

SA: Simple 3-state Markov 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. 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>
- Alarid-Escudero F, Krijkamp EM, Enns EA, Yang A, Hunink MGM, Pechlivanoglou P, Jalal H. Cohort State-Transition Models in R: A Tutorial. *arXiv:200107824v2*. 2020:1-48. <http://arxiv.org/abs/2001.07824>
- 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>

Copyright 2017, THE HOSPITAL FOR SICK CHILDREN AND THE COLLABORATING INSTITUTIONS. All rights reserved in Canada, the United States and worldwide. Copyright, trademarks, trade names and any and all associated intellectual property are exclusively owned by THE HOSPITAL FOR SICK CHILDREN and the collaborating institutions. These materials may be used, reproduced, modified, distributed and adapted with proper attribution.

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", "devtools", "scales", "ellipse", "ggplot2", "lazyeval", "igraph", "ggraph", "reshape2"
# install_github("DARTH-git/dampack", force = TRUE) Uncomment if there is a newer version
# install_github("DARTH-git/darthtools", force = TRUE) Uncomment if there is a newer version
p_load_gh("DARTH-git/dampack", "DARTH-git/darthtools")
```

02 Load functions

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

03 Input model parameters

```
# Strategy names
v_names_str <- c("Standard of Care", "Treatment")

# Markov model parameters
v_names_states <- c("Healthy", "Sick", "Dead") # state names
n_t <- 60 # number of cycles

v_init <- c("Healthy" = 1,
            "Sick" = 0,
            "Dead" = 0) # initial cohort distribution (everyone allocated to the
                        # "healthy" state)

# Transition probabilities
p_HD <- 0.02 # probability of dying when healthy
p_HS <- 0.05 # probability of becoming sick when healthy, under standard of care
p_HS_trt <- 0.03 # probability of becoming sick when healthy, under treatment
p_SD <- 0.1 # probability of dying when sick

# Costs and utilities
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_trt <- 8000 # cost of treatment (per cycle)
u_H <- 0.8 # utility when healthy
u_S <- 0.5 # utility when sick
u_D <- 0 # utility when dead
```

```

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_names_states) # 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)

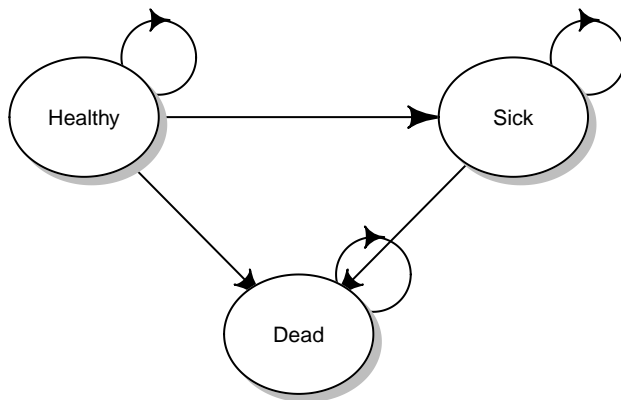
```

Draw the state-transition cohort model

```

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 Define and initialize matrices and vectors

04.1 Cohort trace

```

# create the cohort trace
m_M_SoC <- m_M_trt <- matrix(NA,
                             nrow = n_t + 1, # create Markov trace (n_t + 1 because R doesn't
                                                # understand Cycle 0)
                             ncol = n_states,
                             dimnames = list(0:n_t, v_names_states))

m_M_SoC[1, ] <- m_M_trt[1, ] <- v_init      # initialize first cycle of Markov trace

```

04.2 Transition probability matrix

```
# create the transition probability matrices
m_P_SoC <- m_P_trt <- matrix(0,
                              nrow = n_states, ncol = n_states,
                              dimnames = list(v_names_states, v_names_states)) # name the columns and

# print the probability matrices
m_P_SoC # for standard of care
```

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

```
m_P_trt # treatment
```

