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.

```
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("dplyr", "tidyr", "reshape2", "devtools", "scales", "ellipse", "ggplot2", "lazyeval", "igraph",
# 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
```

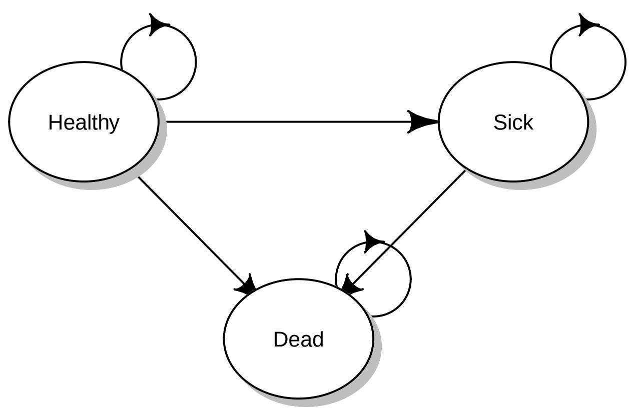
03 Model input

```
## General setup
cycle_length    <- 1                              # cycle length equal to monthly
n_cycles        <- 120                             # 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 factors

annual_rate <- 0.3
d_c <- 0.3  # annual discount rate for costs
d_e <- 0.3  # annual discount rate for QALYs


### Strategies
v_names_str     <- c("Treatment A",
                     "Treatment B")
n_str           <- length(v_names_str)            # number of strategies

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

### Transition probabilities


### State rewards
#### Costs
c_H       <- 400   # cost of one cycle in healthy state
c_S       <- 1000  # cost of one cycle in sick state
c_D       <- 0     # cost of one cycle in dead state
c_trtA    <- 800   # cost of treatment A (per cycle) in healthy state
c_trtB    <- 1500  # cost of treatment B (per cycle) in healthy state
```

```r
#### Utilities
u_H       <- 1     # one year utility when healthy
u_S       <- 0.5   # one year utility when sick
u_D       <- 0     # one year utility when dead


### Discount weight for costs and effects
v_dwc     <- 1 / ((1 + (((1 + d_c)^(1/12) - 1) * cycle_length)) ^ (0:n_cycles))
v_dwe     <- 1 / ((1 + (((1 + d_e)^(1/12) - 1) * cycle_length)) ^ (0:n_cycles))
```

04 Construct state-transition models

```r
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 <- c("Healthy" = 1, "Sick" = 0, "Dead" = 0)
v_m_init
```

```
## Healthy    Sick    Dead
##       1       0       0
```

04.2 Initialize cohort traces

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

## Initialize cohort traces for treatments A and B
# Structure and initial states are the same as for SoC
m_M_trtA <- m_M_trtB
```

04.3.1 Create transition probability arrays

```r
p_trtA<- read.csv("M.trA.csv",header = TRUE, nrows = 121)
p_trtB<- read.csv("M.trB.csv",header = TRUE, nrows = 121)
m_M_trtA<- as.matrix(p_trtA)
m_M_trtB<- as.matrix(p_trtB)
rownames(m_M_trtA) <- v_names_cycles
colnames(m_M_trtA) <- v_names_states
rownames(m_M_trtB) <- v_names_cycles
colnames(m_M_trtB) <- v_names_states
## Store the cohort traces in a list
l_m_M <- list(A   =  m_M_trtA,
              B   =  m_M_trtB)
names(l_m_M) <- v_names_str
```

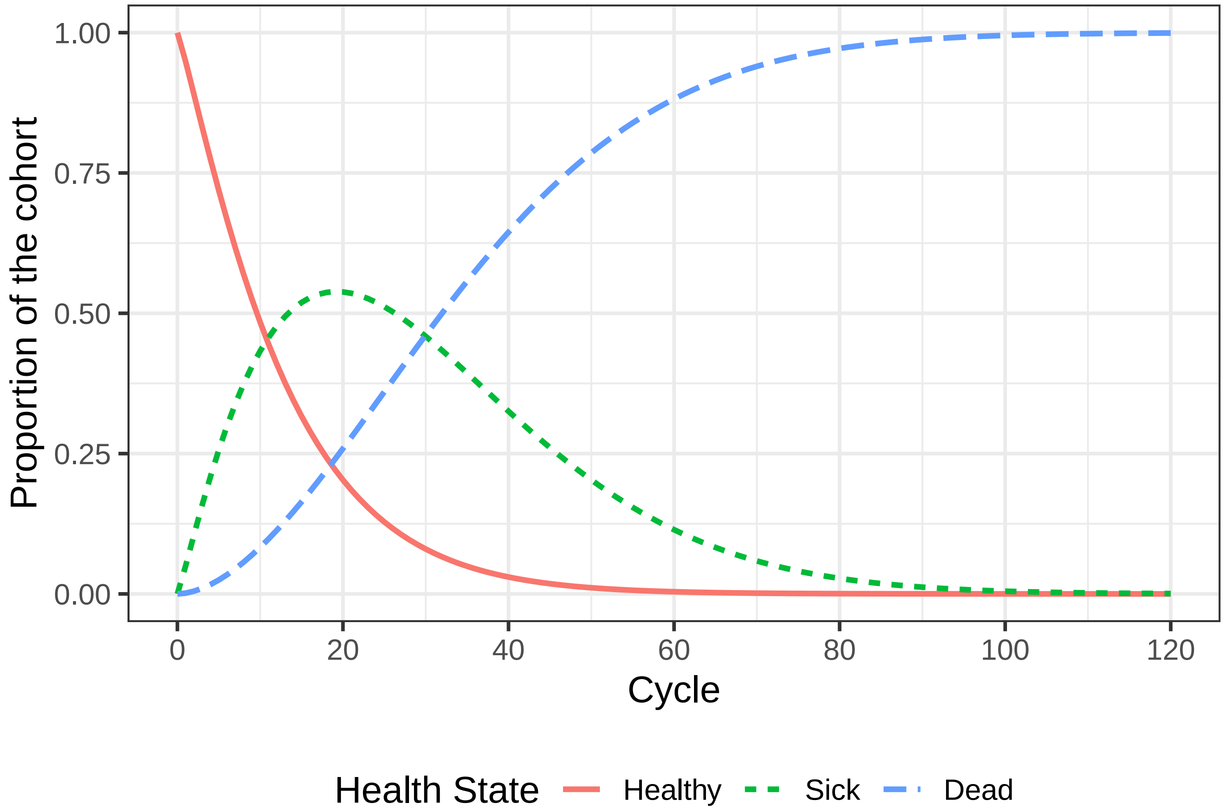
04.3 Create transition probability arrays

05 Run Markov model

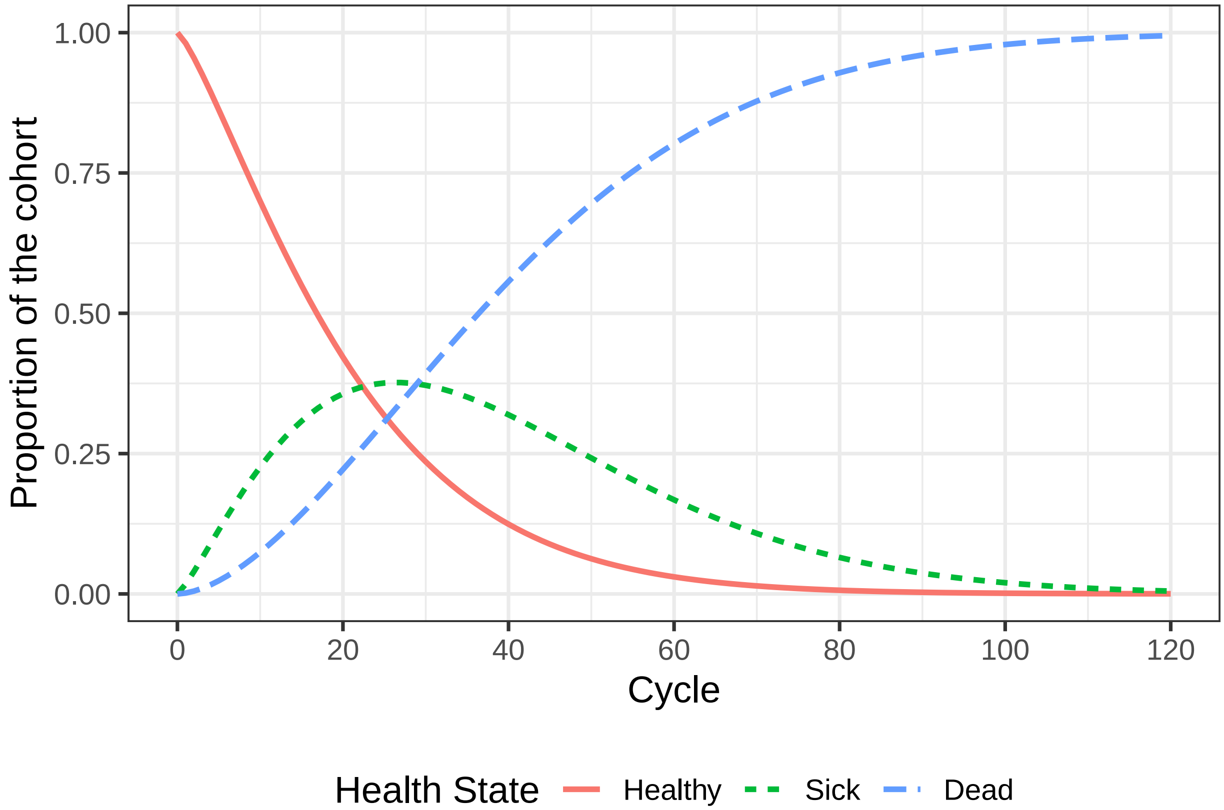
06 Plot Outputs

06.1 Plot the cohort trace for strategies SoC

