = n_states, byrow = T)
  # Expand the transition matrix of state utilities across cycles to form a transition array of state u
  a_R_u_str <- array(m_u_str,
                     dim      = c(n_states, n_states, n_cycles + 1),
                     dimnames = list(v_names_states, v_names_states, 0:n_cycles))
  # Expand the transition matrix of state costs across cycles to form a transition array of state costs
  a_R_c_str <- array(m_c_str,
                     dim      = c(n_states, n_states, n_cycles + 1),
                     dimnames = list(v_names_states, v_names_states, 0:n_cycles))

  ## Apply transition rewards
  # Apply disutility due to transition from H to S1
  a_R_u_str["H", "S1", ]      <- a_R_u_str["H", "S1", ]       - du_HS1
  # Add transition cost per cycle due to transition from H to S1
  a_R_c_str["H", "S1", ]      <- a_R_c_str["H", "S1", ]       + ic_HS1
  # Add transition cost  per cycle of dying from all non-dead states
  a_R_c_str[-n_states, "D", ] <- a_R_c_str[-n_states, "D", ] + ic_D

  ### Expected QALYs and costs for all transitions per cycle
  # QALYs = life years x QoL
  # Note: all parameters are annual in our example. In case your own case example is different make sur
  a_Y_c_str <- a_A_str * a_R_c_str
  a_Y_u_str <- a_A_str * a_R_u_str

  ### Expected QALYs and costs per cycle
  ## Vector of QALYs and costs
  v_qaly_str <- apply(a_Y_u_str, 3, sum) # sum the proportion of the cohort across transitions
  v_cost_str <- apply(a_Y_c_str, 3, sum) # sum the proportion of the cohort across transitions

  ## Discounted total expected QALYs and Costs per strategy and apply within-cycle correction if applic
  # QALYs
  v_tot_qaly[i] <- t(v_qaly_str) %*% (v_dwe * v_wcc)
  # Costs
  v_tot_cost[i] <- t(v_cost_str) %*% (v_dwc * v_wcc)
}
```

09 Cost-effectiveness analysis (CEA)

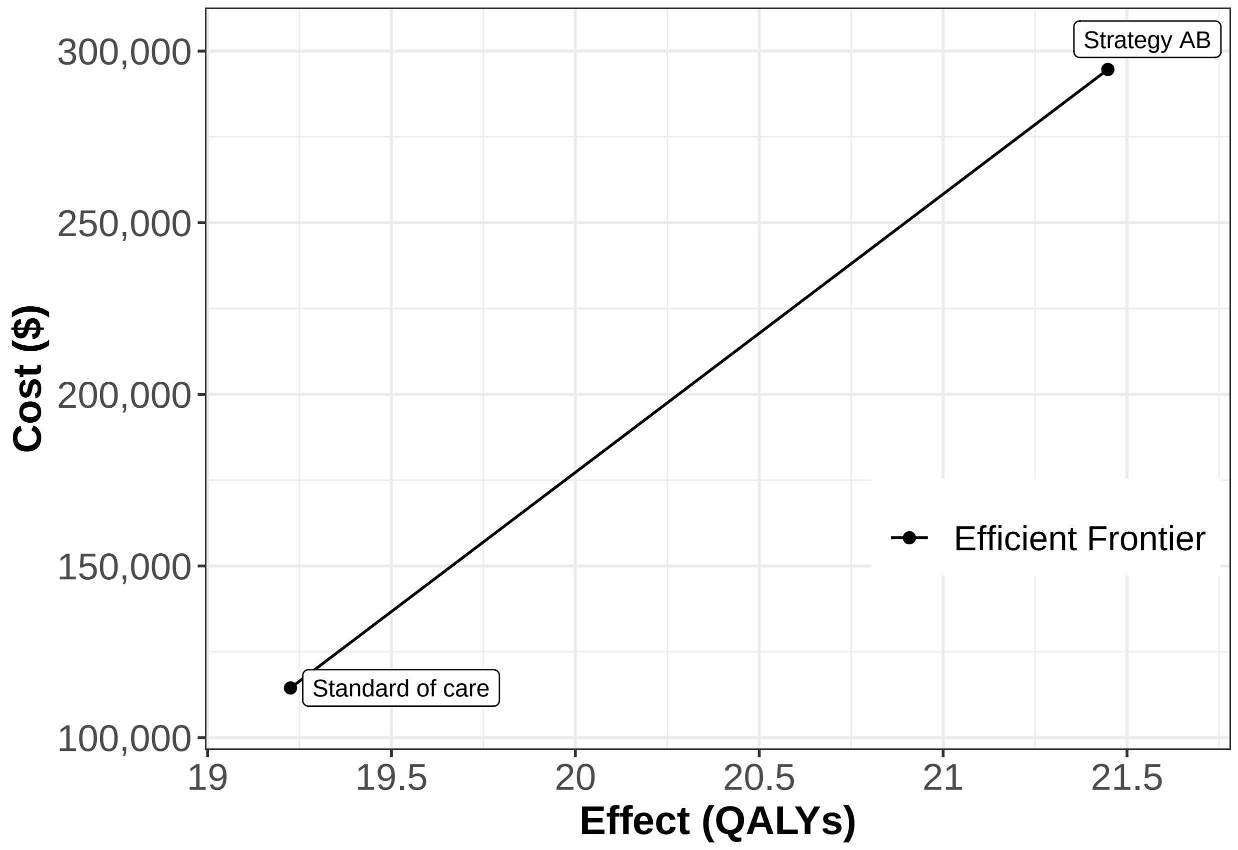
```
## Incremental cost-effectiveness ratios (ICERs)
df_cea <- calculate_icers(cost       = v_tot_cost,
                          effect     = v_tot_qaly,
                          strategies = v_names_str)
df_cea
```

```
##                          Strategy     Cost   Effect Inc_Cost Inc_Effect
## Standard of care Standard of care 114473.9 19.22543       NA         NA
## Strategy AB           Strategy AB 294643.4 21.44797 180169.6   2.222541
##                      ICER Status
## Standard of care       NA     ND
## Strategy AB      81064.67     ND
```

```
## 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   114,474 19.23                  <NA>
## Strategy AB           Strategy AB   294,643 21.45               180,170
##                  Incremental QALYs ICER ($/QALY) Status
## Standard of care                NA          <NA>     ND
## Strategy AB                   2.22        81,065     ND
```

```
## CEA frontier
plot(df_cea, label = "all", txtsize = 16) +
  expand_limits(x = max(table_cea$QALYs) + 0.1) +
  theme(legend.position = c(0.82, 0.3))
```



10 Deterministic Sensitivity Analysis (DSA)

