SA: Markov Sick-Sicker 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. 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. 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
p_load_gh("DARTH-git/dampack", "DARTH-git/darthtools")
```

02 Load functions

```
# No function needed
```

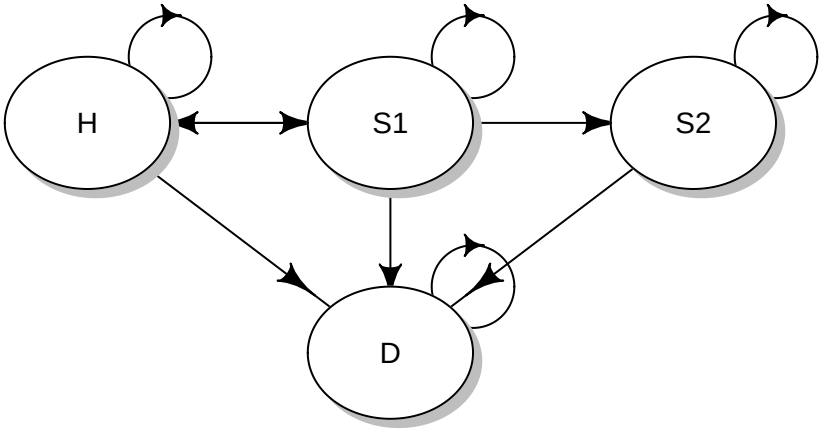
03 Input model parameters

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

# 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)
v_init   <- c(1, 0, 0, 0)            # initial cohort distribution

# Transition probabilities (per cycle)
p_HD     <- 0.005                    # probability to die when healthy
p_HS1    <- 0.15                     # probability to become sick when healthy, conditional on survivin
p_S1H    <- 0.5                      # probability to become healthy when sick, conditional on survivin
p_S1S2   <- 0.105                    # probability to become 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
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
```

```r
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
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)
```

Create a state-transition diagram of the cohort model

```r
m_P_diag <- matrix(0, nrow = n_states, ncol = n_states, dimnames = list(v_n, v_n))
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 Define and initialize matrices and vectors

04.1 Cohort trace

```
# 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_states,
                               dimnames = list(paste("cycle", 0:n_t, sep = " "), v_n))

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

```
##          H S1 S2  D
## cycle 0 NA NA NA NA
## cycle 1 NA NA NA NA
## cycle 2 NA NA NA NA
## cycle 3 NA NA NA NA
## cycle 4 NA NA NA NA
## cycle 5 NA NA NA NA
```

```
# The cohort starts as healthy
m_M_notrt[1, ] <- m_M_trt[1, ] <- c(1, 0, 0, 0) # initiate first cycle of cohort trace
```

04.2 Transition probability matrix

```
# create the transition probability matrix for NO treatment
m_P_notrt  <- matrix(0,
                     nrow = n_states,
                     ncol = n_states,
                     dimnames = list(v_n, v_n)) # name the columns and rows of the matrix
m_P_notrt
```

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

Fill in the transition probability matrix:

```
# from Healthy
m_P_notrt["H", "H"  ] <- (1 - p_HD) * (1 - p_HS1)
m_P_notrt["H", "S1" ] <- (1 - p_HD) * p_HS1
m_P_notrt["H", "D"  ] <- p_HD
# from Sick
m_P_notrt["S1", "H" ] <- (1 - p_S1D) * p_S1H
m_P_notrt["S1", "S1"] <- (1 - p_S1D) * (1 - (p_S1H + p_S1S2))
m_P_notrt["S1", "S2"] <- (1 - p_S1D) * p_S1S2
m_P_notrt["S1", "D" ] <- p_S1D
# from Sicker
m_P_notrt["S2", "S2"] <- 1 - p_S2D
m_P_notrt["S2", "D" ] <- p_S2D
# from Dead
m_P_notrt["D", "D"  ] <- 1

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

# create transition probability matrix for treatment same as no treatment
m_P_trt <- m_P_notrt
```

05 Run Markov model

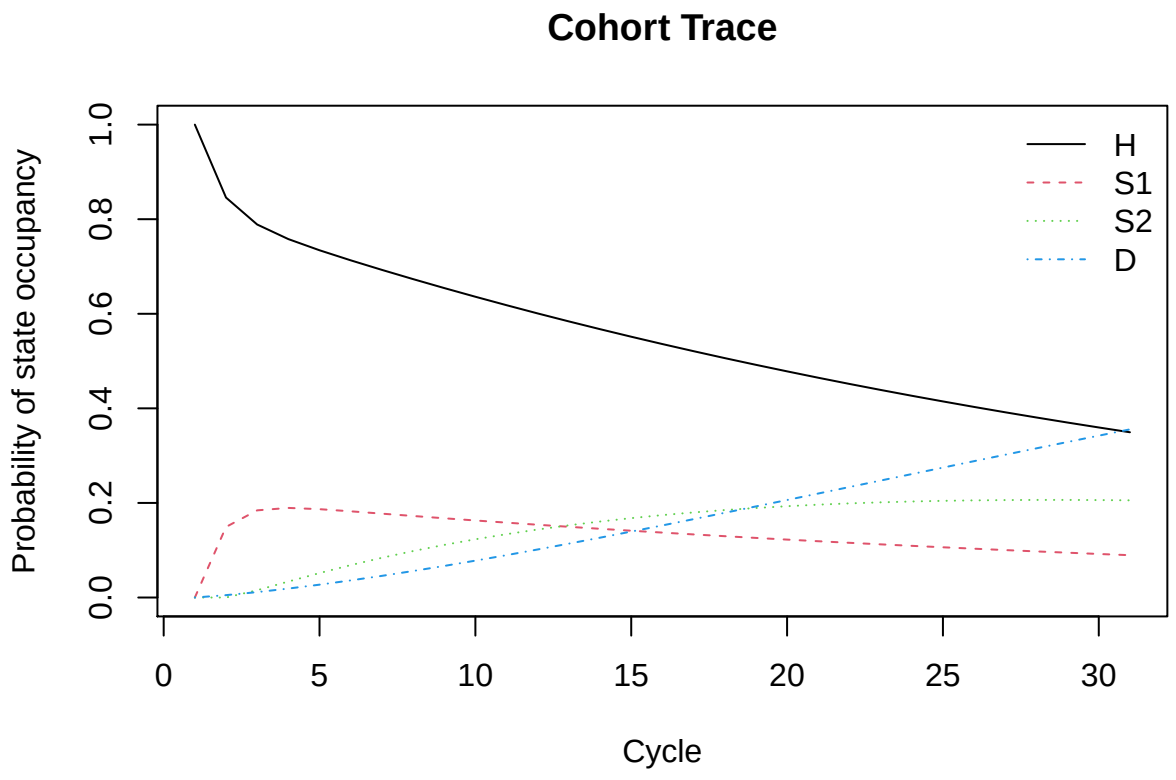
```
for (t in 1:n_t){     # loop through the number of cycles
  m_M_notrt[t + 1, ] <- t(m_M_notrt[t, ]) %*% m_P_notrt  # estimate the Markov trace
                                                         # for the next cycle (t + 1)
  m_M_trt[t + 1, ]   <- t(m_M_trt[t, ])   %*% m_P_trt    # estimate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

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

