# PSA: Three-strategy decision tree in R - HVE

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. Division of Public Administration, Center for Research and Teaching in Economics (CIDE), 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. 2020 Online first. https://doi.org/10.1177/0272989X19893973
- Alarid-Escudero, F., Krijkamp, E. M., Enns, E. A., Hunink, M. G. M., Pechlivanoglou, P., & Jalal, H. (2020). Cohort state-transition models in R: From conceptualization to implementation. arXiv:2001.07824v1, 1–31. http://arxiv.org/abs/2001.07824
- Alarid-Escudero, F., Enns, E. A., Kuntz, K. M., Michaud, T. L., & Jalal, H. (2019). "Time Traveling Is Just Too Dangerous" But Some Methods Are Worth Revisiting: The Advantages of Expected Loss Curves Over Cost-Effectiveness Acceptability Curves and Frontier. Value in Health, 22(5), 611–618. https://doi.org/10.1016/j.jval.2019.02.008

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 install other packages
# 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/dectree", force = TRUE) Uncomment if there is a newer version
# install_github("annaheath/EVSI", force = TRUE) #Uncomment if there is a newer version
p_load_gh("DARTH-git/dectree")
p_load_gh("annaheath/EVSI")
```

## 02 Load functions

```
source("Functions.R")
```

## 03 Define parameter input values

```
v_names_str <- c("No Tx", "Tx All", "Biopsy") # names of strategies
n_str       <- length(v_names_str)            # number of strategies
wtp         <- 100000                         # willingness to pay threshold

# Probabilities
p_HVE         <- 0.52 # prevalence of HVE
p_HVE_comp    <- 0.71 # complications with untreated HVE
p_OVE_comp    <- 0.01 # complications with untreated OVE
p_HVE_comp_tx <- 0.36 # complications with treated HVE
p_OVE_comp_tx <- 0.20 # complications with treated OVE
p_biopsy_comp <- 0.05 # probability of complications due to biopsy

# Costs
c_VE      <- 1200  # cost of viral encephalitis care without complications
c_VE_comp <- 9000  # cost of viral encephalitis care with complications
c_tx      <- 9500  # cost of treatment
c_biopsy  <- 25000 # cost of brain biopsy

# QALYs
q_VE          <- 20    # remaining QALYs for those without VE-related complications
q_VE_comp     <- 19    # remaining QALYs for those with VE-related complications
```

```r
q_loss_biopsy <- -0.01 # one-time QALY loss due to brain biopsy

# store the parameters into a list
l_params_all <- list(p_HVE, p_HVE_comp, p_OVE_comp, p_HVE_comp_tx,
                     p_OVE_comp_tx, p_biopsy_comp,
                     c_VE, c_VE_comp, c_tx, c_biopsy,
                     q_VE, q_VE_comp, q_loss_biopsy)
# store the names of the parameters into a vector
v_names_params <- c('p_HVE', 'p_HVE_comp', 'p_OVE_comp', 'p_HVE_comp_tx',
                    'p_OVE_comp_tx', 'p_biopsy_comp',
                    'c_VE', 'c_VE_comp',  'c_tx', 'c_biopsy',
                    'q_VE', 'q_VE_comp', 'q_loss_biopsy')
names(l_params_all) <- v_names_params
```

## 04 Create and run decision tree model

```r
decision_tree_HVE_output <- with(as.list(l_params_all), {

  # Create vector of weights for each strategy

  v_w_no_tx  <- c(  p_HVE  *    p_HVE_comp  ,  # HVE, complications
                    p_HVE  * (1-p_HVE_comp) ,  # HVE, no complications
                  (1-p_HVE) *    p_OVE_comp  ,  # OVE, complications
                  (1-p_HVE) * (1-p_OVE_comp))   # OVE, no complications