```
## Load model, CEA and PSA functions
source('Functions_markov_sick-sicker_time.R')
```

10.1 Model input for SA

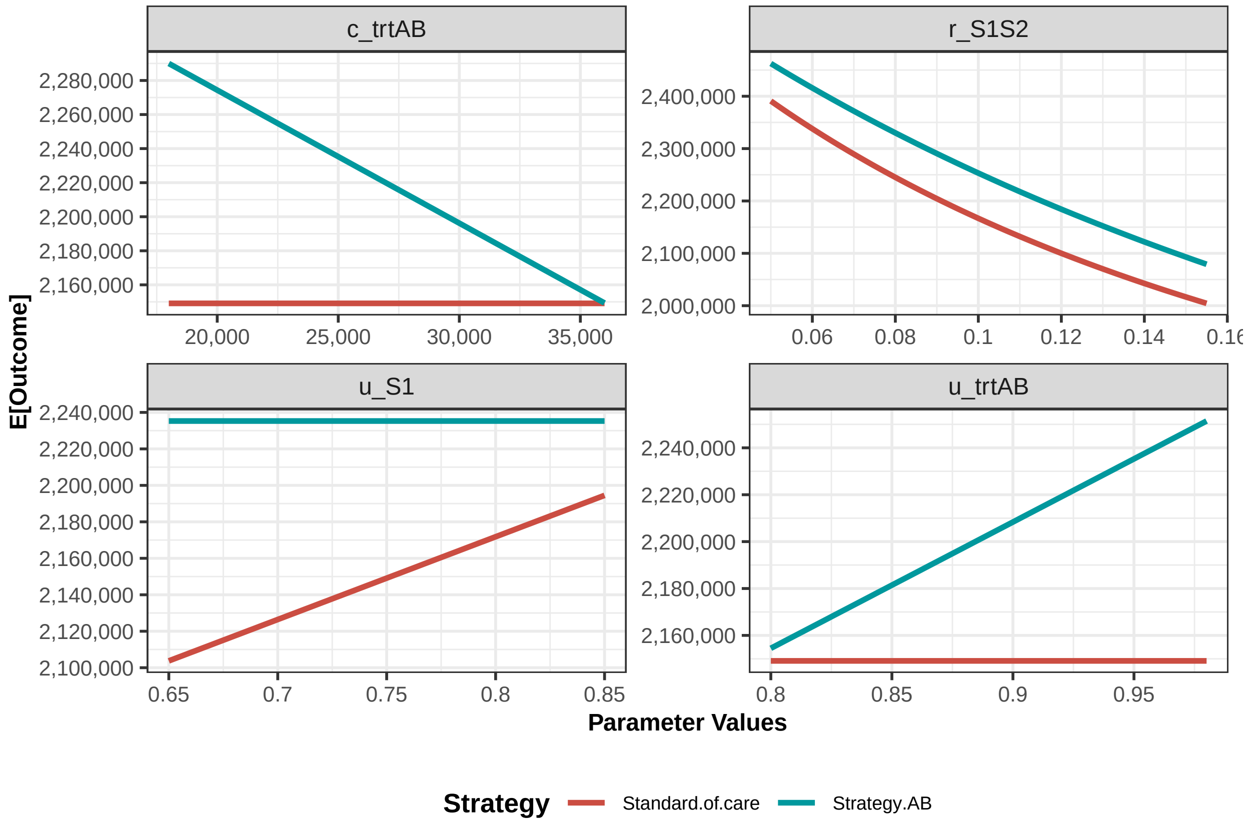
```
## List of input parameters
l_params_all <- list(
  # Transition probabilities (per cycle), hazard ratios
  v_r_HDage = v_r_HDage, # constant rate of dying when Healthy (all-cause mortality)
  r_HS1     = 0.15,      # constant annual rate of becoming Sick when Healthy conditional on surviving
  r_S1H     = 0.5,       # constant annual rate of becoming Healthy when Sick conditional on surviving
  r_S1S2    = 0.105,     # constant annual rate of becoming Sicker when Sick conditional on surviving
  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,      # cost of remaining one cycle in Healthy
  c_S1      = 4000,      # cost of remaining one cycle in Sick
  c_S2      = 15000,     # cost of remaining one cycle in Sicker
  c_D       = 0,         # cost of being dead (per cycle)
  c_trtAB   = 25000,      # cost of treatment A
  # Utilities
  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_trtAB   = 0.95,       # utility when being treated with A
  ## Transition rewards
  du_HS1    = 0.01,       # disutility when transitioning from Healthy to Sick
  ic_HS1    = 1000,       # increase in cost when transitioning from Healthy to Sick
  ic_D      = 2000,       # increase in cost when dying
  # Initial and maximum ages
  n_age_init = 25,
  n_age_max  = 100,
  # Discount rates
  d_c        = 0.03,      # annual discount rate for costs
  d_e        = 0.03,      # annual discount rate for QALYs,
  # Cycle length
  cycle_length = 1
)
```

10.2 One-way sensitivity analysis (OWSA)

```
options(scipen = 999) # disabling scientific notation in R
# data.frame 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("r_S1S2", "c_trtAB", "u_S1", "u_trtAB"),
                             min  = c(0.05 , 18000 , 0.65, 0.80), # min parameter values
                             max  = c(0.155, 36000 , 0.85, 0.98)  # max parameter values
                             )