```
##                 H        S1         S2          D
## cycle 0 1.0000000 0.0000000 0.00000000 0.00000000
## cycle 1 0.8457500 0.1492500 0.00000000 0.00500000
## cycle 2 0.7888043 0.1843020 0.01543735 0.01145632
## cycle 3 0.7579069 0.1894418 0.03374551 0.01890581
## cycle 4 0.7343069 0.1868303 0.05169021 0.02717260
## cycle 5 0.7130610 0.1822918 0.06848747 0.03615973
```

06 Compute and Plot Epidemiological Outcomes

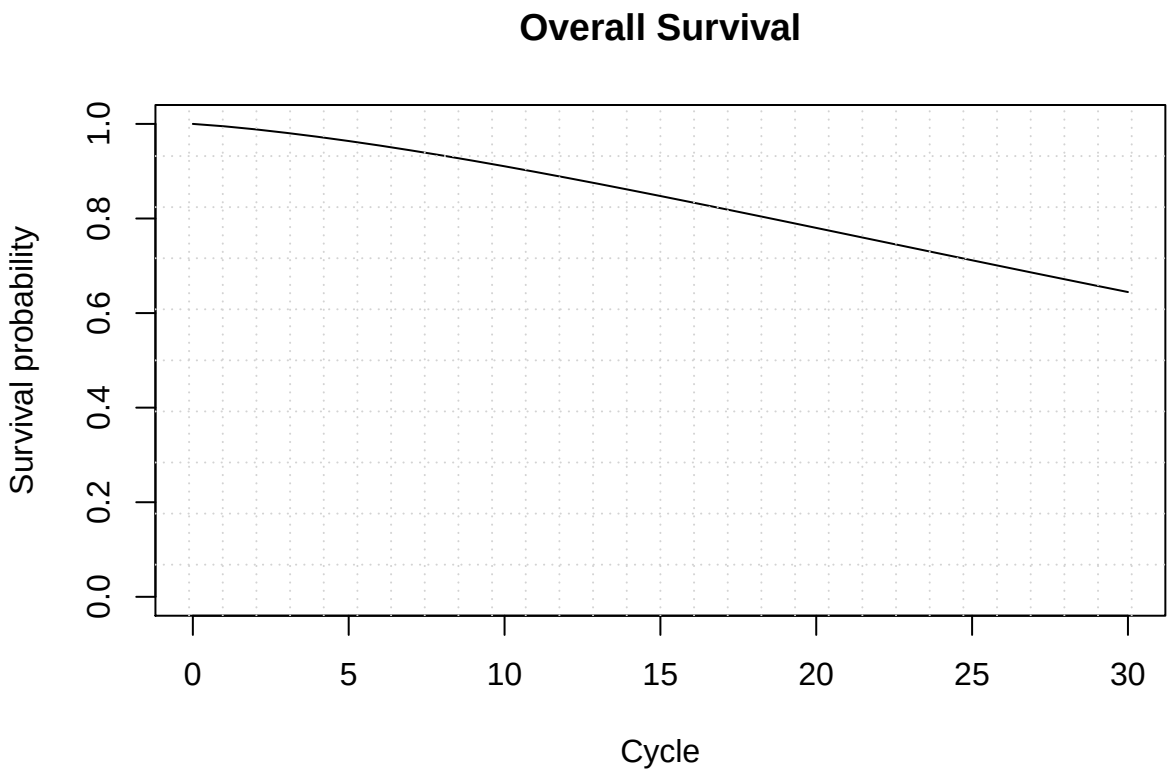
06.1 Cohort trace

```r
# create a plot of the data
matplot(m_M_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_states, lty = 1:n_states, bty = "n")
```



06.2 Overall Survival (OS)

```
# calculate the overall survival (OS) probability for no treatment
v_os_notrt <- 1 - m_M_notrt[, "D"]
# alternative way of calculating the OS probability
v_os_notrt <- rowSums(m_M_notrt[, 1:3])
# create a simple plot showing the OS
plot(0:n_t, v_os_notrt, type = 'l',
     ylim = c(0, 1),
     ylab = "Survival probability",
     xlab = "Cycle",
     main = "Overall Survival")
# add grid
grid(nx = n_t, ny = 10, col = "lightgray", lty = "dotted", lwd = par("lwd"),
     equilogs = TRUE)
```

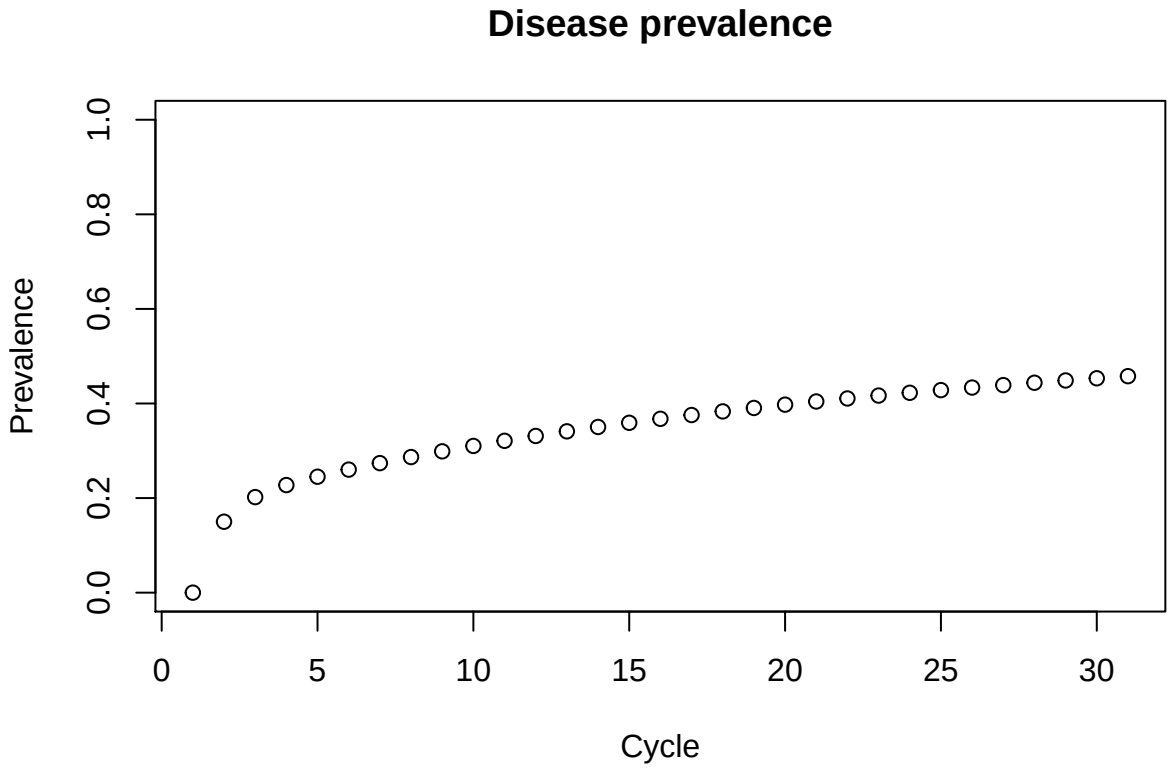


06.2.1 Life Expectancy (LE)

```
v_le <- sum(v_os_notrt)  # summing probability of OS over time  (i.e. life expectancy)
```

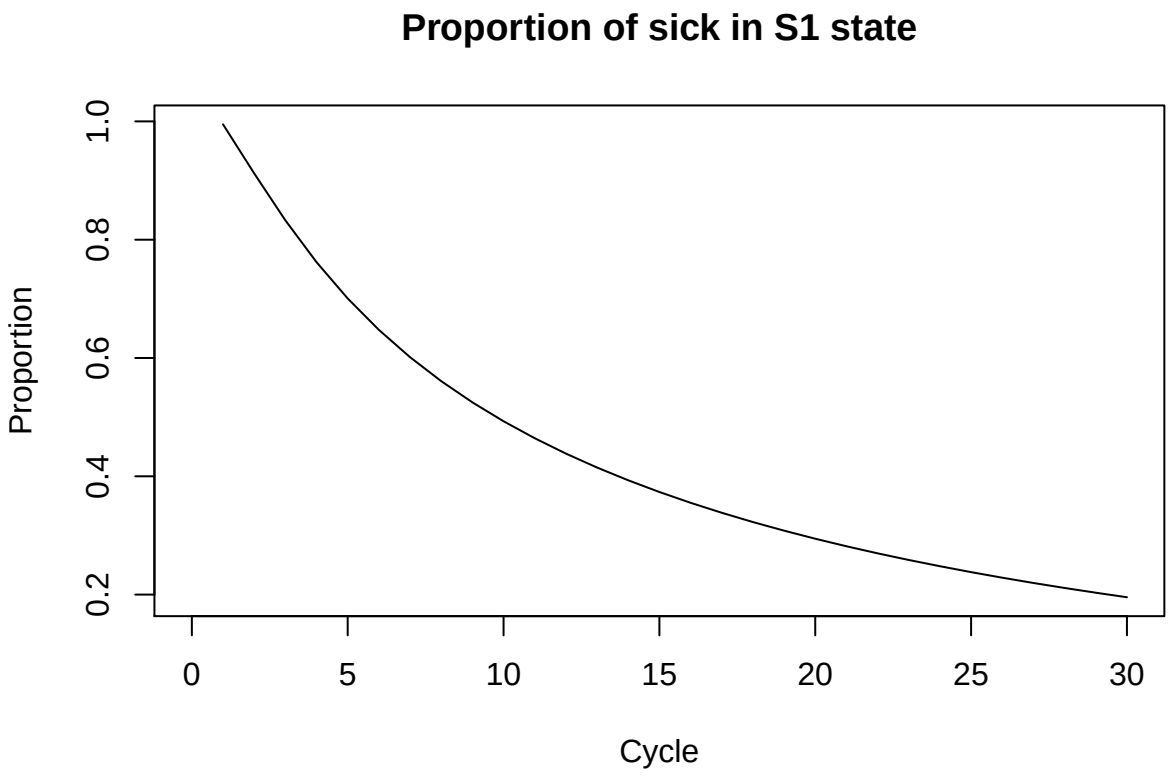
06.3 Disease prevalence

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



06.4 Proportion of sick in S1 state