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

Fill in the transition probability matrix:

```
# For Standard of Care
# from Healthy
m_P_SoC["Healthy", "Healthy"] <- (1 - p_HD) * (1 - p_HS)
m_P_SoC["Healthy", "Sick"]    <- (1 - p_HD) * p_HS
m_P_SoC["Healthy", "Dead"]    <- p_HD

# from Sick
m_P_SoC["Sick", "Sick"] <- 1 - p_SD
m_P_SoC["Sick", "Dead"] <- p_SD

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

# Under treatment
m_P_trt <- m_P_SoC # Assign the matrix for standard of care to the transition probability matrix for t
# replace values that are different under treatment
m_P_trt["Healthy", "Healthy"] <- (1 - p_HD) * (1 - p_HS_trt)
m_P_trt["Healthy", "Sick"]    <- (1 - p_HD) * p_HS_trt
```

04.3 Check if transition probability structure and probabilities are valid

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

05 Run Markov model

```
for (t in 1:n_t){ # loop through the number of cycles
  m_M_SoC[t + 1, ] <- m_M_SoC[t, ] %*% m_P_SoC # estimate the state vector for the next cycle (t + 1)
  m_M_trt[t + 1, ] <- m_M_trt[t, ] %*% m_P_trt # for treatment
}
```

06 Compute and Plot Epidemiological Outcomes

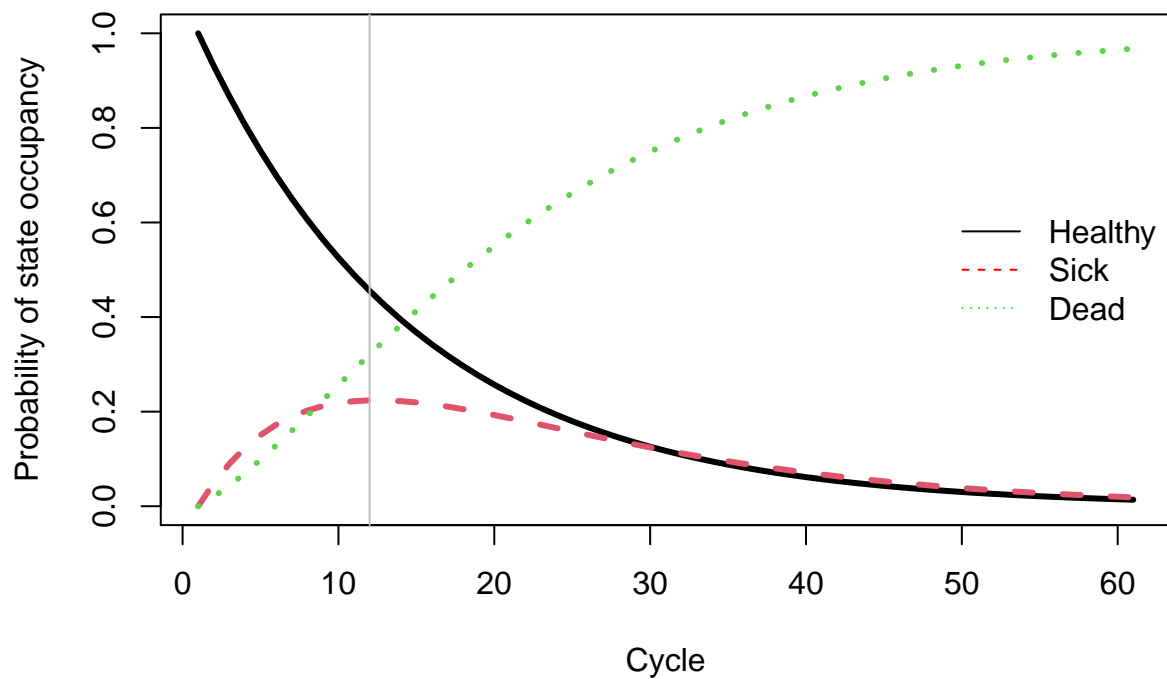
06.1 Cohort trace

Standard of Care:

```
matplot(m_M_SoC, type = 'l',
        ylab = "Probability of state occupancy",
        xlab = "Cycle",
        main = "Cohort Trace - standard of care", lwd = 3) # create a plot of the data
legend("right", v_names_states, col = c("black", "red", "green"),
      lty = 1:3, bty = "n") # add a legend to the graph

abline(v = which.max(m_M_SoC[, "Sick"]), col = "gray") # plot a vertical line that helps identify
```

Cohort Trace – standard of care



Treatment:

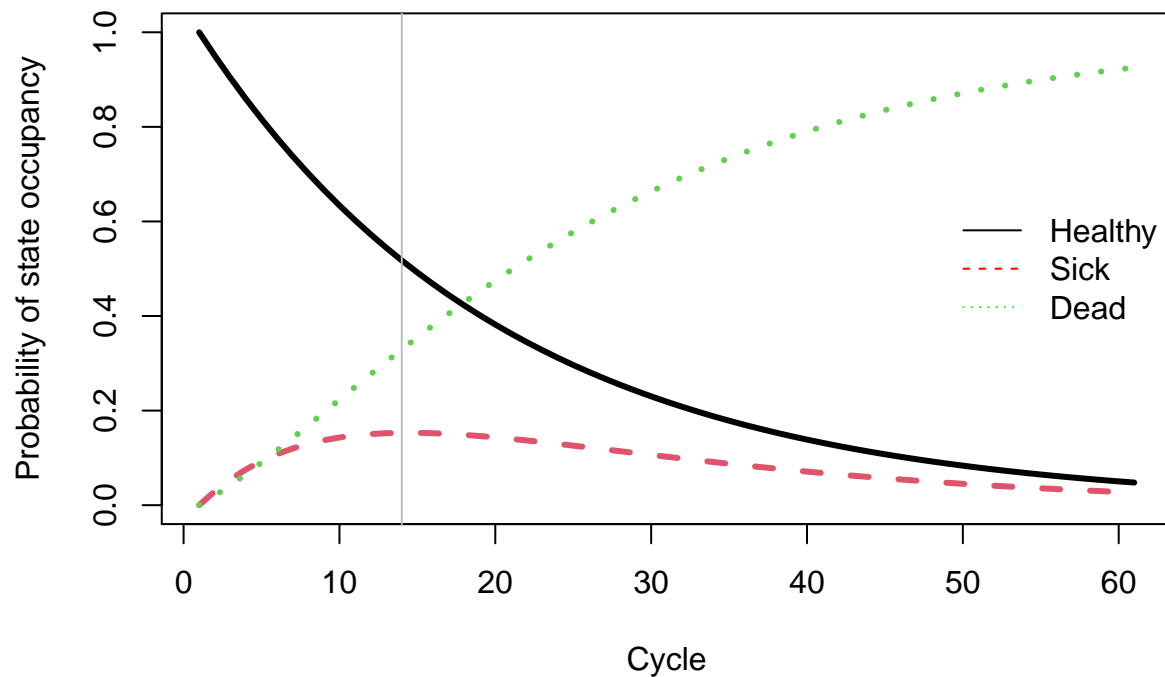
```

matplot(m_M_trt, type = 'l',
        ylab = "Probability of state occupancy",
        xlab = "Cycle",
        main = "Cohort Trace - treatment", lwd = 3)      # create a plot of the data
legend("right", v_names_states, col = c("black", "red", "green"),
        lty = 1:3, bty = "n")                          # add a legend to the graph

abline(v = which.max(m_M_trt[, "Sick"]), col = "gray")  # plot a vertical line that helps identifying