```r
plot_trace(m_M_trtA)
```



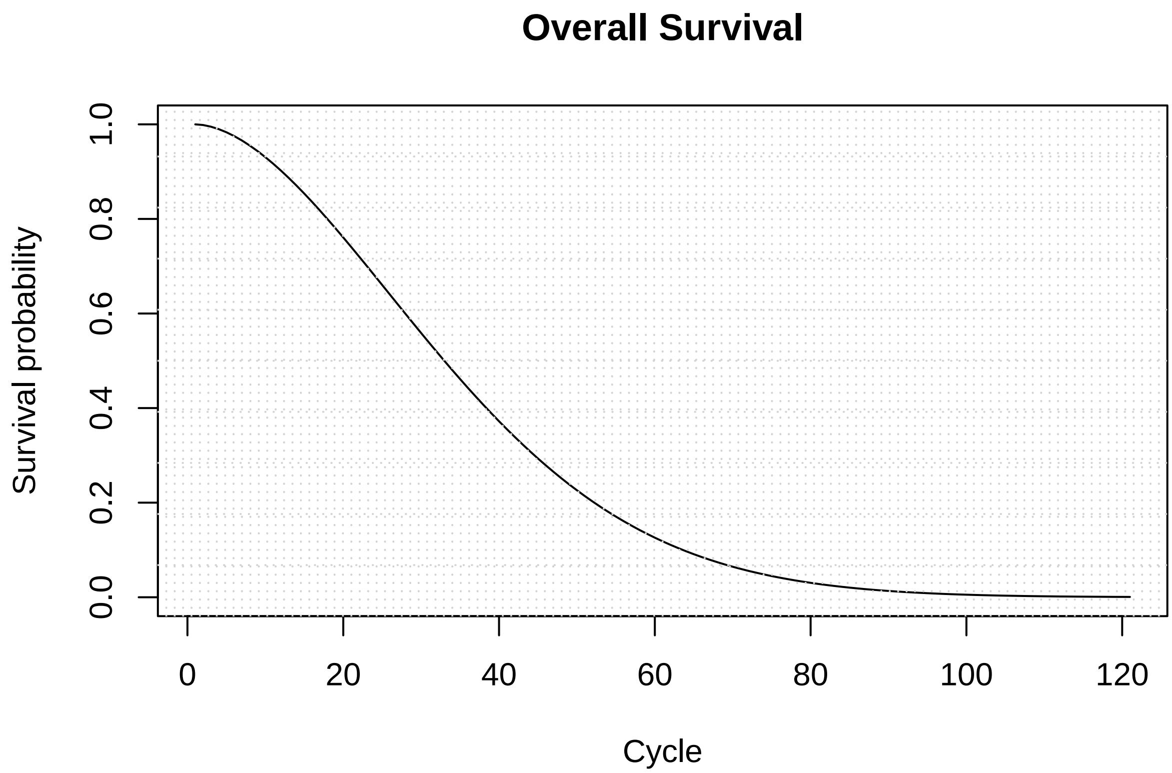

```
plot_trace( m_M_trtB)
```



06.2 Overall Survival (OS)

Print the overall survival for the Standard of Care

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

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

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

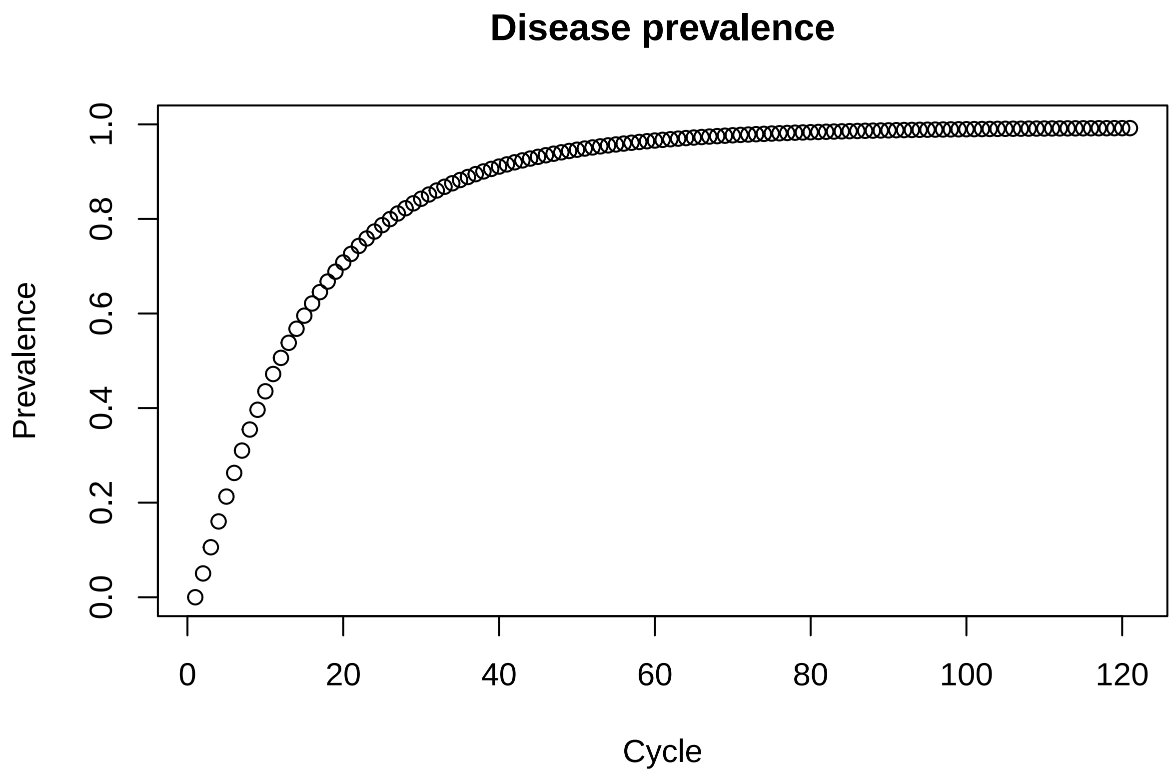


06.2.1 Life Expectancy (LE)

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

06.2.2 Disease prevalence

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



07 State Rewards