```
v_prop_S1 <- m_M_notrt[, "S1"] / v_prev
plot(0:n_t, v_prop_S1,
     xlab = "Cycle",
     ylab = "Proportion",
     main = "Proportion of sick in S1 state",
     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_notrt   %*%  v_u_notrt
v_tu_trt    <- m_M_trt     %*%  v_u_trt

v_tc_notrt  <- m_M_notrt   %*%  v_c_notrt
v_tc_trt    <- m_M_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
```

```
##       Strategy      Cost   Effect
## 1 No Treatment  75795.04 15.84802
## 2    Treatment 141511.41 16.41446
```

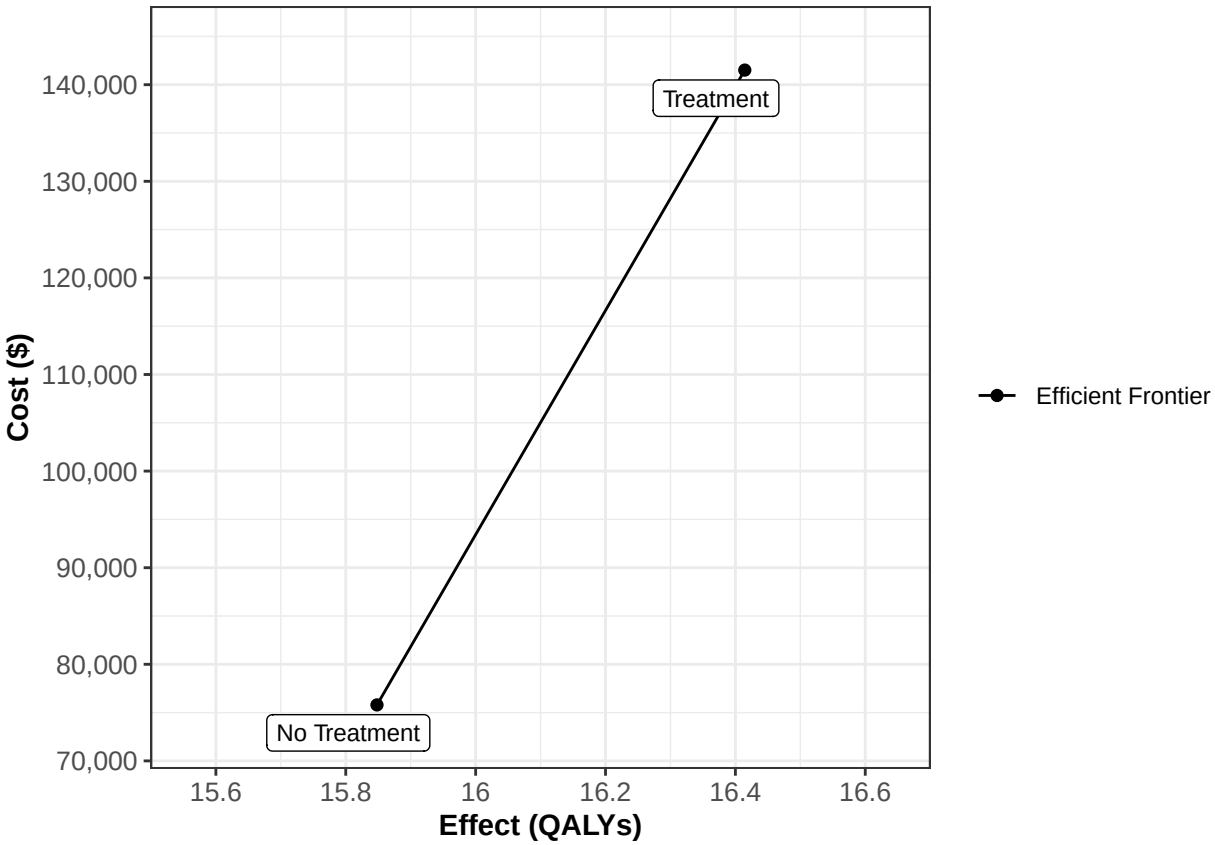
07.3 Compute ICERs of the Markov model

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

```
##       Strategy      Cost   Effect Inc_Cost Inc_Effect     ICER Status
## 1 No Treatment  75795.04 15.84802       NA         NA       NA     ND
## 2    Treatment 141511.41 16.41446 65716.37  0.5664367 116017.2     ND
```

07.4 Plot frontier of the Markov model

```
plot(df_cea, effect_units = "QALYs", xlim = c(15.6, 16.6))
```



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    = 0.005,  # probability to die when healthy
  p_HS1   = 0.15,   # probability to become sick when healthy, conditional on surviving
  p_S1H   = 0.5,    # probability to become healthy when sick, conditional on surviving
  p_S1S2  = 0.105,  # probability to become 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
  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 <- names(l_params_all)
```

08.2 Load Sick-Sicker Markov model function

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

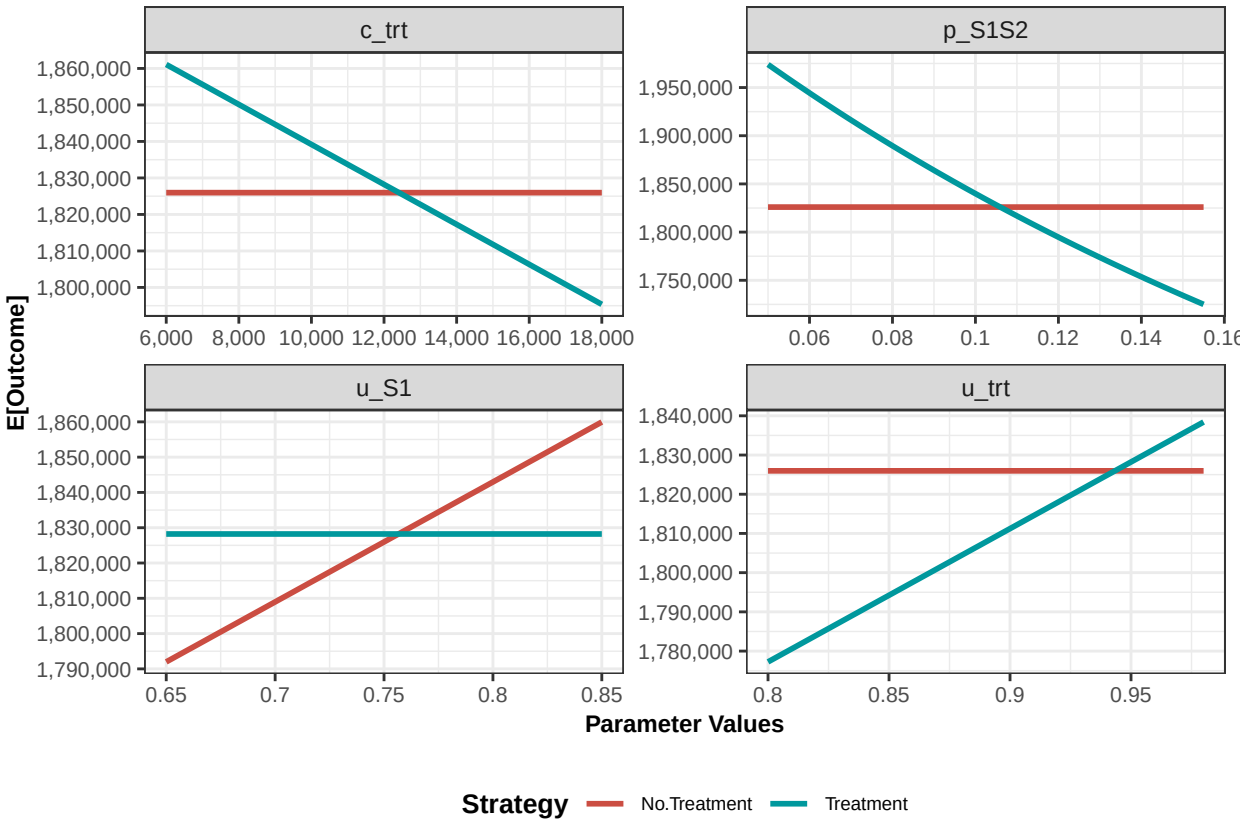
```
##       Strategy      Cost   Effect     NMB
## 1 No Treatment  75795.04 15.84802 1509007
## 2    Treatment 141511.41 16.41446 1499935
```