```

Cohort Trace – treatment



06.2 Overall Survival (OS)

Standard of Care:

```

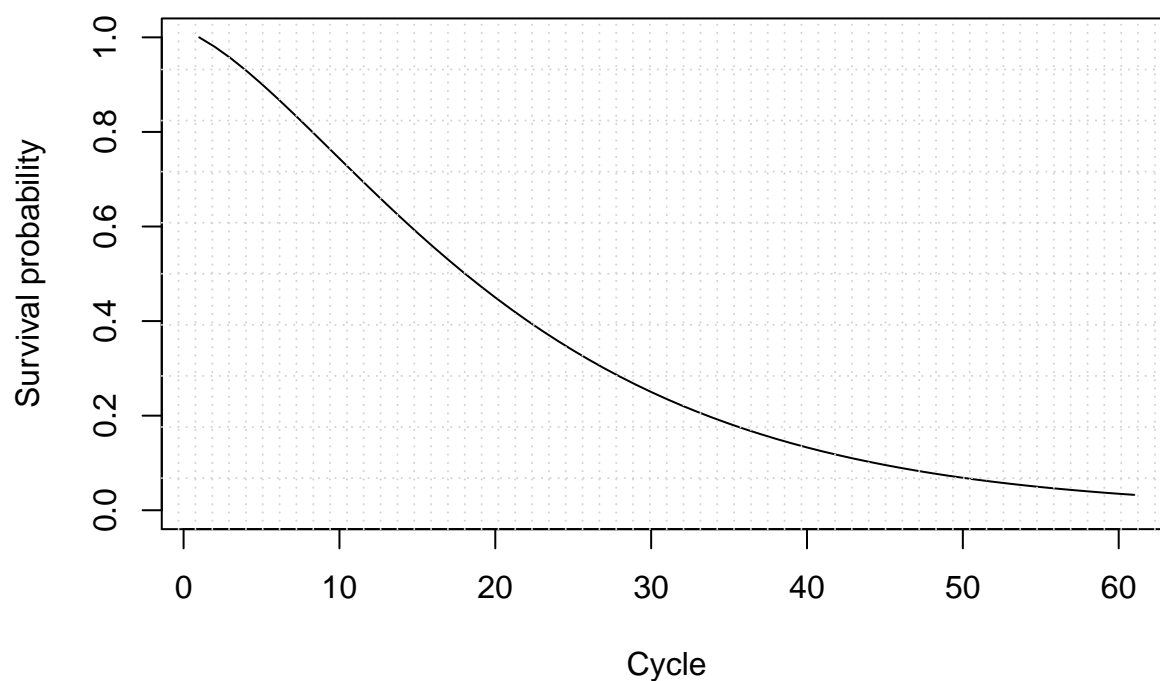
v_os_SoC <- 1 - m_M_SoC[, "Dead"]      # calculate the overall survival (OS) probability
v_os_SoC <- rowSums(m_M_SoC[, 1:2])    # alternative way of calculating the OS probability

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

# add grid
grid(nx = n_t, ny = 10, col = "lightgray", lty = "dotted", lwd = par("lwd"),
     equilog = TRUE)

```

Overall Survival – Standard of Care



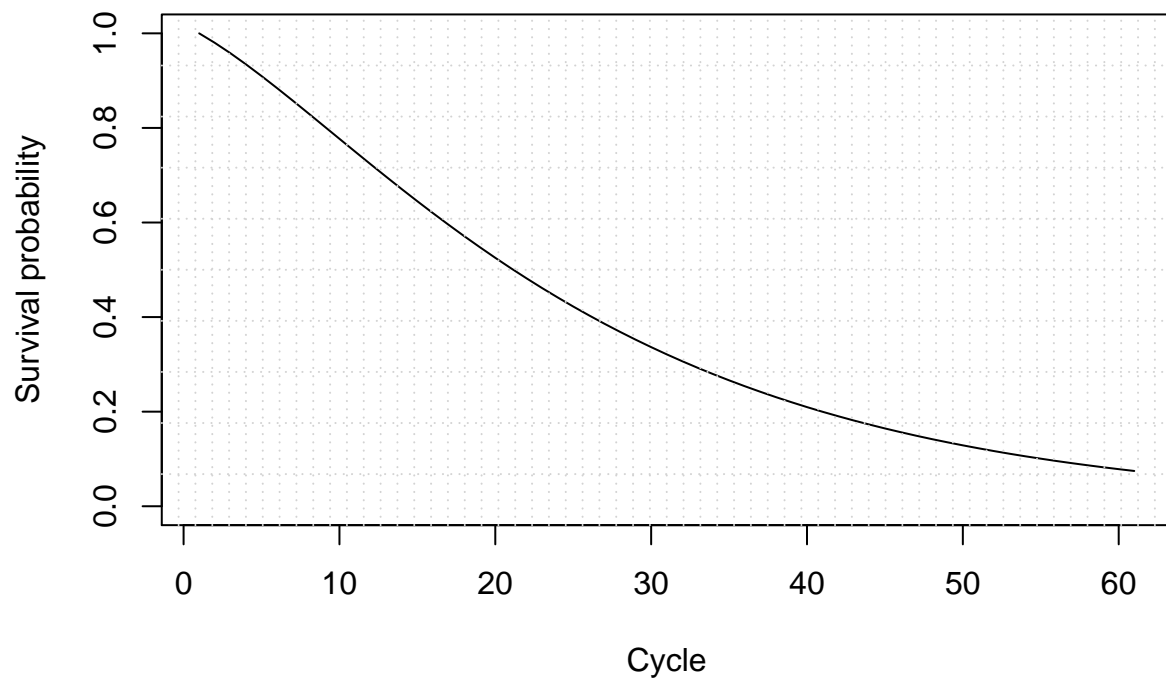
Treatment:

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

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

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

Overall Survival – Treatment



06.2.1 Life Expectancy (LE)

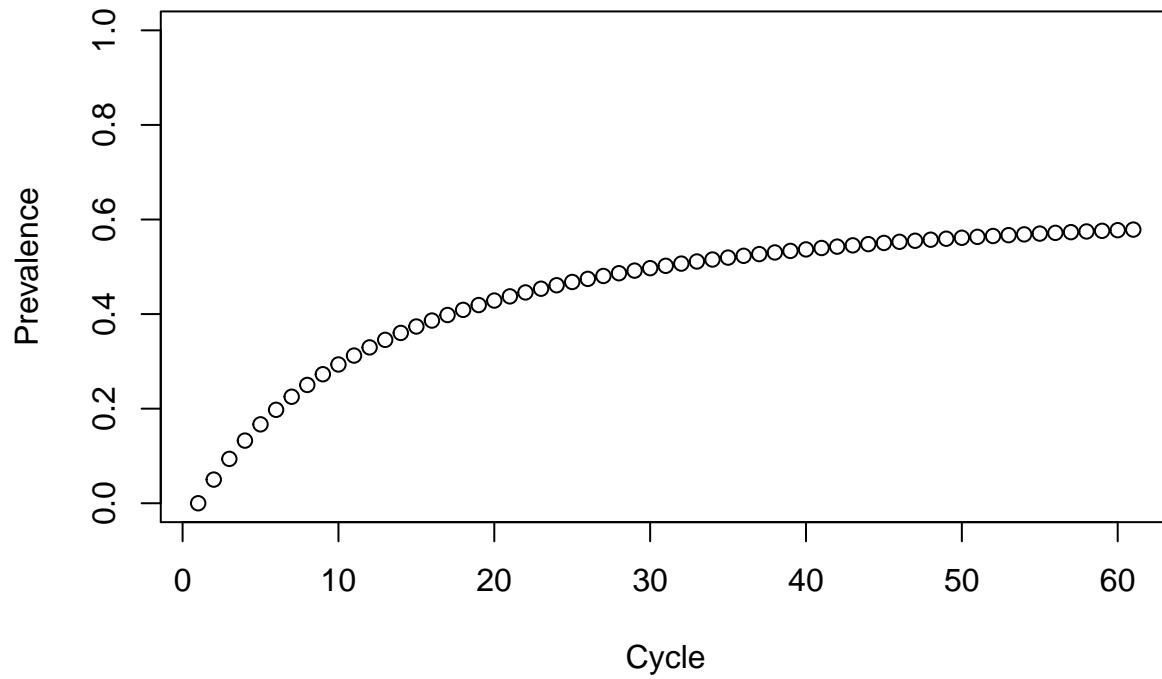
```
v_le_SoC <- sum(v_os_SoC) # summing probability of OS over time (i.e. life expectancy)
v_le_trt <- sum(v_os_trt) # summing probability of OS over time (i.e. life expectancy), treatment
```

06.3 Disease prevalence

Standard of Care:

```
v_prev_SoC <- m_M_SoC[, "Sick"]/v_os_SoC
plot(v_prev_SoC,
     ylim = c(0, 1),
     ylab = "Prevalence",
     xlab = "Cycle",
     main = "Disease prevalence - Standard of care")
```