```r
## Scale by the cycle length
# Vector of state utilities under treatment A
v_u_trtA   <- c(H  = u_H/12,
                S  = u_S/12,
                D  = u_D/12) * cycle_length
# Vector of state costs under treatment A
v_c_trtA   <- c(H  = c_H + c_trtA,
                S  = c_S,
                D  = c_D) * cycle_length
# Vector of state utilities under treatment B
v_u_trtB   <- c(H  = u_H/12,
                S  = u_S/12,
                D  = u_D/12) * cycle_length
# Vector of state costs under treatment B
v_c_trtB   <- c(H  = c_H + c_trtB,
                S  = c_S,
                D  = c_D) * cycle_length

## Store state rewards
# Store the vectors of state utilities for each strategy in a list
l_u   <- list(A  = v_u_trtA,
              B  = v_u_trtB)
# Store the vectors of state cost for each strategy in a list
```

```r
l_c   <- list(A  = v_c_trtA,
              B  = v_c_trtB)

# 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) {
  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

  ### Expected QALYs and costs per cycle
  ## Vector of QALYs and Costs
  # Apply state rewards
  v_qaly_str <- l_m_M[[i]] %*% v_u_str # sum the utilities of all states for each cycle
  v_cost_str <- l_m_M[[i]] %*% v_c_str # sum the costs of all states for each cycle

  ### Discounted total expected QALYs and Costs per strategy and apply within-cycle correction if appli
  # 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)

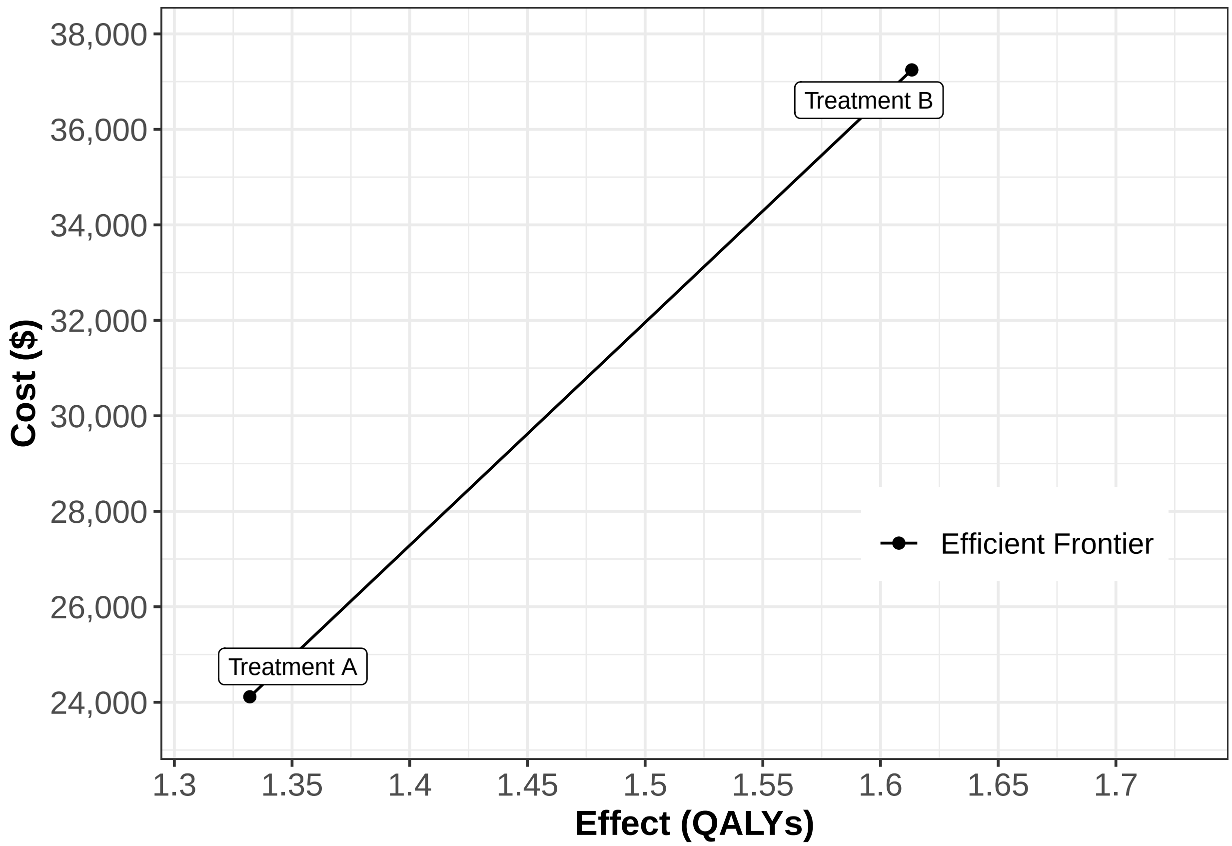
```r
## Incremental cost-effectiveness ratios (ICERs)
df_cea <- calculate_icers(cost       = v_tot_cost,
                          effect     = v_tot_qaly,
                          strategies = v_names_str)
df_cea
```

```
##                 Strategy     Cost   Effect Inc_Cost Inc_Effect     ICER Status
## Treatment A Treatment A 24114.83 1.332060       NA         NA       NA     ND
## Treatment B Treatment B 37244.76 1.613289 13129.93  0.2812287 46687.74     ND
```

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

```
##                 Strategy Costs ($) QALYs Incremental Costs ($) Incremental QALYs
## Treatment A Treatment A    24,115  1.33                  <NA>                NA
## Treatment B Treatment B    37,245  1.61                13,130              0.28
##             ICER ($/QALY) Status
## Treatment A          <NA>     ND
## Treatment B        46,688     ND
```

```r
## CEA frontier
plot(df_cea, label = "all", txtsize = 14) +
  expand_limits(x = max(table_cea$QALYs) + 0.1) +
  theme(legend.position = c(0.8, 0.3))
```



10 Deterministic Sensitivity Analysis (DSA)

```r
## Load model, CEA and PSA functions
source('Functions_markov_3state_time1.R')
```

10.1 Model input for SA

```r
l_params_all <- list(
  # Transition probabilities
m_M_trtA,
m_M_trtB,
 ## State rewards
 # Costs
  c_H       = 400,   # cost of one cycle in healthy state
  c_S       = 1000,  # cost of one cycle in sick state
  c_D       = 0,     # cost of one cycle in dead state
  c_trtA    = 800,   # cost of treatment A (per cycle) in healthy state
  c_trtB    = 1500,  # cost of treatment B (per cycle) in healthy state
  # Utilities
```

```
  u_H       = 1,     # utility when healthy
  u_S       = 0.5,   # utility when sick
  u_D       = 0,     # utility when dead
  # Discount rates
  d_e       = 0.03,  # discount rate per cycle equal discount of costs and QALYs by 3%
  d_c       = 0.03,  # discount rate per cycle equal discount of costs and QALYs by 3%
  # Time horizon
  n_cycles  = 120     # simulation time horizon (number of cycles)
)
```

Test model functions

Two functions were defined in the `Functions_markov_3state_time.R` file. The first was the `decision_model()` function which takes in model parameters and outputs the Markov trace for all three strategies. The second is the `calculate_ce_out()` function which calls `decision_model()` and uses the resulting Markov trace to compute the total costs, QALYs, and net monetary benefit (NMB) for each strategy, returning a data frame of costs and QALYs.

```
# Try the decision_model() function
l_trace <- decision_model(l_params_all)
l_trace$m_M_trtA[1:5,]
```

```
## NULL
```

```
l_trace$a_P_trtA[,,1:3]
```

```
## NULL
```

```
# Try the calculate_ce_out() function
df_ce <- calculate_ce_out(l_params_all)
df_ce
```

```
##                Strategy     Cost   Effect      NMB
## Treatment A Treatment A 24114.83 1.332060 109091.2
## Treatment B Treatment B 37244.76 1.613289 124084.1
```

```
# Get strategies names (will be used to label plots)
v_names_str <- df_ce$Strategy
n_str <- length(v_names_str)
```