08.3 One-way sensitivity analysis (OWSA)

```
options(scipen = 999) # disabling scientific notation in R
# dataframe containing all parameters, their basecase values, and the min and
# max values of the parameters of interest
df_params_owsa <- data.frame(pars = c("p_S1S2", "c_trt", "u_S1", "u_trt"),
                             min  = c(0.05 ,  6000 , 0.65, 0.80),  # min parameter values
                             max  = c(0.155, 18000 , 0.85, 0.98)   # 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            = 120000)            # extra argument to pass to FUN
```

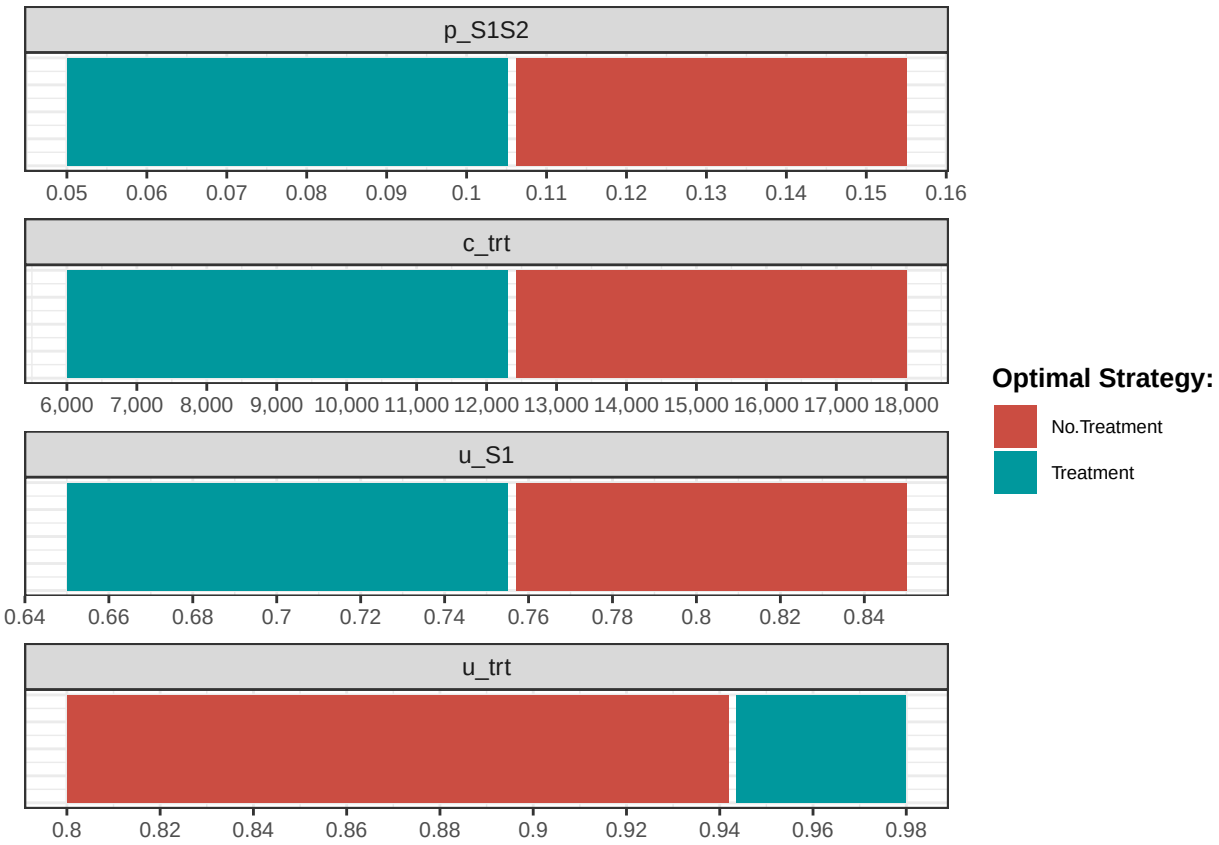
```
##   |                                                                      |
```