owsa_nmb  <- run_owsa_det(params_range    = df_params_owsa,    # data.frame 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           = 120000)            # extra argument to pass to FUN
```

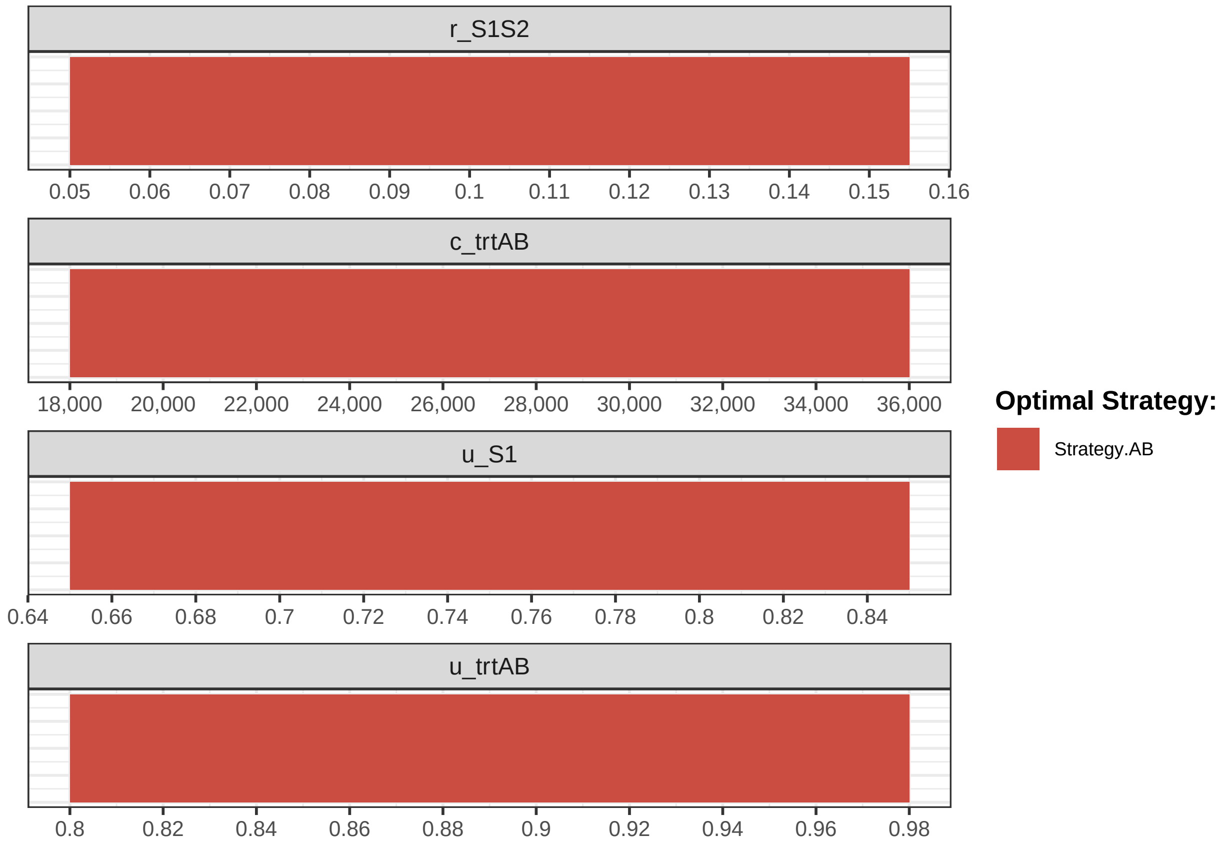
```
##   |                                                                              |
```

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



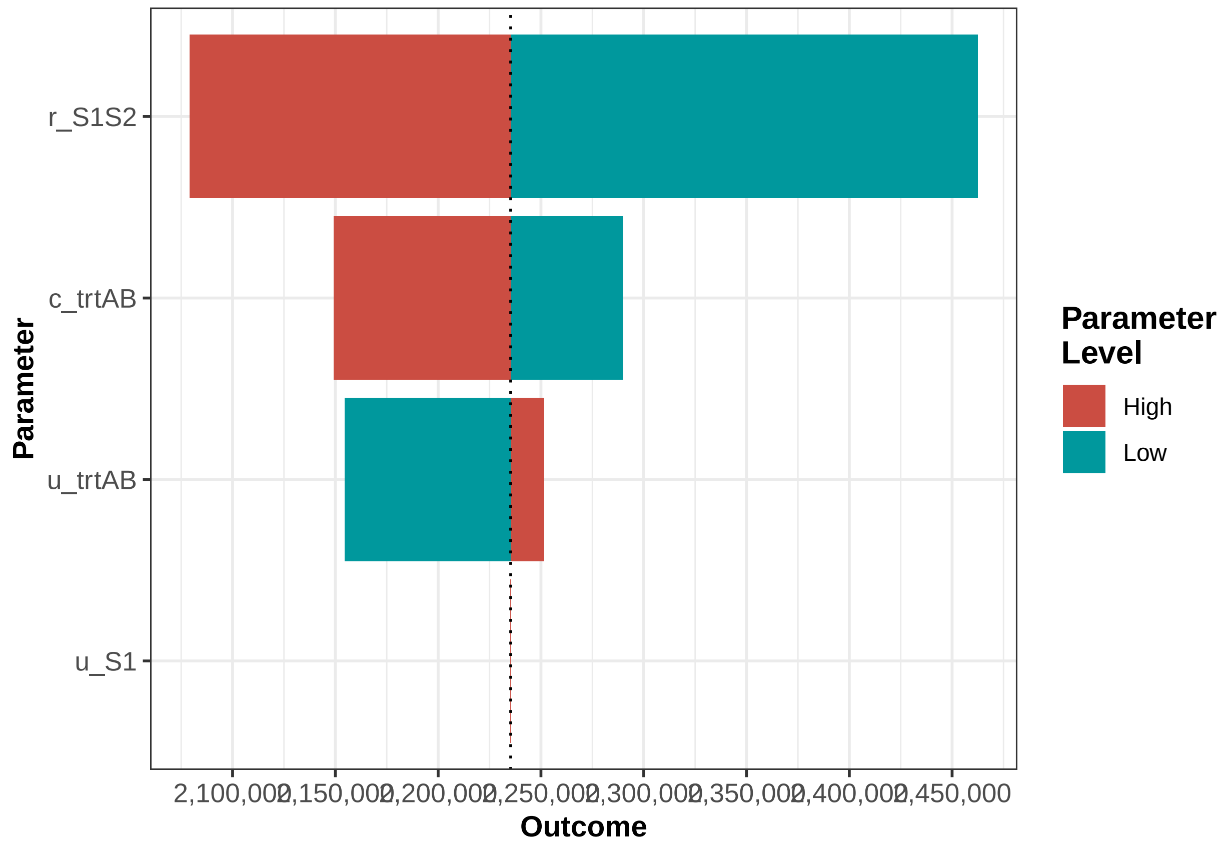
10.2.1 Optimal strategy with OWSA

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



10.2.2 Tornado plot

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

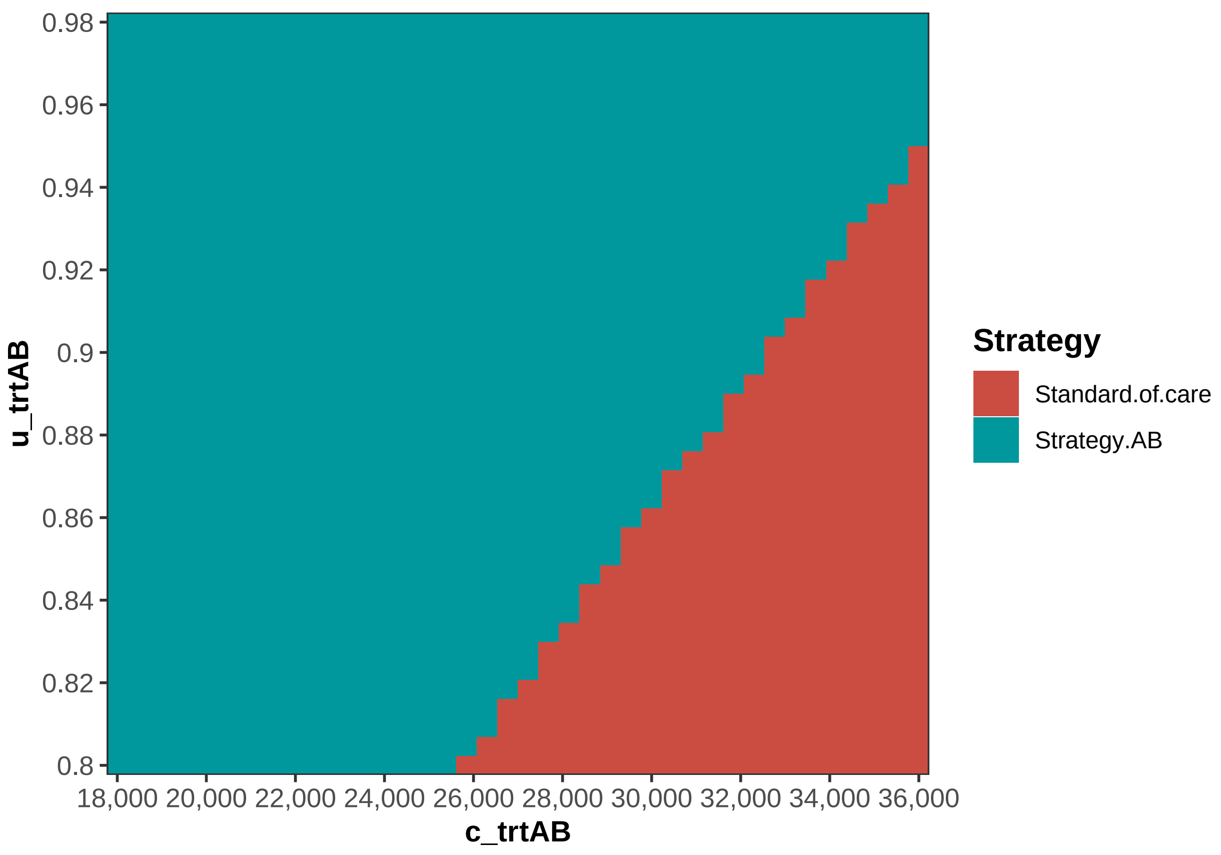


10.3 Two-way sensitivity analysis (TWSA)