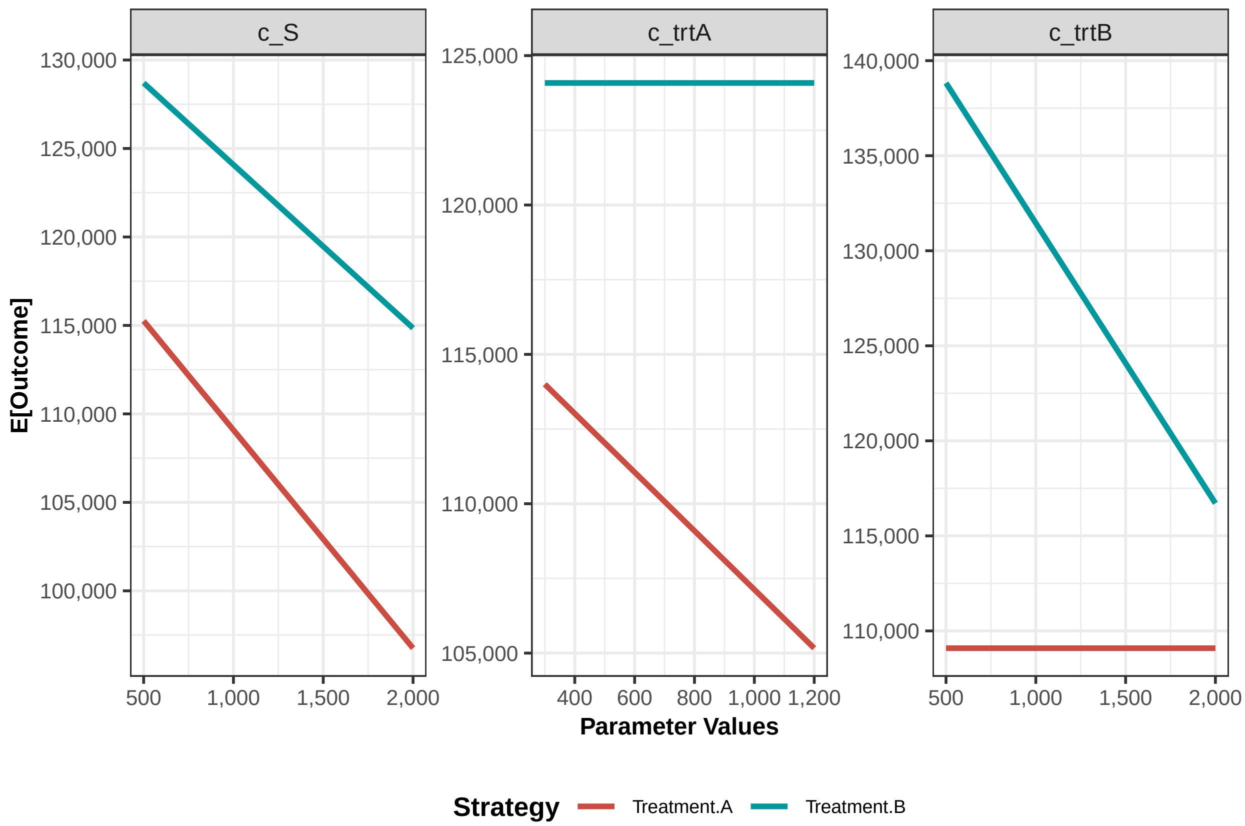
10.2 One-way sensitivity analysis (OWSA)

```
options(scipen = 999) # disabling scientific notation in R
# dataframe containing all parameters, their base case values, and the min and
# max values of the parameters of interest
df_params_owsa <- data.frame(pars = c("c_trtA", "c_trtB", "c_S"),
                             min  = c(300 , 500,   500),  # min parameter values
                             max  = c(1200, 2000, 2000)  # 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            = 100000)              # extra argument to pass to FUN
```

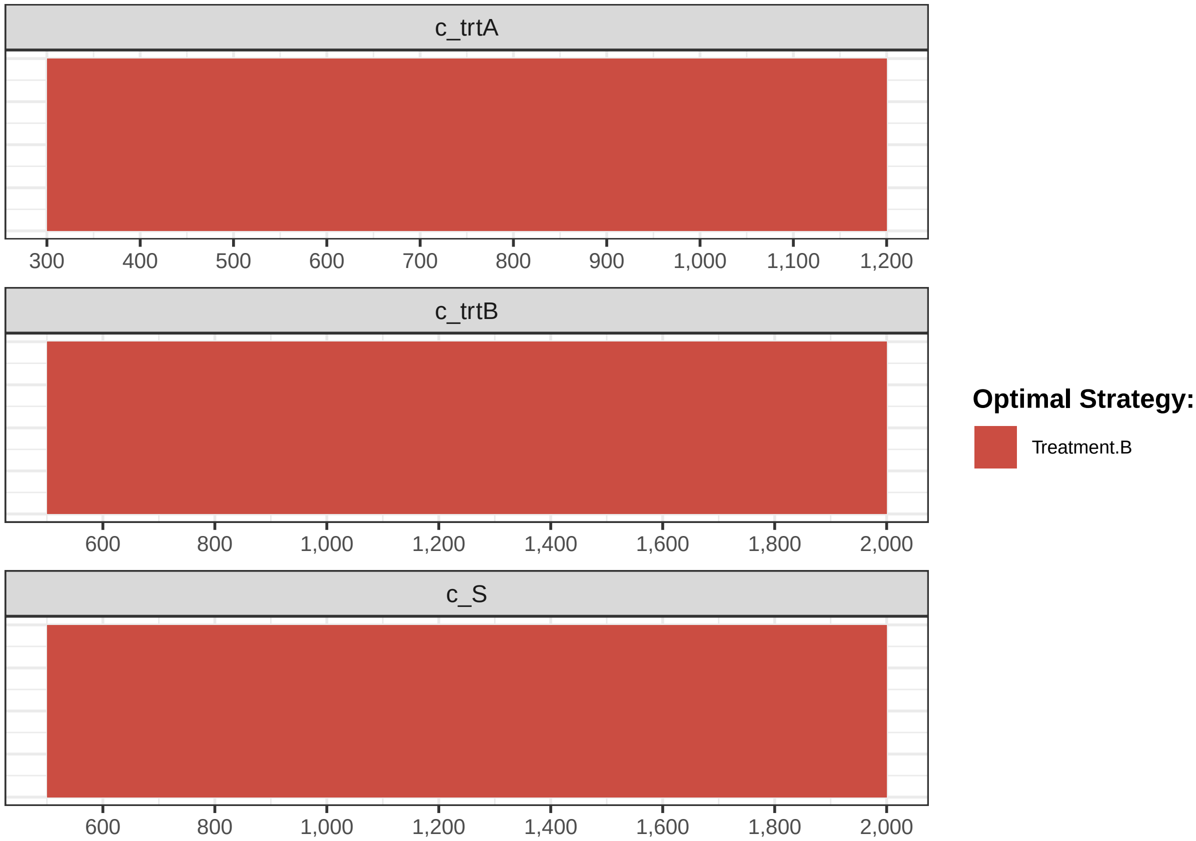
```
##   |                                                                                |
```