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



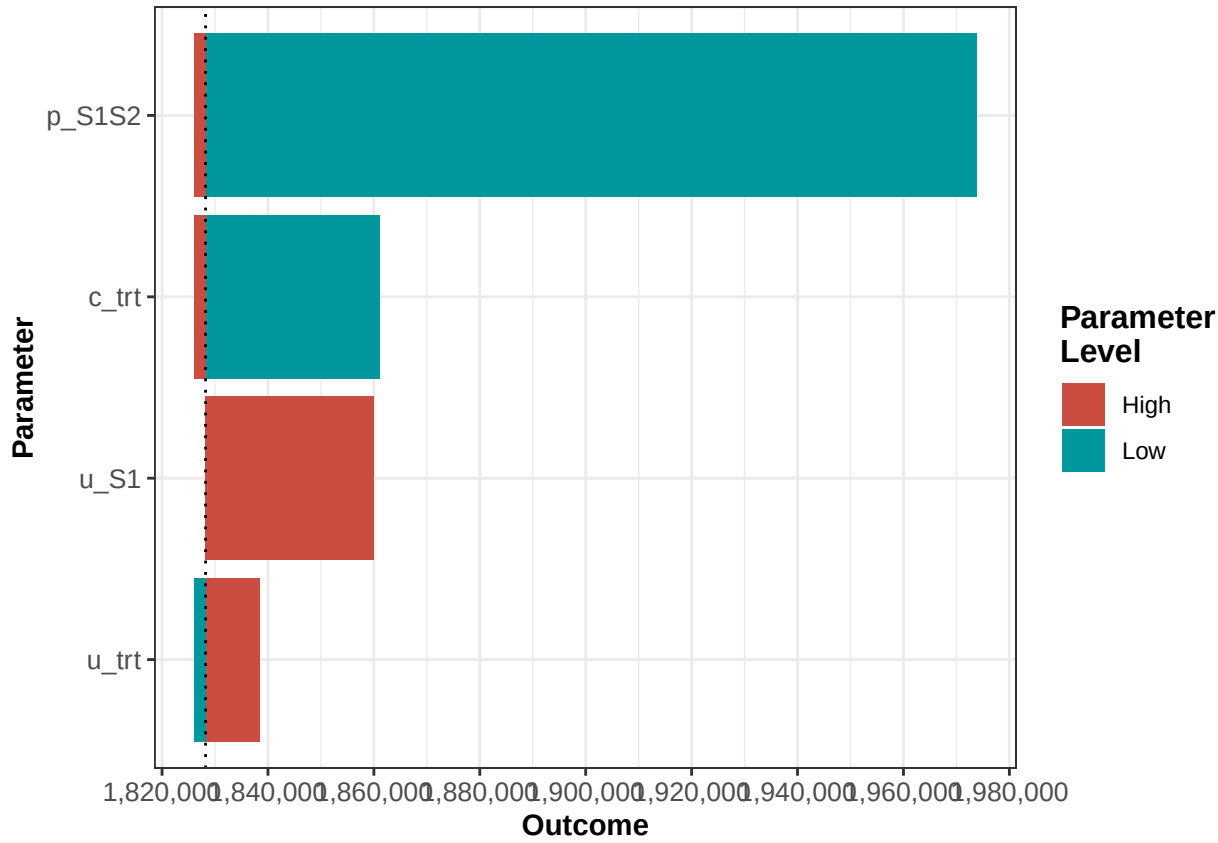
08.3.2 Optimal strategy with OWSA

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



08.3.3 Tornado plot

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

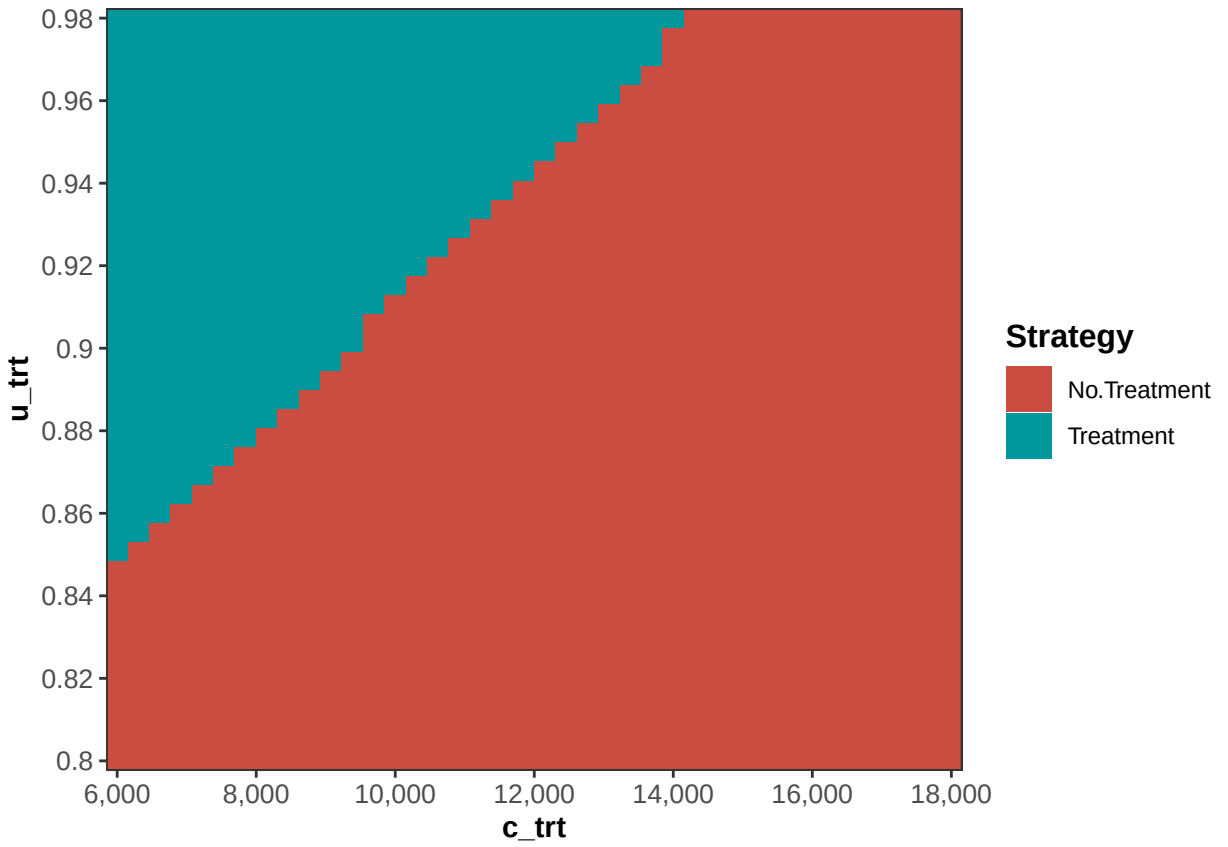


08.4 Two-way sensitivity analysis (TWSA)

```
# 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_trt"),
                             min  = c(6000, 0.80),  # min parameter values
                             max  = c(18000, 0.98)  # 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           = 120000)            # extra argument to pass to FUN
```

```
##   |                                                                      |
```

```
plot(twsa_nmb)
```



09 Probabilistic Sensitivity Analysis (PSA)

```r
# 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)
    p_HS1   = rbeta(n_sim, shape1 = 30, shape2 = 170),     # probability to become sick when healthy
    p_S1H   = rbeta(n_sim, shape1 = 60, shape2 = 60) ,     # probability to become healthy when sick
    p_S1S2  = rbeta(n_sim, shape1 = 84, shape2 = 716),     # probability to become sicker when sick
    p_HD    = rbeta(n_sim, shape1 = 10, shape2 = 1990),    # probability to die when healthy
    hr_S1   = rlnorm(n_sim, meanlog = log(3),  sdlog = 0.01),  # rate ratio of death in S1 vs healthy
    hr_S2   = rlnorm(n_sim, meanlog = log(10), sdlog = 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
```

```r
    u_H   = rbeta(n_sim, shape1 = 200, shape2 = 3),      # utility when healthy
    u_S1  = rbeta(n_sim, shape1 = 130, shape2 = 45),     # utility when sick
    u_S2  = rbeta(n_sim, shape1 = 230, shape2 = 230),    # utility when sicker
    u_D   = 0,                                           # utility when dead
    u_Trt = rbeta(n_sim, shape1 = 300, shape2 = 15),     # 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
gen_psa(10)
```

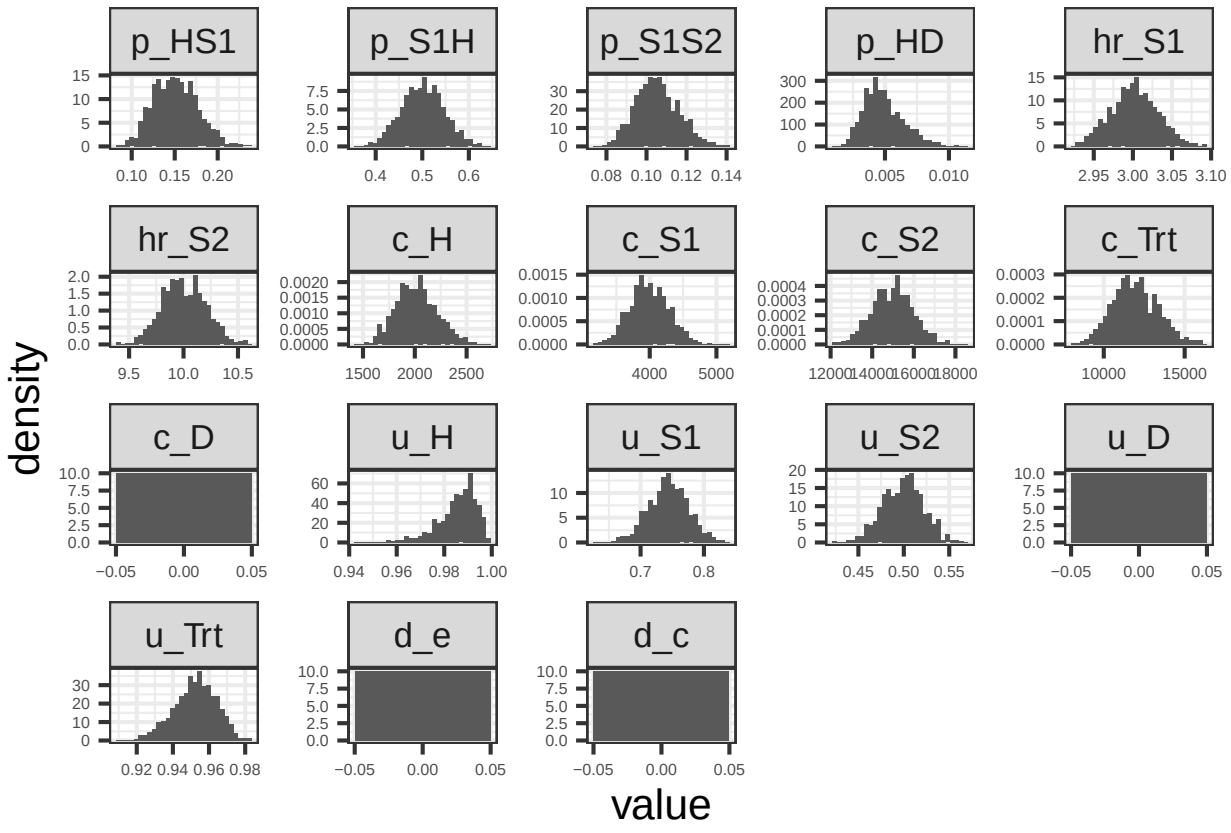
```
##         p_HS1     p_S1H     p_S1S2        p_HD    hr_S1     hr_S2      c_H
## 1  0.15132349 0.4820445 0.08485456 0.003973762 3.055410 10.148768 1887.449
## 2  0.12522484 0.4392546 0.11683397 0.004813328 3.005252  9.928044 1966.291
## 3  0.15673720 0.5311020 0.11925841 0.004491843 2.982392 10.021998 2101.059
## 4  0.11276858 0.4493088 0.09393848 0.010646056 2.981918 10.167486 1728.664
## 5  0.14292946 0.5825087 0.11628065 0.007045338 3.024975 10.006505 2160.895
## 6  0.20088601 0.4580278 0.10403415 0.004058637 2.971345 10.012715 1969.304
## 7  0.14347796 0.5492245 0.09294048 0.005658464 2.989181 10.001860 1641.192
## 8  0.13149270 0.4899896 0.12838861 0.007354969 2.984080  9.919332 2276.032
## 9  0.15593267 0.5705602 0.09237907 0.004162148 2.988974 10.142331 1717.867
## 10 0.09824029 0.5127637 0.10683148 0.002141140 2.968692  9.749869 2132.549
##        c_S1     c_S2    c_Trt c_D       u_H      u_S1      u_S2 u_D     u_Trt
## 1  4173.600 14607.88 10673.74   0 0.9829120 0.7252701 0.4757751   0 0.9435773
## 2  4188.264 14714.84 13380.61   0 0.9693926 0.7610392 0.4591563   0 0.9501528
## 3  4578.682 16664.54 12522.58   0 0.9910033 0.7076433 0.5032719   0 0.9670124
## 4  3395.922 15669.87 13833.65   0 0.9928844 0.7124119 0.5132492   0 0.9649128
## 5  3878.562 15302.70 13154.65   0 0.9748972 0.7344718 0.4895693   0 0.9325969
## 6  3970.522 15303.55 13914.95   0 0.9832065 0.7169406 0.4628316   0 0.9456881
## 7  3680.588 15378.20 12117.86   0 0.9917677 0.7851270 0.4504530   0 0.9093339
## 8  3781.956 15614.73 12070.99   0 0.9727485 0.7582961 0.4727426   0 0.9390084
## 9  3448.146 15401.50 11760.17   0 0.9836544 0.7648816 0.4851046   0 0.9537789
## 10 4656.666 15198.97 10216.98   0 0.9915345 0.7751203 0.5311969   0 0.9577708
##     d_e  d_c
## 1  0.03 0.03
## 2  0.03 0.03
## 3  0.03 0.03
## 4  0.03 0.03
## 5  0.03 0.03
## 6  0.03 0.03
## 7  0.03 0.03
## 8  0.03 0.03
## 9  0.03 0.03
## 10 0.03 0.03
```