```
# 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_trtAB", "u_trtAB"),
                             min  = c(18000, 0.80),  # min parameter values
                             max  = c(36000, 0.98)   # max parameter values
                             )

twsa_nmb <- run_twsa_det(params_range    = df_params_twsa,    # data.frame 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        = 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
```

```
##   |                                                                              |
```

```
plot(twsa_nmb)
```



11 Probabilistic Sensitivity Analysis (PSA)

11.1 Model input

```
# Store the parameter names into a vector
v_names_params <- names(l_params_all)

## Test functions to generate CE outcomes and PSA dataset
# Test function to compute CE outcomes
calculate_ce_out(l_params_all)
```

```
##                          Strategy     Cost   Effect     NMB
## Standard of care Standard of care 116374.4 18.87920 1771545
## Strategy AB           Strategy AB 296300.3 21.09653 1813352
```

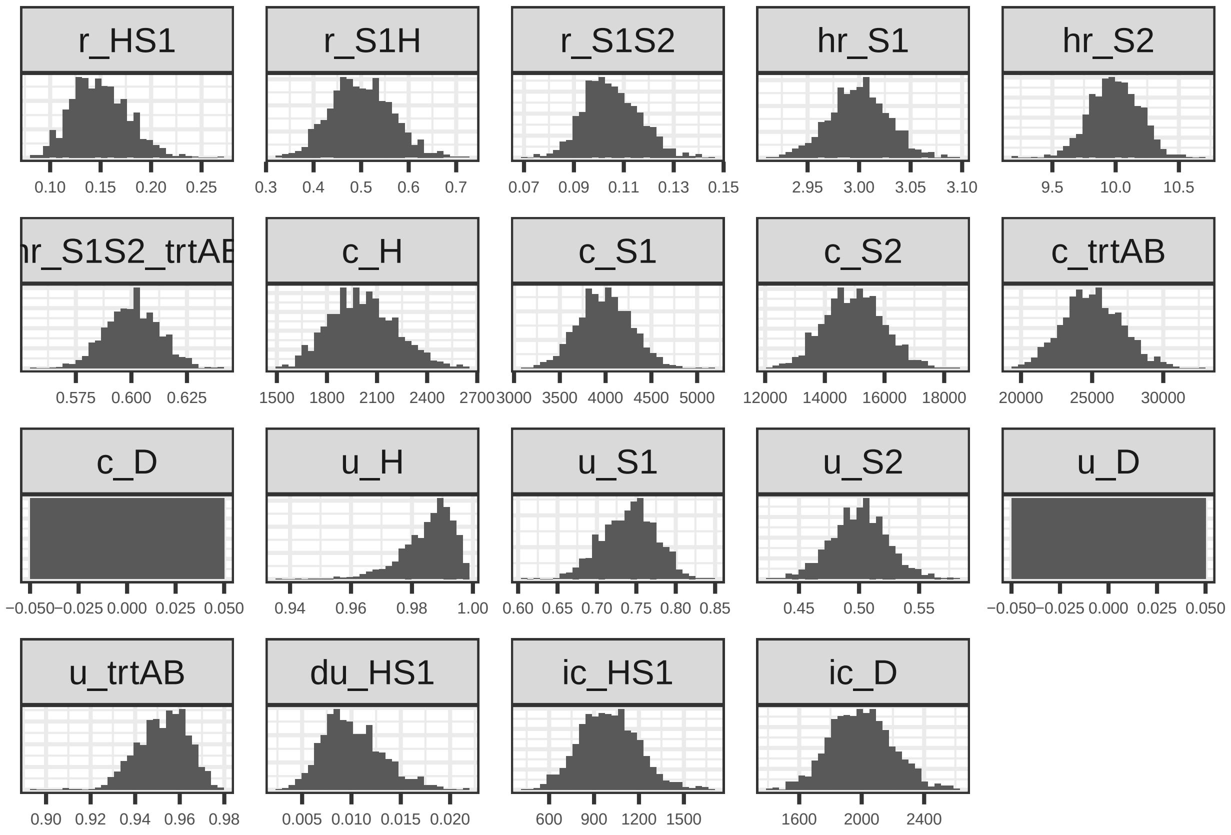
```
# Test function to generate PSA input dataset
generate_psa_params(10)
```

```
##        r_HS1     r_S1H      r_S1S2    hr_S1     hr_S2 hr_S1S2_trtAB      c_H
## 1  0.1879013 0.4801301 0.10796555 3.000891 10.137621     0.5787234 1969.304
## 2  0.1487534 0.4461028 0.11015130 2.977455  9.720297     0.5895503 1641.192
## 3  0.1240989 0.4277723 0.12019699 3.019838 10.140862     0.5994554 2276.032
## 4  0.1207401 0.6823019 0.10356461 2.986256 10.506942     0.5933100 1717.867
## 5  0.1407570 0.4834377 0.10607418 2.964711 10.186604     0.6094500 2132.549
## 6  0.1444703 0.5199434 0.09743299 2.979785 10.290916     0.5985713 2113.222
## 7  0.1298975 0.4864323 0.11804713 2.984690 10.307978     0.5986025 2123.062
## 8  0.1953886 0.5546269 0.09521281 3.041686  9.847810     0.5952682 2387.026
## 9  0.1092301 0.5654439 0.11498572 2.983202 10.073315     0.5858269 1600.231
## 10 0.1824868 0.4388704 0.09804320 3.044841 10.093864     0.6101527 1916.467
##        c_S1     c_S2   c_trtAB c_D       u_H      u_S1      u_S2 u_D    u_trtAB
## 1  3970.522 15303.55 27747.45   0 0.9832065 0.7169406 0.4628316   0 0.9456881
## 2  3680.588 15378.20 25209.56   0 0.9917677 0.7851270 0.4504530   0 0.9093339
## 3  3781.956 15614.73 25142.62   0 0.9727485 0.7582961 0.4727426   0 0.9390084
## 4  3448.146 15401.50 24697.69   0 0.9836544 0.7648816 0.4851046   0 0.9537789
## 5  4656.666 15198.97 22459.38   0 0.9915345 0.7751203 0.5311969   0 0.9577708
## 6  3879.536 14070.22 27794.29   0 0.9964569 0.7757487 0.5155318   0 0.9582262
## 7  3911.558 16006.75 25340.20   0 0.9795568 0.7965897 0.5041742   0 0.9349559
## 8  4497.693 15402.85 26952.90   0 0.9896162 0.7381845 0.4668479   0 0.9412370
## 9  4198.111 16322.33 23576.54   0 0.9743259 0.7235973 0.4723743   0 0.9570896
## 10 4087.810 15848.52 24120.75   0 0.9827154 0.7573872 0.5319503   0 0.9618487
##         du_HS1    ic_HS1     ic_D
## 1  0.008412808  792.4331 1939.526
## 2  0.010606291  926.6452 2048.008
## 3  0.010531334  838.5385 2807.393
## 4  0.009906859 1048.1656 1764.510
## 5  0.011257731  875.4987 2162.258
## 6  0.016727498 1092.0825 1808.114
## 7  0.010509535 1037.3152 2368.812
## 8  0.009053723  809.3469 2073.669
## 9  0.012068970 1130.1533 1974.209
## 10 0.011478733 1108.5800 2059.040
```

```
## Generate PSA dataset
# 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)
```