  v_w_tx     <- c(  1                          ,  # On treatment
                    p_HVE  *    p_HVE_comp_tx  ,  # HVE w/tx, complications
                    p_HVE  * (1-p_HVE_comp_tx) ,  # HVE w/tx, no complications
                  (1-p_HVE) *    p_OVE_comp_tx  ,  # OVE w/tx, complications
                  (1-p_HVE) * (1-p_OVE_comp_tx))   # OVE w/tx, no complications

  v_w_biopsy <- c(1                                    , # Undergo biopsy
                  p_biopsy_comp                        , # biopsy complications
                 # no biopsy comp., HVE w/tx,  complications
                 (1-p_biopsy_comp) *    p_HVE  *    p_HVE_comp_tx ,
                 # no biopsy comp., HVE w/tx, no complications
                 (1-p_biopsy_comp) *    p_HVE  * (1-p_HVE_comp_tx),
                 # no biopsy comp., OVE, complications
                 (1-p_biopsy_comp) * (1-p_HVE) *    p_OVE_comp    ,
                 # no biopsy comp., OVE, no complications
                 (1-p_biopsy_comp) * (1-p_HVE) * (1-p_OVE_comp))

  # Create vector of outcomes (QALYs) for each strategy

  v_qaly_no_tx  <- c(q_VE_comp, # HVE, complications
                     q_VE     , # HVE, no complications
                     q_VE_comp, # OVE, complications
                     q_VE)      # OVE, no complications

  v_qaly_tx     <- c(0         , # treatment does not directly add any QALYs
                     q_VE_comp , # HVE, complications
```

```
                    q_VE      , # HVE, no complications
                    q_VE_comp , # OVE, complications
                    q_VE)       # OVE, no complications


v_qaly_biopsy <- c(q_loss_biopsy, # loss due to biopsy
                    q_VE_comp    , # biopsy complications
                    q_VE_comp    , # no biopsy comp., HVE w/tx, complications
                    q_VE         , # no biopsy comp., HVE w/tx, no complications
                    q_VE_comp    , # no biopsy comp., OVE, complications
                    q_VE)          # no biopsy comp., OVE, no complications

# Create vector of costs for each strategy

v_cost_no_tx  <- c(c_VE_comp , # HVE, complications
                    c_VE      , # HVE, no complications
                    c_VE_comp , # OVE, complications
                    c_VE)       # OVE, no complications

v_cost_tx     <- c(c_tx      , # cost of treatment
                    c_VE_comp , # HVE, complications
                    c_VE      , # HVE, no complications
                    c_VE_comp , # OVE, complications
                    c_VE)       # OVE, no complications


v_cost_biopsy <- c(c_biopsy         , # cost of biopsy procedure
                    c_VE_comp        , # biopsy complications
                    c_VE_comp + c_tx , # no biopsy comp., HVE w/tx, complications
                    c_VE + c_tx      , # no biopsy comp., HVE w/tx, no complications
                    c_VE_comp        , # no biopsy comp., OVE, complications
                    c_VE)              # no biopsy comp., OVE, no complications

# Calculate total utilities for each strategy
total_qaly_no_tx  <- v_w_no_tx  %*% v_qaly_no_tx
total_qaly_tx     <- v_w_tx     %*% v_qaly_tx
total_qaly_biopsy <- v_w_biopsy %*% v_qaly_biopsy

# Calculate total costs for each strategy
total_cost_no_tx  <- v_w_no_tx  %*% v_cost_no_tx
total_cost_tx     <- v_w_tx     %*% v_cost_tx
total_cost_biopsy <- v_w_biopsy %*% v_cost_biopsy

# vector of total QALYs
v_total_qaly <- c(total_qaly_no_tx, total_qaly_tx, total_qaly_biopsy)
# vector of total costs
v_total_cost <- c(total_cost_no_tx, total_cost_tx, total_cost_biopsy)
# calculate vector of nmb
v_nmb        <- v_total_qaly * wtp - v_total_cost

# Name outcomes
names(v_total_qaly) <- v_names_str  # names for the elements of the total QALYs vector
names(v_total_cost) <- v_names_str  # names for the elements of the total cost vector
```