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

# Generate PSA input dataset
df_psa_input <- gen_psa(n_sim = n_sim)
# First six observations
```

```r
head(df_psa_input)
```

```
##       p_HS1     p_S1H    p_S1S2        p_HD    hr_S1     hr_S2      c_H
## 1 0.1513235 0.5617628 0.1033817 0.006269864 2.962948  9.820157 1658.447
## 2 0.1252248 0.4380343 0.1152594 0.005371105 3.014068 10.100813 2050.281
## 3 0.1567372 0.5458102 0.1274505 0.004301015 3.018588  9.825985 2094.425
## 4 0.1127686 0.5106953 0.1015742 0.005319105 2.979709  9.697122 2135.411
## 5 0.1429295 0.5277308 0.1020384 0.004225209 3.025490 10.254387 1665.257
## 6 0.2008860 0.5252364 0.1030676 0.008319206 2.979175 10.366400 2049.794
##       c_S1     c_S2     c_Trt c_D       u_H      u_S1      u_S2 u_D     u_Trt
## 1 4191.109 15165.80 10292.050   0 0.9816602 0.7395884 0.5106656   0 0.9496993
## 2 4133.823 15042.66 10779.006   0 0.9906084 0.7240259 0.4927761   0 0.9731185
## 3 3567.033 15052.49 12811.443   0 0.9834960 0.6910869 0.5024542   0 0.9598494
## 4 3726.226 16084.60 10072.741   0 0.9919197 0.7065183 0.5118351   0 0.9357411
## 5 3654.486 15201.84 14917.228   0 0.9953850 0.6838064 0.5021483   0 0.9635901
## 6 3927.001 17060.57  9581.097   0 0.9813330 0.7766346 0.5035428   0 0.9506926
##    d_e  d_c
## 1 0.03 0.03
## 2 0.03 0.03
## 3 0.03 0.03
## 4 0.03 0.03
## 5 0.03 0.03
## 6 0.03 0.03
```

```r
# Histogram of parameters
ggplot(melt(df_psa_input, variable.name = "Parameter"), aes(x = value)) +
       facet_wrap(~Parameter, scales = "free") +
       geom_histogram(aes(y = ..density..)) +
       scale_x_continuous(breaks = scales::pretty_breaks(n = 3)) +
       theme_bw(base_size = 16) +
       theme(axis.text = element_text(size=6))
```




```
# 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

```
# 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 = " "))
  }
}
```

```
##  10 % done 20 % done 30 % done 40 % done 50 % done 60 % done 70 % done 80 % done 90 % done 100 % done
```

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_PSA_dataset.RData")
```

09.3 Create probabilistic analysis graphs

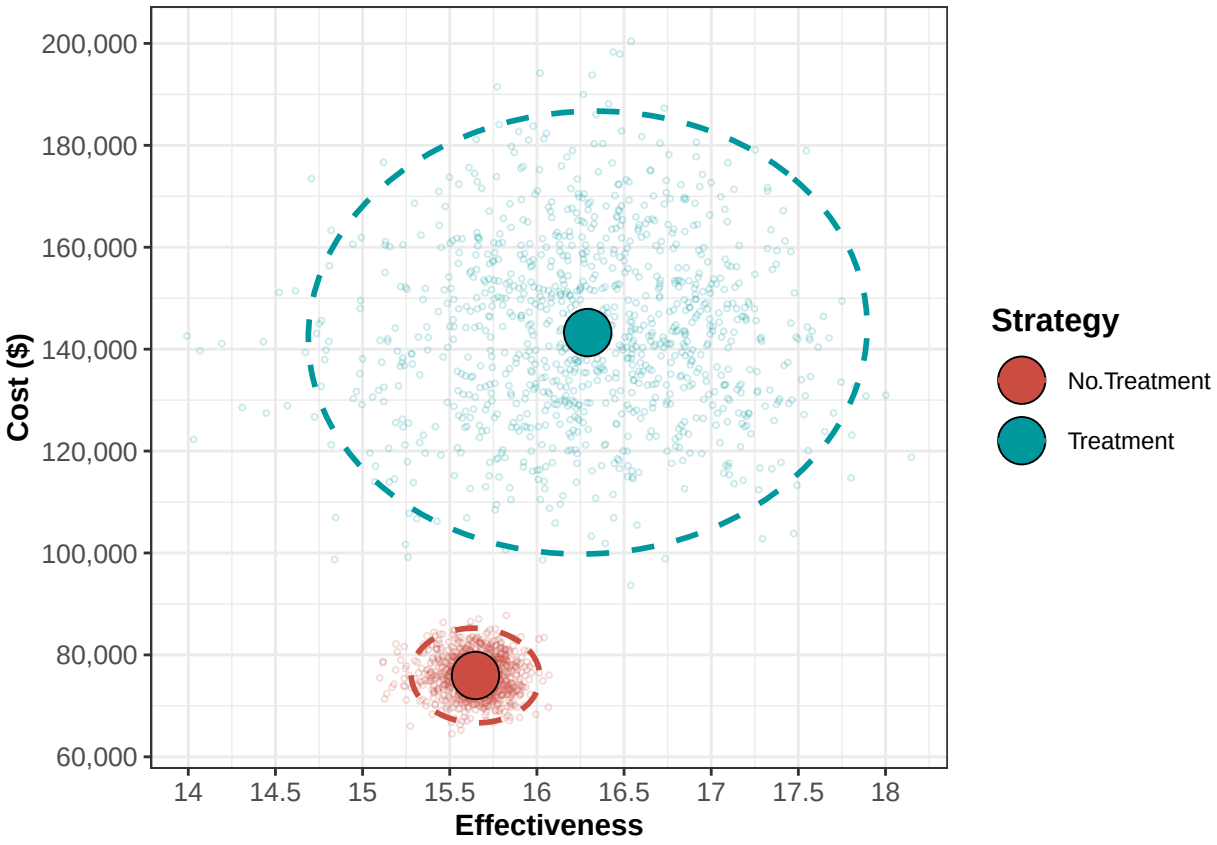
```
load(file = "markov_sick-sicker_PSA_dataset.RData")
```

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

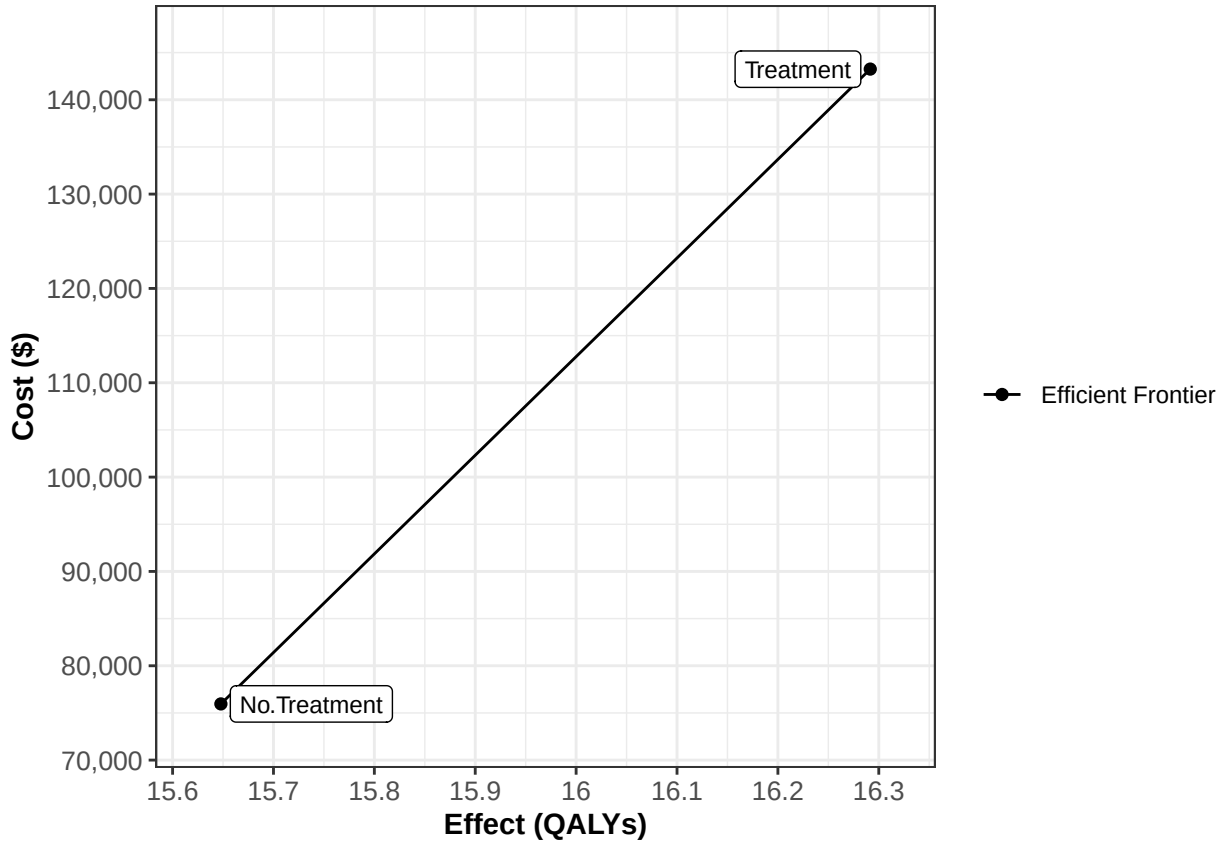
```
# Compute expected costs and effects for each strategy from the PSA
df_out_ce_psa <- summary(l_psa)