```
##       r_HS1     r_S1H      r_S1S2    hr_S1     hr_S2 hr_S1S2_trtAB      c_H
## 1 0.1879013 0.5386300 0.11118482 2.984211  9.955194     0.5967508 1976.371
## 2 0.1487534 0.5341772 0.08823294 3.032752  9.655530     0.6382210 1919.375
## 3 0.1240989 0.5081914 0.10561891 3.005960 10.210922     0.6055352 2119.283
## 4 0.1207401 0.5807883 0.10238229 3.021883  9.896945     0.6157481 2591.358
## 5 0.1407570 0.5669941 0.09952737 2.939640  9.863104     0.6024367 2250.708
## 6 0.1444703 0.5624339 0.12142602 2.982949 10.108564     0.5975063 2504.241
##       c_S1     c_S2   c_trtAB c_D       u_H      u_S1      u_S2 u_D    u_trtAB
## 1 3932.928 15714.69 25348.12   0 0.9950597 0.6992784 0.5018781   0 0.9728172
## 2 3843.274 14724.57 24976.52   0 0.9795858 0.7503645 0.5000770   0 0.9531826
## 3 4314.533 18349.98 28545.94   0 0.9765326 0.6703114 0.4778468   0 0.9593837
## 4 3903.295 15191.92 23475.91   0 0.9849616 0.7582911 0.4750196   0 0.9379568
## 5 4132.504 15029.64 20168.13   0 0.9864103 0.7889100 0.4999208   0 0.9376958
## 6 3961.236 16738.03 25931.16   0 0.9734870 0.7256934 0.4970604   0 0.9607761
##        du_HS1    ic_HS1     ic_D
## 1 0.012672847  922.6349 1851.850
## 2 0.007971746 1075.9420 1970.602
## 3 0.006323387 1138.3535 2035.958
## 4 0.010696387 1108.6087 2071.939
## 5 0.010152628 1128.2519 1937.251
## 6 0.008681084  905.7657 1982.216
```

```
### Histogram of parameters
ggplot(melt(df_psa_input, variable.name = "Parameter"), aes(x = value)) +
  facet_wrap(~Parameter, scales = "free") +
  geom_histogram(aes(y = ..density..)) +
  ylab("") +
  theme_bw(base_size = 16) +
  theme(axis.text = element_text(size = 6),
        axis.title.x = element_blank(),
        axis.title.y = element_blank(),
        axis.text.y  = element_blank(),
        axis.ticks.y = element_blank())
```



11.2 Run PSA

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

```r
colnames(df_e) <- v_names_str

# Conduct probabilistic sensitivity analysis
# Run Markov model on each parameter set of PSA input dataset
n_time_init_psa_series <- Sys.time()
for (i in 1:n_sim) { # i <- 1
  l_psa_input <- update_param_list(l_params_all, df_psa_input[i,])
  # Outcomes
  l_out_ce_temp  <- calculate_ce_out(l_psa_input)
  df_c[i, ]  <- l_out_ce_temp$Cost
  df_e[i, ]  <- l_out_ce_temp$Effect
  # Display simulation progress
  if (i/(n_sim/100) == round(i/(n_sim/100), 0)) { # display progress every 5%
    cat('\r', paste(i/n_sim * 100, "% done", sep = " "))
  }
}
```

```
##  1 % done 2 % done 3 % done 4 % done 5 % done 6 % done 7 % done 8 % done 9 % done 10 % done 11 % don
```

```r
n_time_end_psa_series <- Sys.time()
n_time_total_psa_series <- n_time_end_psa_series - n_time_init_psa_series
print(paste0("PSA with ", scales::comma(n_sim), " simulations run in series in ",
             round(n_time_total_psa_series, 2), " ",
             units(n_time_total_psa_series)))
```

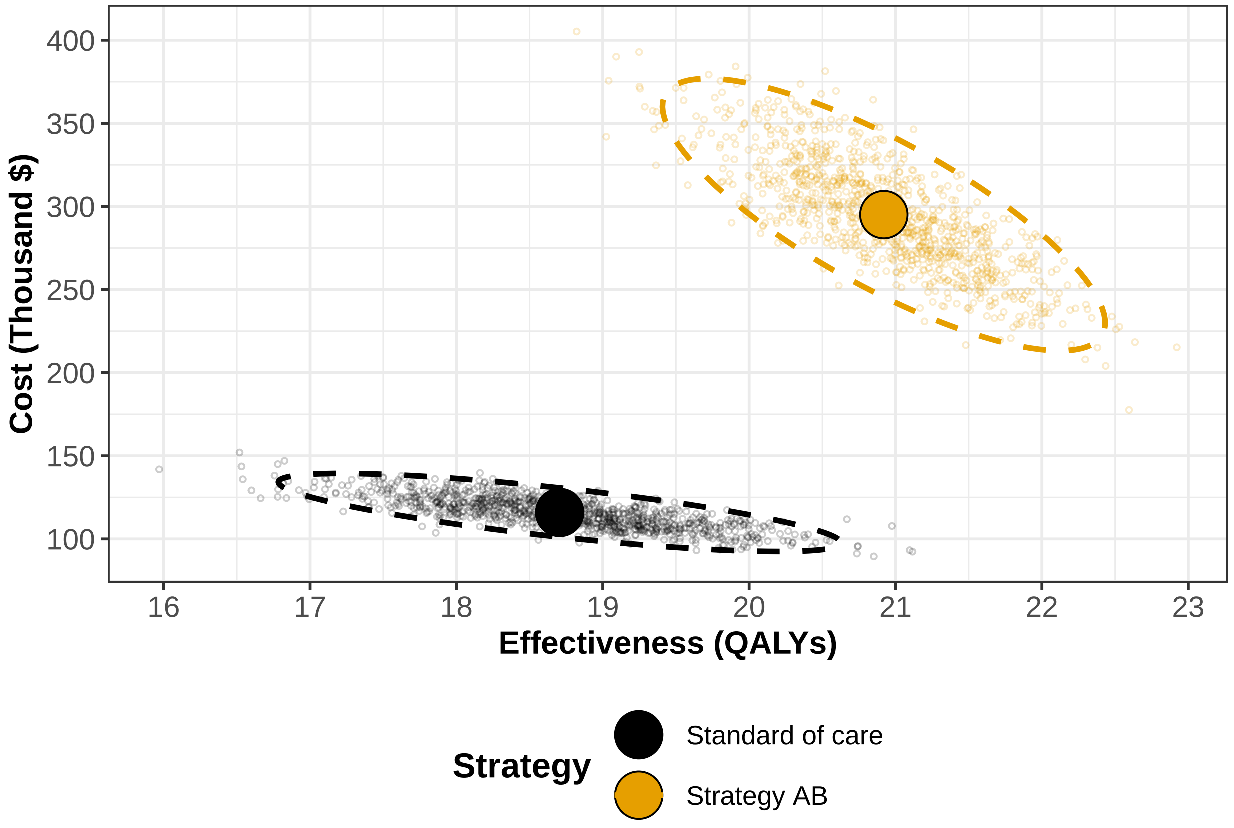
```
## [1] "PSA with 1,000 simulations run in series in 4.81 secs"
```

11.3 Visualize PSA results for CEA

```r
### Create PSA object
l_psa <- make_psa_obj(cost          = df_c,
                      effectiveness = df_e,
                      parameters    = df_psa_input,
                      strategies    = v_names_str)
l_psa$strategies <- v_names_str
colnames(l_psa$effectiveness) <- v_names_str
colnames(l_psa$cost) <- v_names_str

# Vector with willingness-to-pay (WTP) thresholds.
v_wtp <- seq(0, 200000, by = 5000)
```

11.3.1 Cost-Effectiveness Scatter plot