```
  names(v_nmb)        <- v_names_str  # names for the elements of the nmb vector

  df_output <- data.frame(Strategy = v_names_str,
                          Cost     = v_total_cost,
                          Effect   = v_total_qaly,
                          NMB      = v_nmb)
  return(df_output)
})

# model output
decision_tree_HVE_output
```

### 04.1 Plot the decision tree

```
# branches <- read.csv(here('data','decision_tree_HVE_branches.csv'),
#                      stringsAsFactors = F, header = T)
# tree     <- create_tree(branches)
# plot_tree(tree, font.size = 5)
```

### 05 Cost-Effectiveness Analysis

```
# create the transition probability matrix for NO treatment
decision_tree_HVE_cea  <- calculate_icers(cost       = decision_tree_HVE_output$Cost,
                                          effect     = decision_tree_HVE_output$Effect,
                                          strategies = decision_tree_HVE_output$Strategy)
decision_tree_HVE_cea
```

### 05.1 Plot frontier of Decision Tree

```
plot(decision_tree_HVE_cea, effect_units = "QALYs")
```

## 06 Deterministic Sensitivity Analysis

### 06.1 List of input parameters

```
l_params_all
```

### 06.2 Load decision tree model function

```r
#### We wrapped the decision tree in a function which we called calculate_ce_out
# This function is stored in "Functions_decision_tree_HVE.R" and needes the list of parameters
source("Functions_decision_tree_HVE.R")
# Test function to see if it gives the CE results
calculate_ce_out(l_params_all)
```

### 06.3 One-way sensitivity analysis (OWSA)

```r
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_HVE", "p_biopsy_comp", "c_tx", "c_biopsy"),
                             min  = c(0.01, 0.01,  1000, 5000),  # min parameter values
                             max  = c(0.99, 0.50, 15000, 40000)  # 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      = 450000)            # extra argument to pass to FUN
```

#### 06.3.1 Plot OWSA

```r
plot(owsa_nmb, txtsize = 16, n_x_ticks = 5, n_y_ticks = 3,
     facet_scales = "free") +
     theme(legend.position = "bottom",
           axis.text.x = element_text(angle = 45, vjust = 0.5))
```

#### 06.3.2 Optimal strategy with OWSA

```r
owsa_opt_strat(owsa = owsa_nmb)
```

#### 06.3.3 Tornado plot

```r
owsa_tornado(owsa = owsa_nmb) +
  theme(axis.text.x = element_text(angle = 45, vjust = 0.5))
```

### 06.4 Two-way sensitivity analysis (TWSA)

```r
# 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("p_HVE", "c_biopsy"),
                             min  = c(0.01, 2000), # min parameter values
                             max  = c(0.99, 40000) # 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   = c("NMB"),         # output to do the twsa on
                         strategies = v_names_str,      # names of the strategies
                         n_wtp      = 200000)           # extra argument to pass to FUN
```

### 06.4.1 Plot TWSA

```r
plot(twsa_nmb) +
  ggtitle(label = "Two-way sensitivity analysis",
          subtitle = "Net monetary benefit") +
          theme(legend.position = "bottom")
```

## 07 Probabilistic Sensitivity Analysis (PSA)

```r
# Function to generate PSA input dataset
generate_psa_params <- function(n_sim = 1000, seed = 071518){
  set.seed(seed)
  # Dataframe of input parameters
  df_psa_params   <- data.frame(

    # Transition probabilities (per cycle)
    p_HVE         = rbeta(n_sim, 52, 48), # prevalence of HVE
    p_HVE_comp    = rbeta(n_sim, 71, 29), # complications with untreated HVE
    p_OVE_comp    = rbeta(n_sim,  1, 99), # complications with untreated OVE
    p_HVE_comp_tx = rbeta(n_sim, 36, 64), # complications with treated HVE
    p_OVE_comp_tx = rbeta(n_sim, 20, 80), # complications with treated OVE
    p_biopsy_comp = rbeta(n_sim,  1, 19), # probability of complications due to biopsy