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



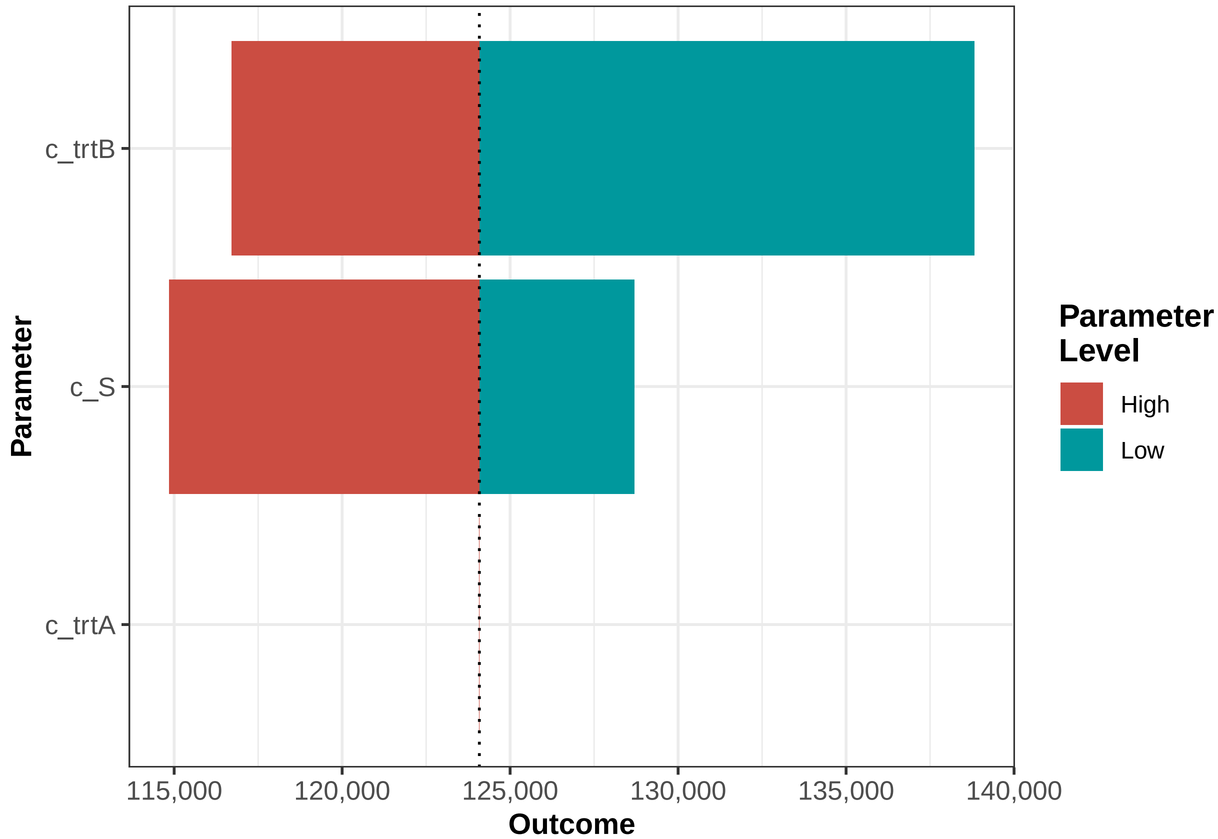
10.2.1 Optimal strategy with OWSA

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



10.2.2 Tornado plot

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

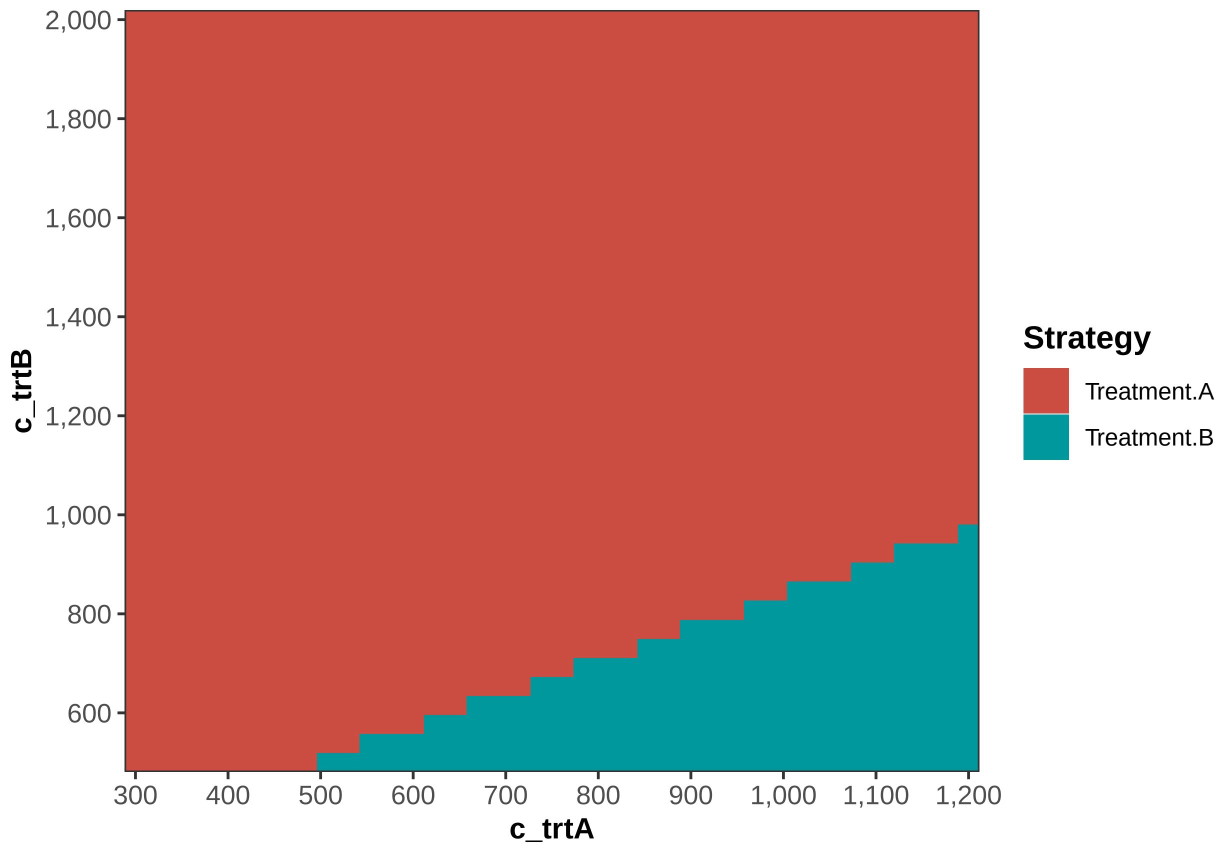


10.3 Two-way sensitivity analysis (TWSA)

```
# 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_trtA", "c_trtB"),
                             min  = c(300, 500),  # min parameter values
                             max  = c(1200, 2000) # 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           = 5000)              # 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
## Treatment A Treatment A 24114.83 1.332060 109091.2
## Treatment B Treatment B 37244.76 1.613289 124084.1
```