```r
### Cost-Effectiveness Scatter plot
txtsize <- 13
gg_scattter <- plot_psa(l_psa, txtsize = txtsize) +
  ggthemes::scale_color_colorblind() +
  ggthemes::scale_fill_colorblind() +
  scale_y_continuous("Cost (Thousand $)",
                     breaks = number_ticks(10),
                     labels = function(x) x/1000) +
  xlab("Effectiveness (QALYs)") +
  guides(col = guide_legend(nrow = 2)) +
  theme(legend.position = "bottom")
gg_scattter
```



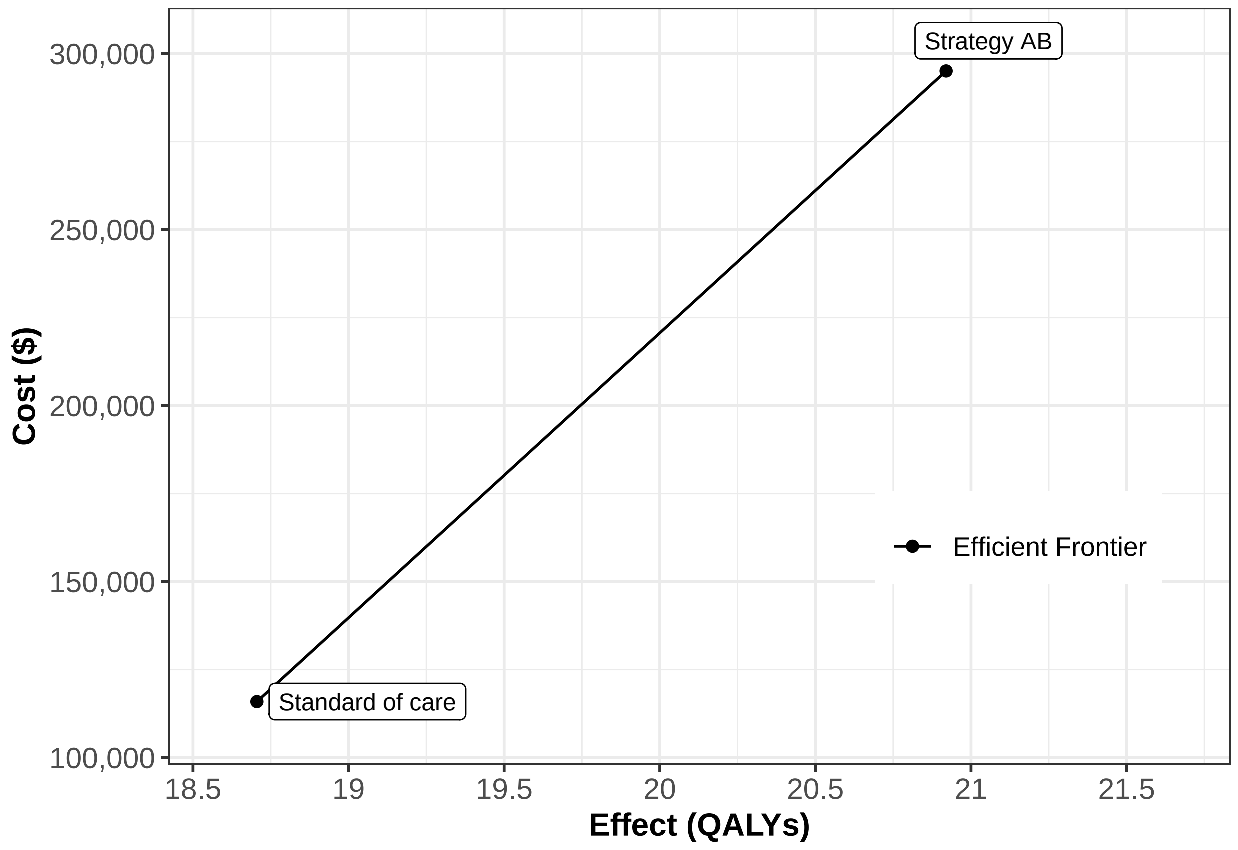
11.3.2 Incremental cost-effectiveness ratios (ICERs) with probabilistic output

```
### Incremental cost-effectiveness ratios (ICERs) with probabilistic output
# Compute expected costs and effects for each strategy from the PSA
df_out_ce_psa <- summary(l_psa)
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
```

```
##          Strategy     Cost   Effect Inc_Cost Inc_Effect     ICER Status
## 1 Standard of care 115908.2 18.70547       NA         NA       NA     ND
## 2      Strategy AB 295075.0 20.92011 179166.8   2.214639 80901.14     ND
```

11.3.3 Plot cost-effectiveness frontier with probabilistic output

```
### Plot cost-effectiveness frontier with probabilistic output
plot_icers(df_cea_psa, label = "all", txtsize = txtsize) +
  expand_limits(x = max(table_cea$QALYs) + 0.1) +
  theme(legend.position = c(0.8, 0.3))
```

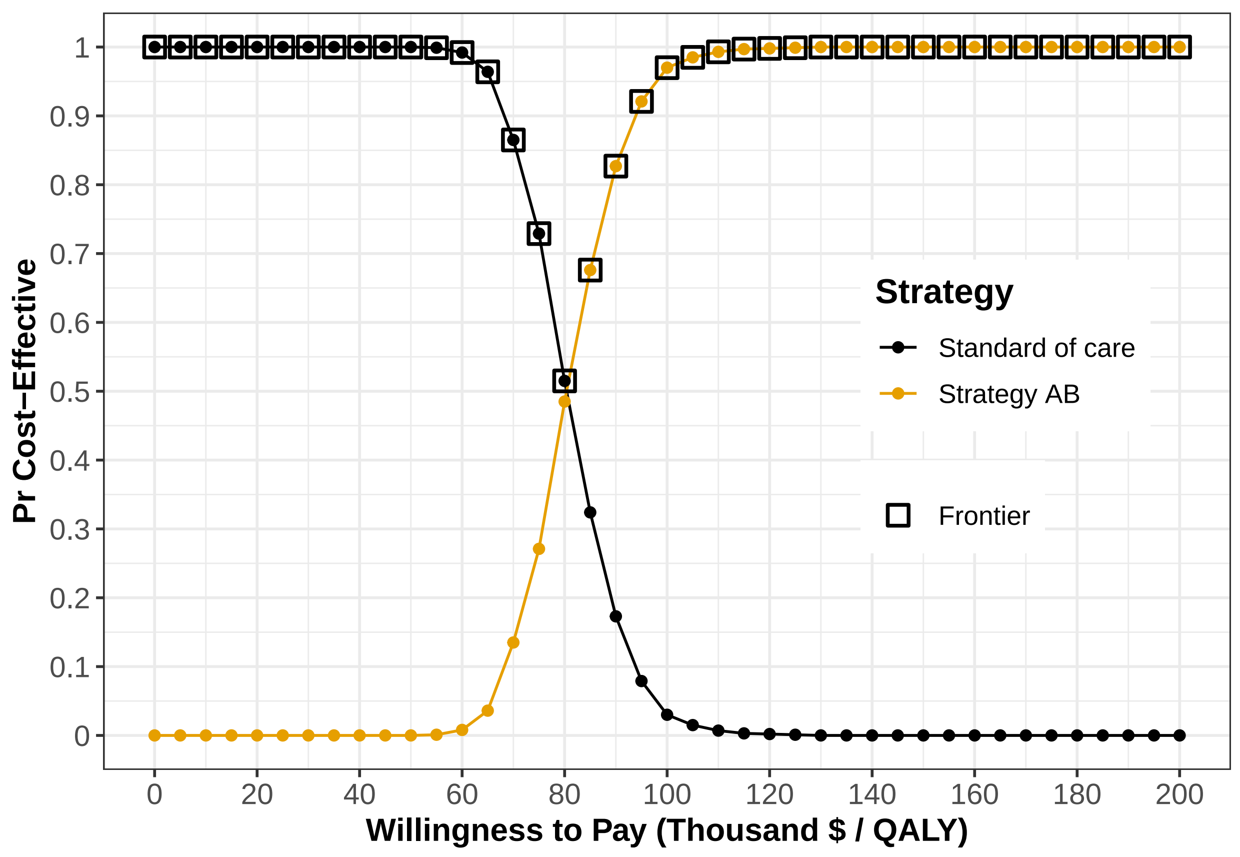


11.3.4 Cost-effectiveness acceptability curves (CEACs) and frontier (CEAF)