    # Costs
    c_VE      = rgamma(n_sim, shape = 36.0, scale = 33.33), # cost of remaining one cycle in state H
    c_VE_comp = rgamma(n_sim, shape = 81.0, scale = 111.1), # cost of remaining one cycle in state S1
    c_tx      = rgamma(n_sim, shape = 74.6, scale = 127.4), # cost of remaining one cycle in state S2
    c_biopsy  = rgamma(n_sim, shape = 25.0, scale = 1000) , # cost of treatment (per cycle)

    # Utilities
    q_VE          = rnorm(n_sim, mean = 20, sd = 1), # utility when healthy
    q_VE_comp     = rnorm(n_sim, mean = 19, sd = 2), # utility when sick
    q_loss_biopsy = -rbeta(n_sim, shape1 = 4, shape2 = 380)
```

```
  )
  return(df_psa_params)
}
# Try it
generate_psa_params(10)
```

```
# Generate PSA dataset for CEA
# 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)

# Histogram of parameters
ggplot(reshape2::melt(df_psa_input, variable.name = "Parameter",
                      value.name = "Parameter value"),
                      aes(x = `Parameter value`)) +
                      facet_wrap(~Parameter, scales = "free") +
                      geom_histogram(aes(y = ..density..), alpha = 0.8) +
                      scale_x_continuous(breaks = number_ticks(3)) +
                      ylab("") +
                      theme_bw(base_size = 14) +
                      theme(axis.text.y = element_blank())

# 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
```

## 07.1 Conduct probabilistic sensitivity analysis

```
# Run decision tree on each parameter set of PSA input dataset
for(i in 1:n_sim){
  l_psa_input <- update_param_list(l_params_all, df_psa_input[i, ])
  df_out_psa  <- calculate_ce_out(l_psa_input)
  df_c[i, ] <- df_out_psa$Cost
  df_e[i, ] <- df_out_psa$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 = " "))
  }
}
```

### 07.2 Create PSA object for dampack

```
l_psa <- make_psa_obj(cost          = df_c,
                      effectiveness = df_e,
                      parameters    = df_psa_input,
                      strategies    = v_names_str)
```

#### 07.2.1 Save PSA objects

```
save(df_psa_input, df_c, df_e, v_names_str, n_str,
     l_psa,
     file = "decision_tree_HVE_PSA_dataset.RData")
```

### 07.3 Create probabilistic analysis graphs

```
load(file = "decision_tree_HVE_PSA_dataset.RData")
```

Vector with willingness-to-pay (WTP) thresholds.

```
v_wtp <- seq(0, 600000, by = 10000)
```

#### 07.3.1 Cost-Effectiveness Scatter plot

```
plot(l_psa)
```

### 07.4 Conduct CEA with probabilistic output

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

# Save CEA table with ICERs
# As .RData
save(df_cea_psa,
     file = "decision_tree_HVE_probabilistic_CEA_results.RData")
```

```
# As .csv
write.csv(df_cea_psa,
          file = "decision_tree_HVE_probabilistic_CEA_results.csv")
```

### 07.4.1 Plot cost-effectiveness frontier

```
plot(df_cea_psa)
```

### 07.4.2 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)
# CEAC & CEAF plot
plot(ceac_obj)
```

### 07.4.3 Expected Loss Curves (ELCs)

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

# ELC plot
plot(elc_obj, log_y = FALSE)
```

### 07.4.4 Expected value of perfect information (EVPI)

```
evpi <- calc_evpi(wtp = v_wtp, psa = l_psa)
# EVPI plot
plot(evpi, effect_units = "QALY")
```

### 07.4.5 Expected value of partial perfect information (EVPPI)

```
evppi <- calc_evppi(psa = l_psa,
                    wtp = v_wtp,
                    params = c("q_VE_comp"),
                    outcome = c("nmb"),
                    type = c("gam", "poly"),
                    poly.order = 2,
                    k = -1,
                    pop = 1
)

dampack:::plot.evppi(evppi)
```