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

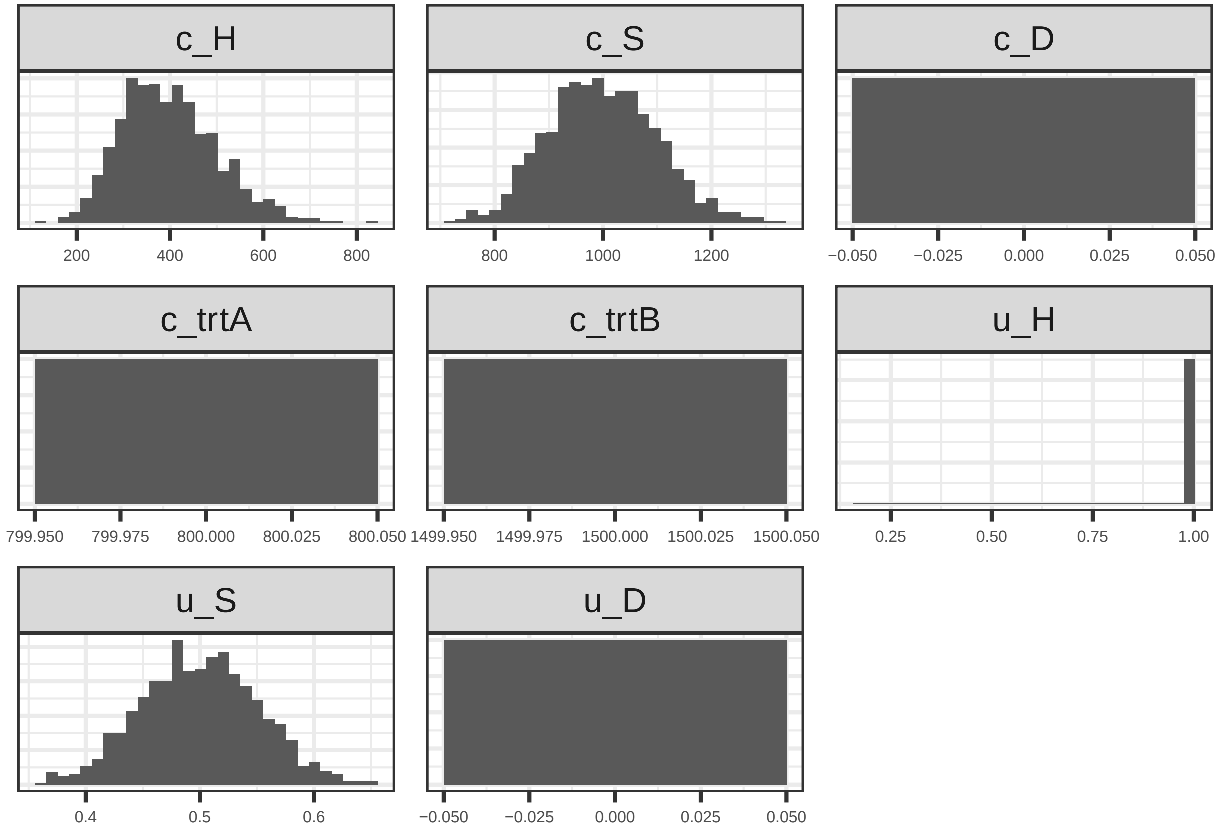
```
##         c_H       c_S c_D c_trtA c_trtB u_H       u_S u_D
## 1  537.2865  970.5874   0    800   1500   1 0.3847906   0
## 2  392.0464  917.3663   0    800   1500   1 0.4486215   0
## 3  304.0767  888.4998   0    800   1500   1 0.4974254   0
## 4  292.3548 1279.1368   0    800   1500   1 0.4678000   0
## 5  363.1699  975.7369   0    800   1500   1 0.5454105   0
## 6  376.5410 1032.3132   0    800   1500   1 0.4932370   0
## 7  324.4721  980.3957   0    800   1500   1 0.4933852   0
```

```
## 8  565.6859 1085.6563   0    800   1500   1 0.4774074   0
## 9  252.7441 1102.2173   0    800   1500   1 0.4283304   0
## 10 516.8618  905.9940   0    800   1500   1 0.5489382   0
```

```
## 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)
```

```
##        c_H      c_S c_D c_trtA c_trtB u_H       u_S u_D
## 1 537.2865 1125.652   0    800   1500   1 0.5093511   0
## 2 392.0464 1104.588   0    800   1500   1 0.5025504   0
## 3 304.0767 1097.612   0    800   1500   1 0.4180469   0
## 4 292.3548 1133.857   0    800   1500   1 0.5667343   0
## 5 363.1699 1004.396   0    800   1500   1 0.4877285   0
## 6 376.5410 1087.334   0    800   1500   1 0.4692128   0
```

```
### 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))
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 % done
```

```
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)))
```

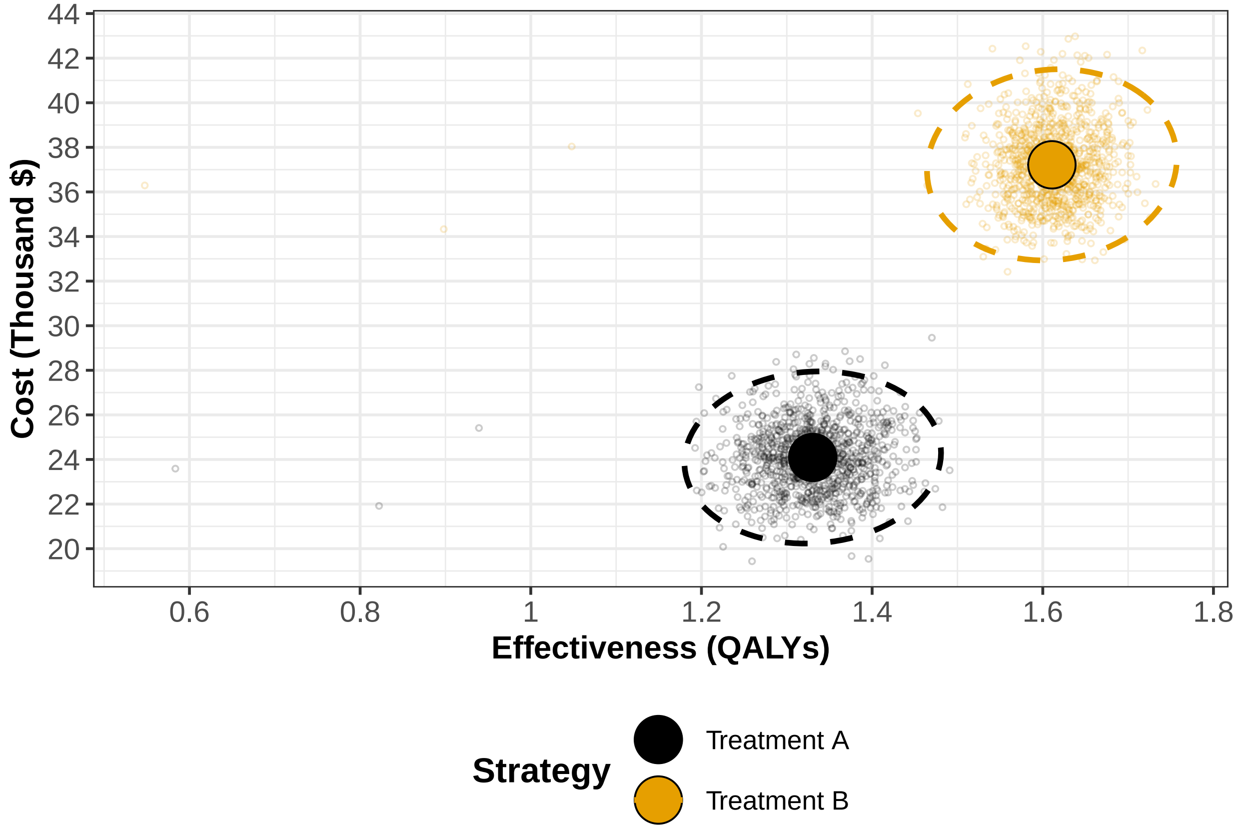
```
## [1] "PSA with 1,000 simulations run in series in 0.85 secs"
```

11.3 Visualize PSA results for CEA

```
### 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, 100000, by = 5000)
```