# 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
```

```
##       Strategy      Cost   Effect Inc_Cost Inc_Effect     ICER Status
## 1 No.Treatment  75955.61 15.64787       NA         NA       NA     ND
## 2    Treatment 143251.54 16.29156 67295.93  0.6436874 104547.5     ND
```

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

09.4.1 Plot cost-effectiveness frontier

```
plot(df_cea_psa)
```

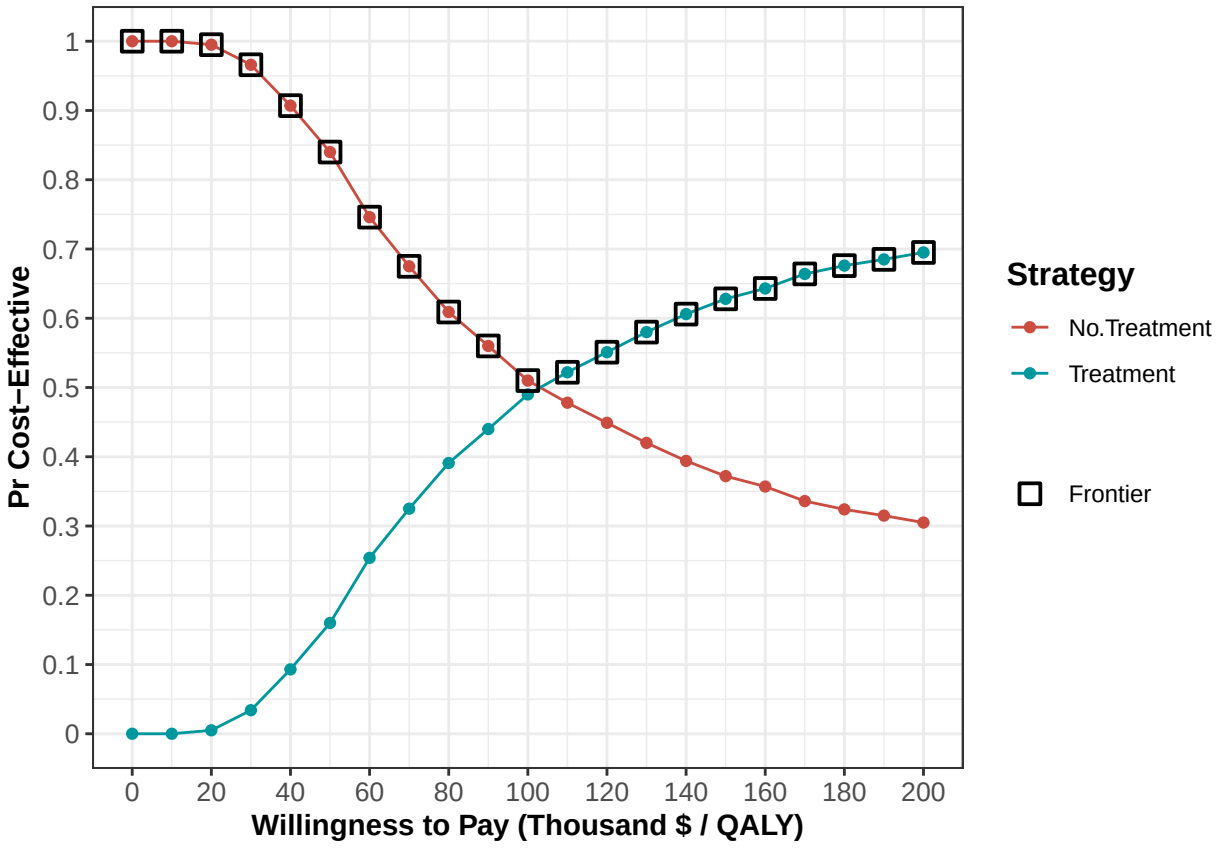


09.4.2 Cost-effectiveness acceptability curves (CEACs) and frontier (CEAF)

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

```
##   range_min range_max cost_eff_strat
## 1         0    110000   No.Treatment
## 2    110000    200000      Treatment
```