```
### 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)
```

```
##   range_min range_max   cost_eff_strat
## 1         0     85000 Standard of care
## 2     85000    200000      Strategy AB
```

```
# CEAC & CEAF plot
gg_ceac <- plot_ceac(ceac_obj, txtsize = txtsize, xlim = c(0, NA), n_x_ticks = 14) +
  ggthemes::scale_color_colorblind() +
  ggthemes::scale_fill_colorblind() +
  theme(legend.position = c(0.8, 0.48))
gg_ceac
```

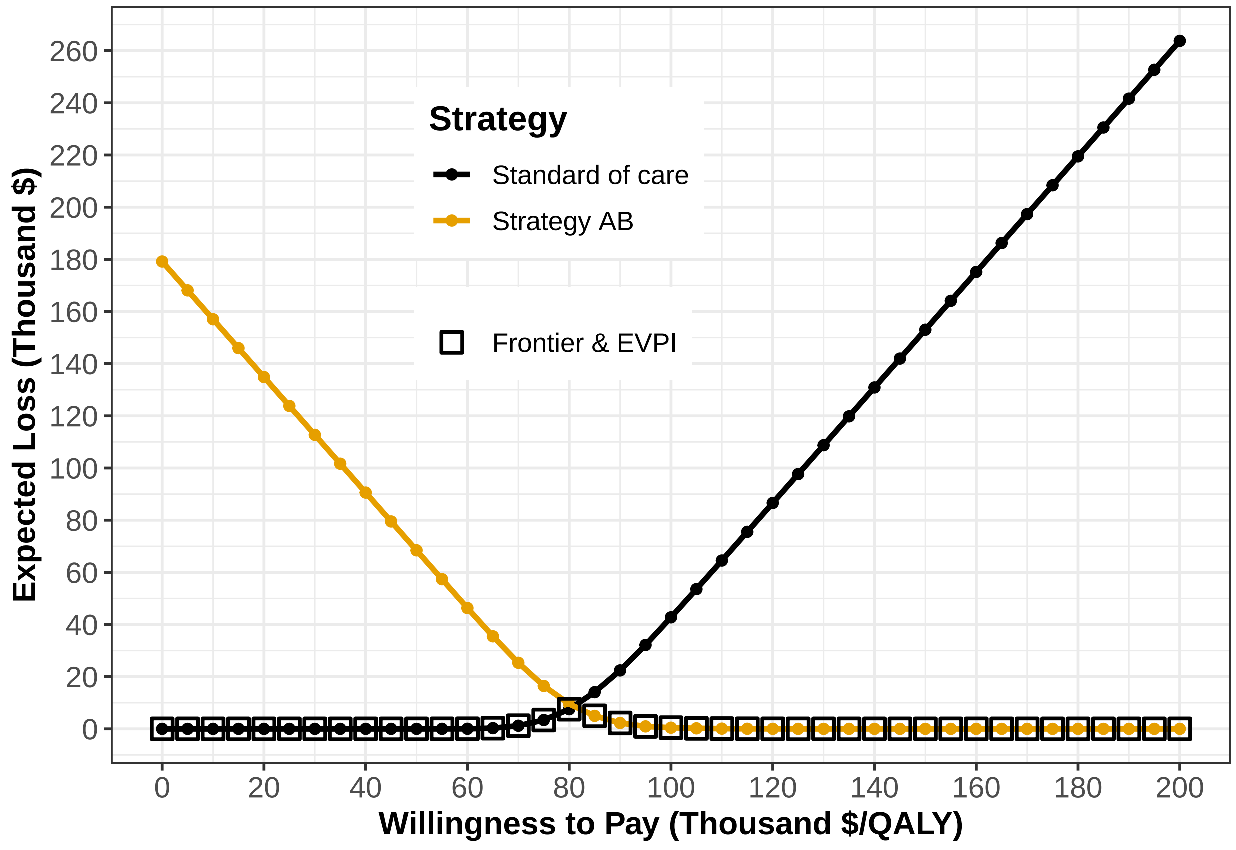


11.3.5 Expected Loss Curves (ELCs)