11.3.1 Cost-Effectiveness Scatter plot

```
### 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
```



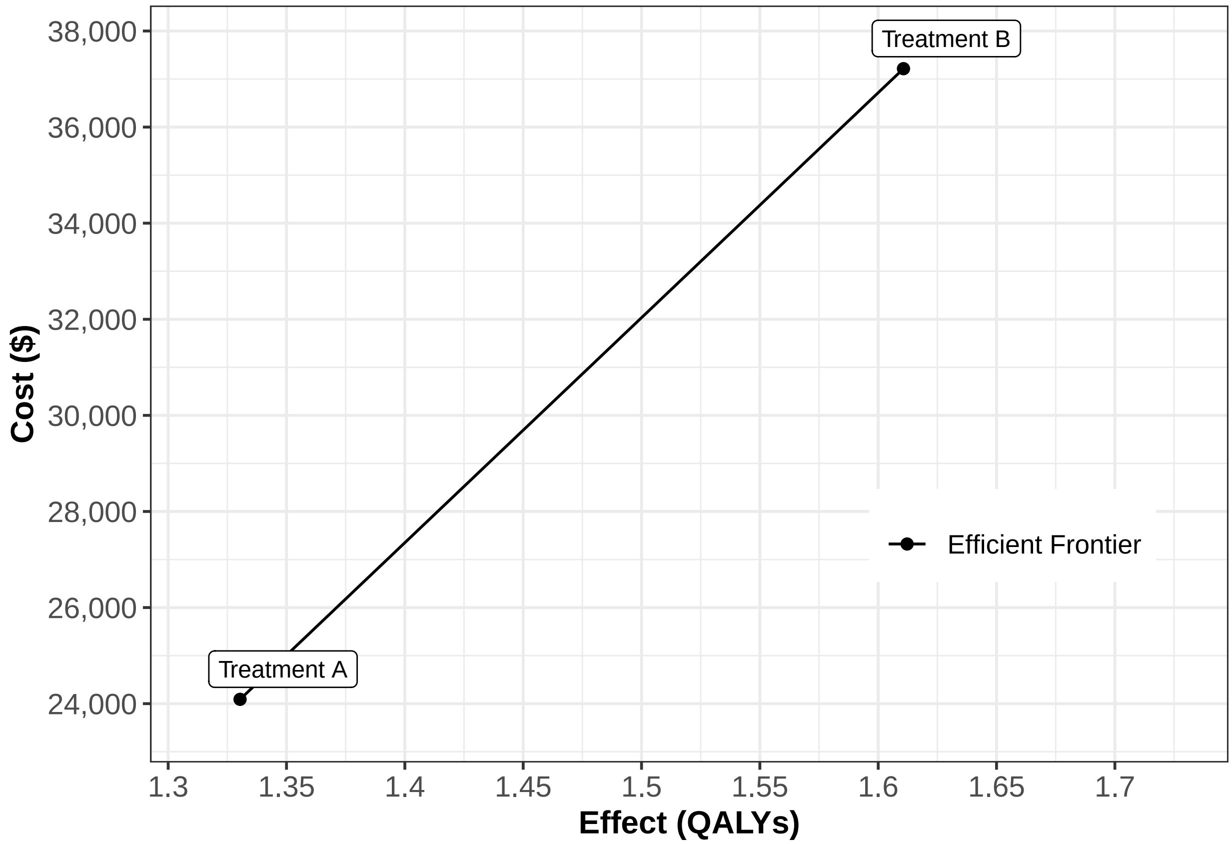
11.3.2 Incremental cost-effectiveness ratios (ICERs) with probabilistic output

```
### 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 Treatment A 24090.03 1.330379       NA         NA       NA     ND
## 2 Treatment B 37216.07 1.610679 13126.03  0.2803002 46828.49     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))
```

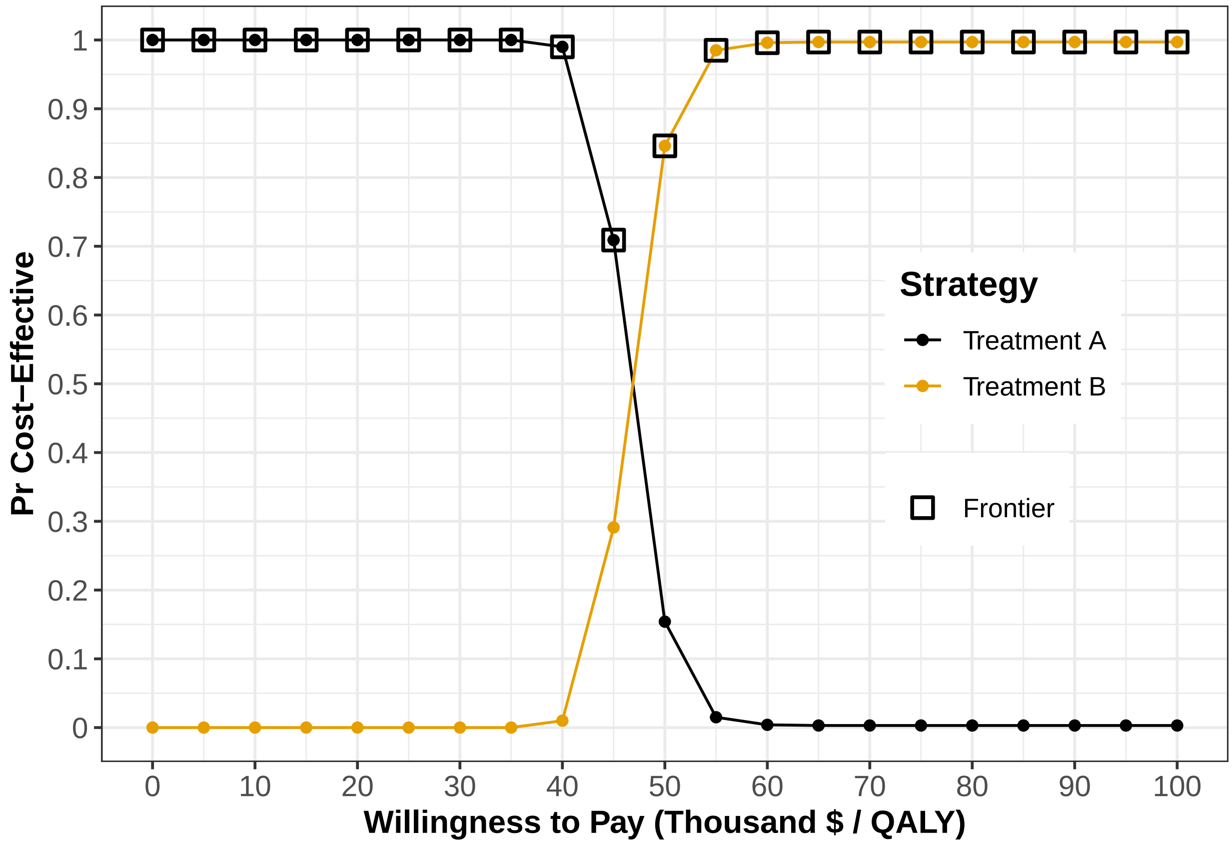


11.3.4 Cost-effectiveness acceptability curves (CEACs) and frontier (CEAF)

```
### 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     50000    Treatment A
## 2     50000    100000    Treatment B
```

```
# 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
```