```
# CEAC & CEAF plot
plot(ceac_obj)
```



09.4.3 Expected Loss Curves (ELCs)

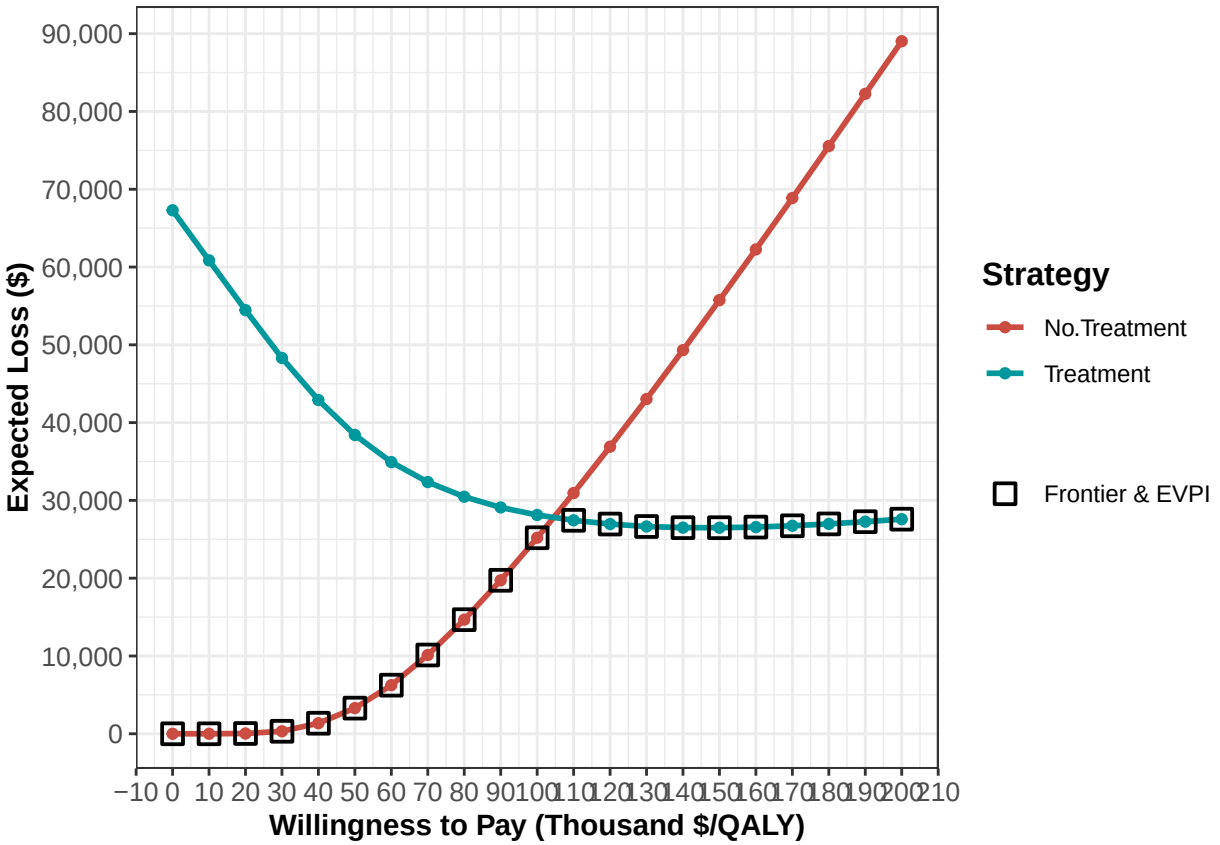
The expected loss is the the quantification of the foregone benefits when choosing a suboptimal strategy given current evidence.

```
elc_obj <- calc_exp_loss(wtp = v_wtp, psa = l_psa)
elc_obj
```

```
##       WTP     Strategy Expected_Loss On_Frontier
## 1       0 No.Treatment       0.00000        TRUE
## 2       0    Treatment   67295.92714       FALSE
## 3   10000 No.Treatment       0.00000        TRUE
## 4   10000    Treatment   60859.05317       FALSE
## 5   20000 No.Treatment      35.95817        TRUE
## 6   20000    Treatment   54458.13738       FALSE
## 7   30000 No.Treatment     322.38988        TRUE
## 8   30000    Treatment   48307.69512       FALSE
## 9   40000 No.Treatment    1357.10595        TRUE
## 10  40000    Treatment   42905.53722       FALSE
## 11  50000 No.Treatment    3297.41004        TRUE
## 12  50000    Treatment   38408.96734       FALSE
## 13  60000 No.Treatment    6248.02097        TRUE
## 14  60000    Treatment   34922.70430       FALSE
## 15  70000 No.Treatment   10123.60585        TRUE
```

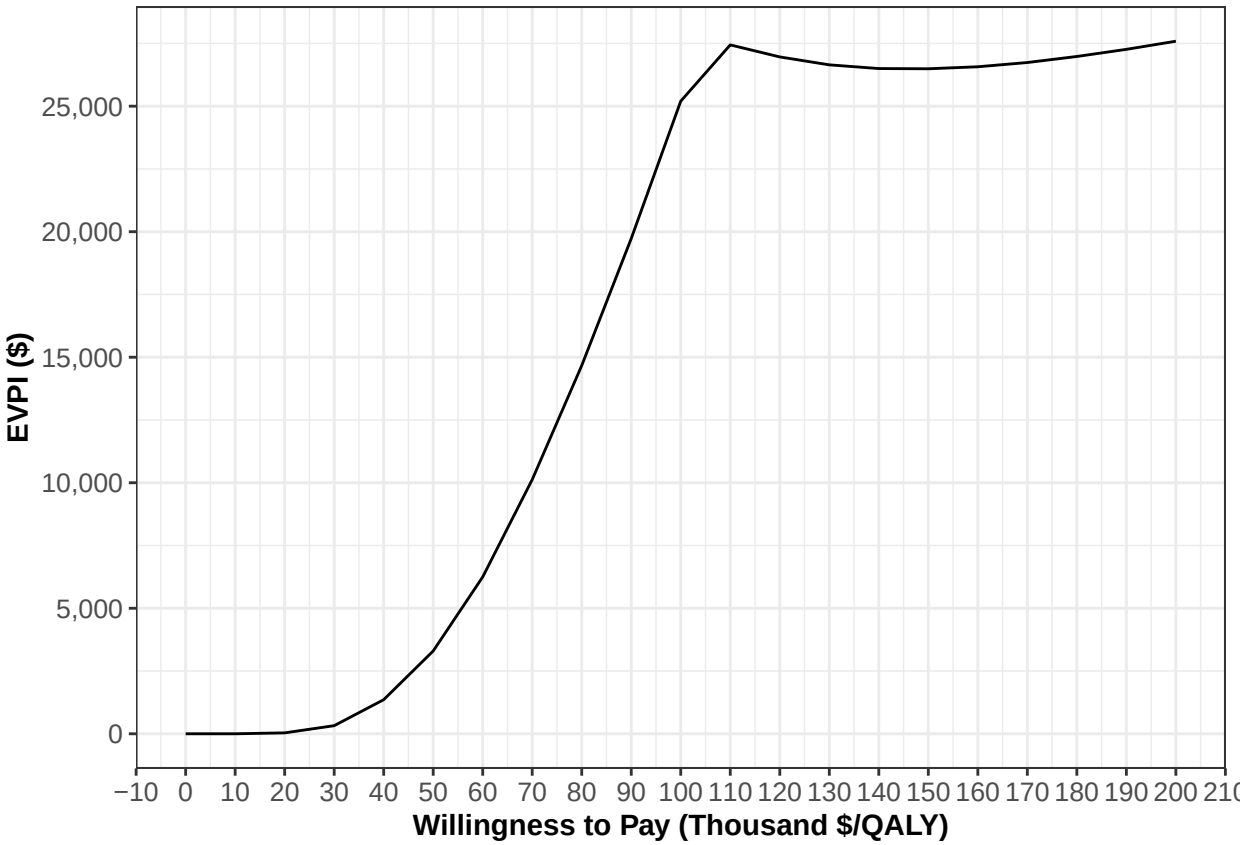
```
## 16  70000    Treatment   32361.41521       FALSE
## 17  80000 No.Treatment   14668.48694        TRUE
## 18  80000    Treatment   30469.42234       FALSE
## 19  90000 No.Treatment   19729.40367        TRUE
## 20  90000    Treatment   29093.46510       FALSE
## 21 100000 No.Treatment   25199.14250        TRUE
## 22 100000    Treatment   28126.32997       FALSE
## 23 110000 No.Treatment   30950.62051       FALSE
## 24 110000    Treatment   27440.93400        TRUE
## 25 120000 No.Treatment   36908.01787       FALSE
## 26 120000    Treatment   26961.45740        TRUE
## 27 130000 No.Treatment   43029.50480       FALSE
## 28 130000    Treatment   26646.07036        TRUE
## 29 140000 No.Treatment   49318.83174       FALSE
## 30 140000    Treatment   26498.52333        TRUE
## 31 150000 No.Treatment   55747.56036       FALSE
## 32 150000    Treatment   26490.37798        TRUE
## 33 160000 No.Treatment   62263.42425       FALSE
## 34 160000    Treatment   26569.36791        TRUE
## 35 170000 No.Treatment   68869.16074       FALSE
## 36 170000    Treatment   26738.23043        TRUE
## 37 180000 No.Treatment   75547.14077       FALSE
## 38 180000    Treatment   26979.33649        TRUE
## 39 190000 No.Treatment   82267.39582       FALSE
## 40 190000    Treatment   27262.71757        TRUE
## 41 200000 No.Treatment   89028.28011       FALSE
## 42 200000    Treatment   27586.72789        TRUE
```

```r
# 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")
```


25,000
20,000
15,000
10,000
5,000
0
EVPI ($)
−10 0 10 20 30 40 50 60 70 80 90 100 110 120 130 140 150 160 170 180 190 200 210
Willingness to Pay (Thousand $/QALY)