```
### Expected Loss Curves (ELCs)
elc_obj <- calc_exp_loss(wtp = v_wtp, psa = l_psa)
elc_obj
```

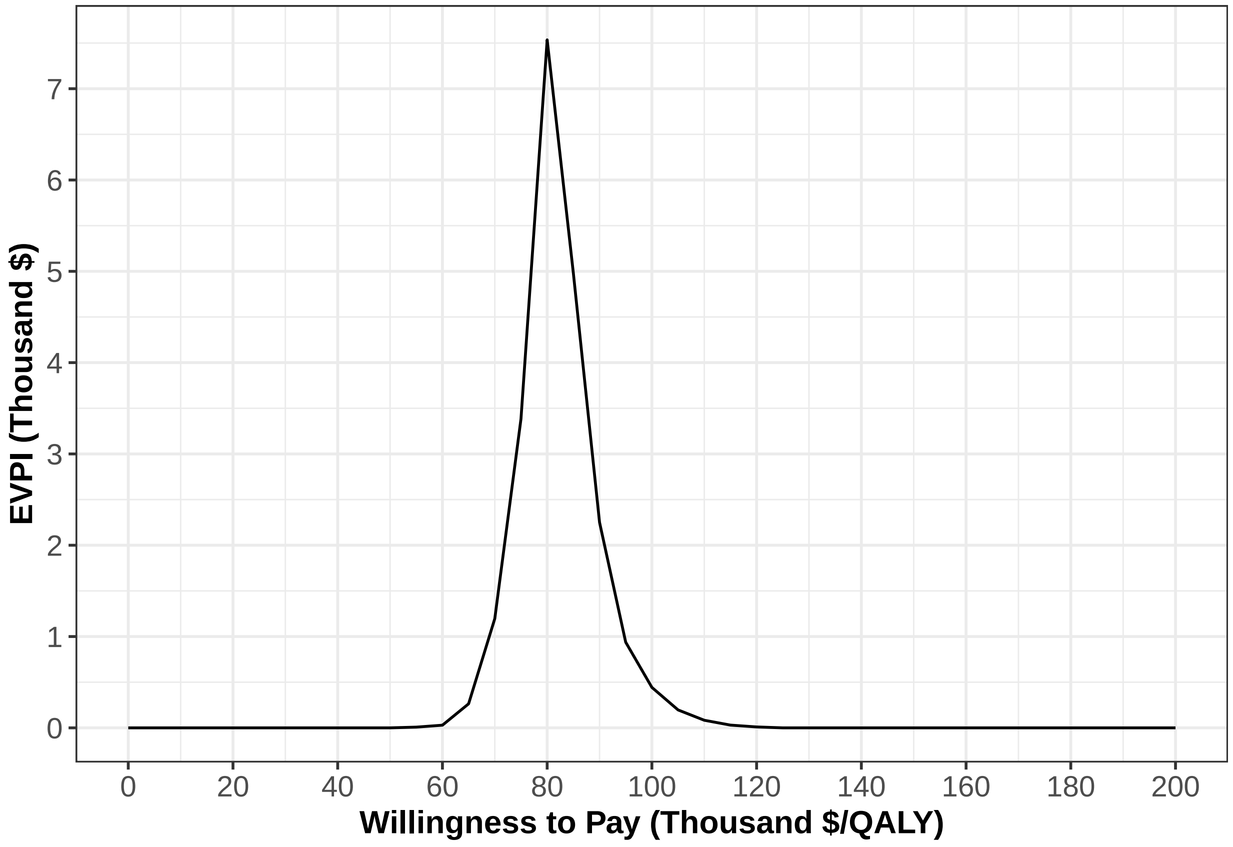
```
##        WTP         Strategy   Expected_Loss On_Frontier
## 1        0 Standard of care      0.00000000        TRUE
## 2        0      Strategy AB 179166.84111189       FALSE
## 3     5000 Standard of care      0.00000000        TRUE
## 4     5000      Strategy AB 168093.64420636       FALSE
## 5    10000 Standard of care      0.00000000        TRUE
## 6    10000      Strategy AB 157020.44730082       FALSE
## 7    15000 Standard of care      0.00000000        TRUE
## 8    15000      Strategy AB 145947.25039529       FALSE
## 9    20000 Standard of care      0.00000000        TRUE
## 10   20000      Strategy AB 134874.05348975       FALSE
## 11   25000 Standard of care      0.00000000        TRUE
## 12   25000      Strategy AB 123800.85658422       FALSE
## 13   30000 Standard of care      0.00000000        TRUE
## 14   30000      Strategy AB 112727.65967868       FALSE
## 15   35000 Standard of care      0.00000000        TRUE
## 16   35000      Strategy AB 101654.46277314       FALSE
## 17   40000 Standard of care      0.00000000        TRUE
## 18   40000      Strategy AB  90581.26586761       FALSE
## 19   45000 Standard of care      0.00000000        TRUE
## 20  45000       Strategy AB  79508.06896207       FALSE
## 21  50000 Standard of care      0.00000000        TRUE
## 22  50000       Strategy AB  68434.87205654       FALSE
## 23  55000 Standard of care      7.79331782        TRUE
## 24  55000       Strategy AB  57369.46846883       FALSE
## 25  60000 Standard of care     28.95908467        TRUE
## 26  60000       Strategy AB  46317.43733014       FALSE
## 27  65000 Standard of care    263.13980366        TRUE
## 28  65000       Strategy AB  35478.42114360       FALSE
## 29  70000 Standard of care   1195.75830262        TRUE
## 30  70000       Strategy AB  25337.84273702       FALSE
## 31  75000 Standard of care   3383.87077235        TRUE
## 32  75000       Strategy AB  16452.75830121       FALSE
## 33  80000 Standard of care   7534.96974238        TRUE
## 34  80000       Strategy AB   9530.66036570       FALSE
## 35  85000 Standard of care  14058.14964550       FALSE
## 36  85000       Strategy AB   4980.64336329        TRUE
## 37  90000 Standard of care  22401.51600768       FALSE
## 38  90000       Strategy AB   2250.81281993        TRUE
## 39  95000 Standard of care  32161.76748056       FALSE
## 40  95000       Strategy AB    937.86738728        TRUE
## 41 100000 Standard of care  42740.07083657       FALSE
## 42 100000       Strategy AB    442.97383775        TRUE
## 43 105000 Standard of care  53566.36291415       FALSE
## 44 105000       Strategy AB    196.06900980        TRUE
## 45 110000 Standard of care  64526.51511177       FALSE
## 46 110000       Strategy AB     83.02430188        TRUE
## 47 115000 Standard of care  75546.59381131       FALSE
## 48 115000       Strategy AB     29.90609589        TRUE
## 49 120000 Standard of care  86599.80991635       FALSE
## 50 120000       Strategy AB      9.92529539        TRUE
## 51 125000 Standard of care  97663.09274292       FALSE
## 52 125000       Strategy AB      0.01121643        TRUE
## 53 130000 Standard of care 108736.27843203       FALSE
## 54 130000       Strategy AB      0.00000000        TRUE
## 55 135000 Standard of care 119809.47533757       FALSE
## 56 135000       Strategy AB      0.00000000        TRUE
## 57 140000 Standard of care 130882.67224310       FALSE
## 58 140000       Strategy AB      0.00000000        TRUE
## 59 145000 Standard of care 141955.86914864       FALSE
## 60 145000       Strategy AB      0.00000000        TRUE
## 61 150000 Standard of care 153029.06605417       FALSE
## 62 150000       Strategy AB      0.00000000        TRUE
## 63 155000 Standard of care 164102.26295971       FALSE
## 64 155000       Strategy AB      0.00000000        TRUE
## 65 160000 Standard of care 175175.45986524       FALSE
## 66 160000       Strategy AB      0.00000000        TRUE
## 67 165000 Standard of care 186248.65677078       FALSE
## 68 165000       Strategy AB      0.00000000        TRUE
## 69 170000 Standard of care 197321.85367631       FALSE
## 70 170000       Strategy AB      0.00000000        TRUE
## 71 175000 Standard of care 208395.05058185       FALSE
## 72 175000       Strategy AB      0.00000000        TRUE
## 73 180000 Standard of care 219468.24748739       FALSE
## 74 180000       Strategy AB      0.00000000        TRUE
## 75 185000 Standard of care 230541.44439292       FALSE
## 76 185000       Strategy AB      0.00000000        TRUE
## 77 190000 Standard of care 241614.64129846       FALSE
## 78 190000       Strategy AB      0.00000000        TRUE
## 79 195000 Standard of care 252687.83820399       FALSE
## 80 195000       Strategy AB      0.00000000        TRUE
## 81 200000 Standard of care 263761.03510953       FALSE
## 82 200000       Strategy AB      0.00000000        TRUE
```

```
# ELC plot
gg_elc <- plot_exp_loss(elc_obj, log_y = FALSE,
               txtsize = txtsize, xlim = c(0, NA), n_x_ticks = 14,
               col = "full") +
  ggthemes::scale_color_colorblind() +
  ggthemes::scale_fill_colorblind() +
  # geom_point(aes(shape = as.name("Strategy"))) +
  scale_y_continuous("Expected Loss (Thousand $)",
                     breaks = number_ticks(10),
                     labels = function(x) x/1000) +
  theme(legend.position = c(0.4, 0.7),)
gg_elc
```



11.3.6 Expected value of perfect information (EVPI)

```
### Expected value of perfect information (EVPI)
evpi <- calc_evpi(wtp = v_wtp, psa = l_psa)
# EVPI plot
gg_evpi <- plot_evpi(evpi, effect_units = "QALY",
                     txtsize = txtsize, xlim = c(0, NA), n_x_ticks = 14) +
  scale_y_continuous("EVPI (Thousand $)",
                     breaks = number_ticks(10),
                     labels = function(x) x/1000)
gg_evpi
```



REFERENCES

- 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 (In press): 1-21. https://doi.org/10.1177/0272989X221121747
- 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/10.1177/0272989X2211 03163
- 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/10.1177/0272989X2211

03163

- 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