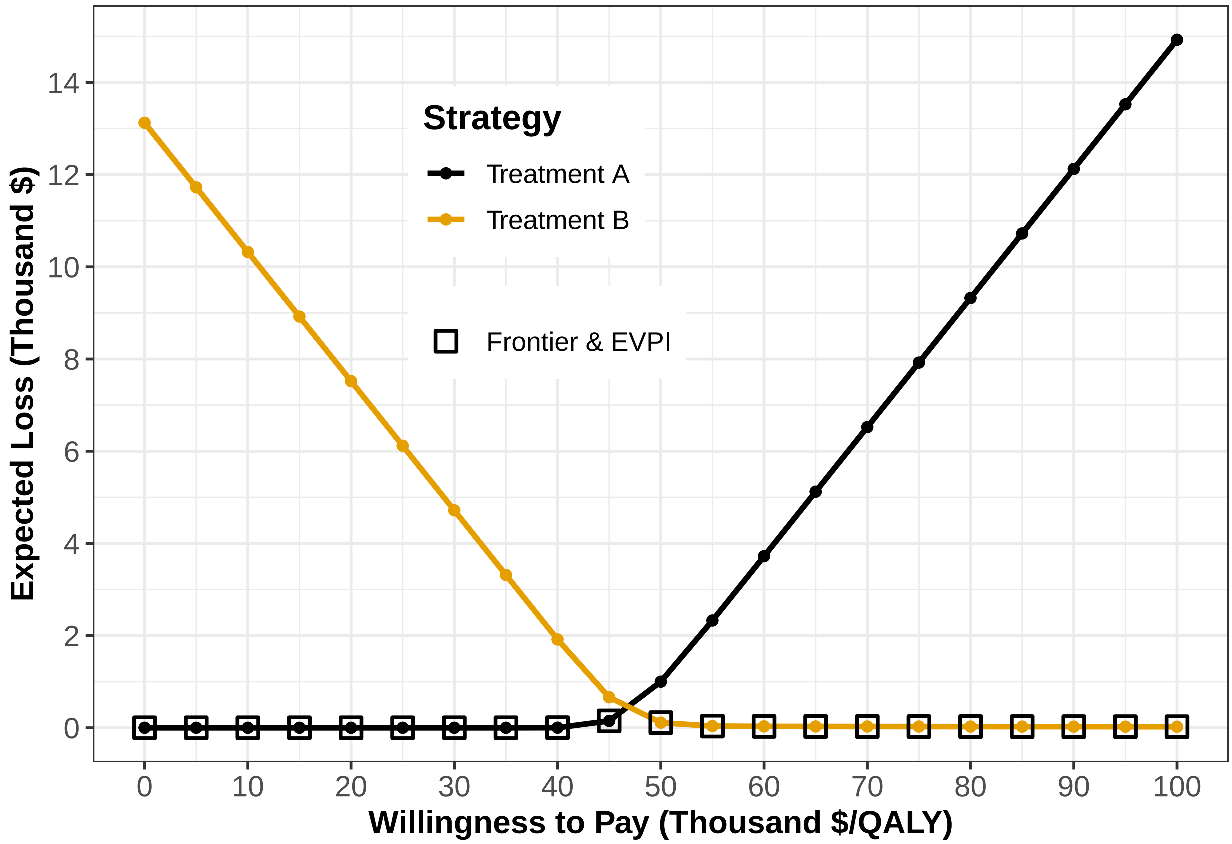
11.3.5 Expected Loss Curves (ELCs)

```
### 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 Treatment A      0.000000        TRUE
## 2        0 Treatment B  13126.032833       FALSE
## 3     5000 Treatment A      0.000000        TRUE
## 4     5000 Treatment B  11724.532073       FALSE
## 5    10000 Treatment A      0.000000        TRUE
## 6    10000 Treatment B  10323.031313       FALSE
## 7    15000 Treatment A      0.000000        TRUE
## 8    15000 Treatment B   8921.530552       FALSE
## 9    20000 Treatment A      0.000000        TRUE
## 10   20000 Treatment B   7520.029792       FALSE
## 11   25000 Treatment A      0.000000        TRUE
## 12   25000 Treatment B   6118.529032       FALSE
## 13   30000 Treatment A      0.000000        TRUE
## 14   30000 Treatment B   4717.028271       FALSE
## 15   35000 Treatment A      0.000000        TRUE
## 16   35000 Treatment B   3315.527511       FALSE
## 17   40000 Treatment A      1.834479        TRUE
## 18   40000 Treatment B   1915.861230       FALSE
## 19   45000 Treatment A    148.875620        TRUE
```

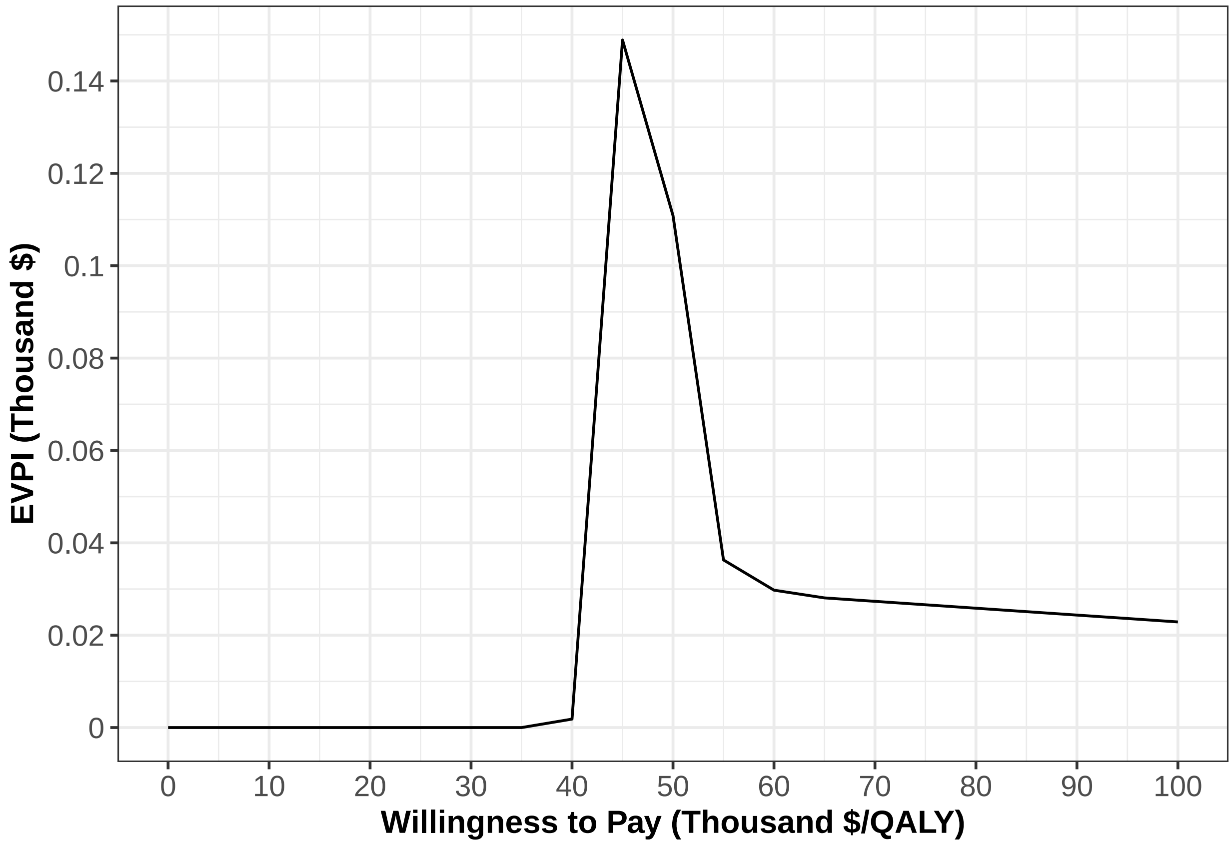
```
## 20  45000 Treatment B    661.401610       FALSE
## 21  50000 Treatment A    999.819901       FALSE
## 22  50000 Treatment B    110.845131        TRUE
## 23  55000 Treatment A   2326.770265       FALSE
## 24  55000 Treatment B     36.294734        TRUE
## 25  60000 Treatment A   3721.726427       FALSE
## 26  60000 Treatment B     29.750137        TRUE
## 27  65000 Treatment A   5121.551110       FALSE
## 28  65000 Treatment B     28.074059        TRUE
## 29  70000 Treatment A   6522.308375       FALSE
## 30  70000 Treatment B     27.330564        TRUE
## 31  75000 Treatment A   7923.065641       FALSE
## 32  75000 Treatment B     26.587069        TRUE
## 33  80000 Treatment A   9323.822907       FALSE
## 34  80000 Treatment B     25.843575        TRUE
## 35  85000 Treatment A  10724.580172       FALSE
## 36  85000 Treatment B     25.100080        TRUE
## 37  90000 Treatment A  12125.337438       FALSE
## 38  90000 Treatment B     24.356585        TRUE
## 39  95000 Treatment A  13526.094704       FALSE
## 40  95000 Treatment B     23.613091        TRUE
## 41 100000 Treatment A  14926.851969       FALSE
## 42 100000 Treatment B     22.869596        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/10.1177/0272989X221103163
- Alarid-Escudero F, Krijkamp EM, Enns EA, Yang A, Hunink MGM Pechlivanoglou P, Jalal H. An Introductory Tutorial on Cohort State-Transition Models in R Using a Cost-Effectiveness Analysis Example. Medical Decision Making, 2022 (Online First):1-18. https://doi.org/10.1177/0272989X221103163
- 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