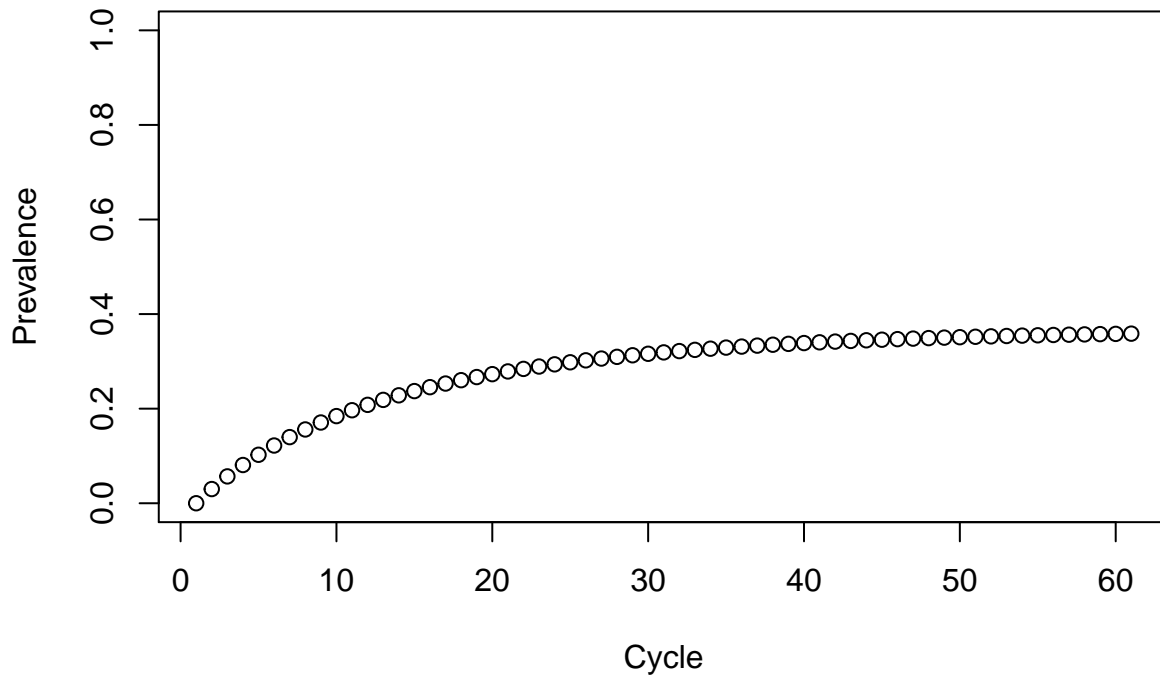

Disease prevalence – Standard of care



Treatment:

```
v_prev_trt <- m_M_trt[, "Sick"]/v_os_trt
plot(v_prev_trt,
     ylim = c(0, 1),
     ylab = "Prevalence",
     xlab = "Cycle",
     main = "Disease prevalence - Treatment")
```

Disease prevalence – Treatment



07 Compute Cost-Effectiveness Outcomes

07.1 Mean Costs and QALYs

```
# per cycle
# calculate expected costs by multiplying m_M with the cost vector for the different
# health states
v_tc_SoC <- m_M_SoC %*% c(c_H, c_S, c_D) # Standard of Care
v_tc_trt <- m_M_trt %*% c(c_H, c_S, c_D) # Treatment
v_tc_trt[1] <- v_tc_trt[1] + c_trt
# calculate expected QALYs by multiplying m_M with the utilities for the different
# health states
v_tu_SoC <- m_M_SoC %*% c(u_H, u_S, u_D) # Standard of Care
v_tu_trt <- m_M_trt %*% c(u_H, u_S, u_D) # Treatment
```

07.2 Discounted Mean Costs and QALYs

```
# Discount costs by multiplying the cost vector with discount weights
tc_d_SoC <- t(v_tc_SoC) %*% v_dwc # Standard of Care
tc_d_trt <- t(v_tc_trt) %*% v_dwc # Treatment
# Discount QALYS by multiplying the QALYs vector with discount weights
tu_d_SoC <- t(v_tu_SoC) %*% v_dwe # Standard of Care
tu_d_trt <- t(v_tu_trt) %*% v_dwe # Treatment
```

```

# store them into a vector
v_tc_d    <- c(tc_d_SoC, tc_d_trt)
v_tu_d    <- c(tu_d_SoC, 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 Standard of Care  8043.139 10.25087
## 2           Treatment 16028.490 11.73928

```

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 Standard of Care  8043.139 10.25087      NA      NA      NA      ND
## 2           Treatment 16028.490 11.73928 7985.351  1.488415 5365.002      ND

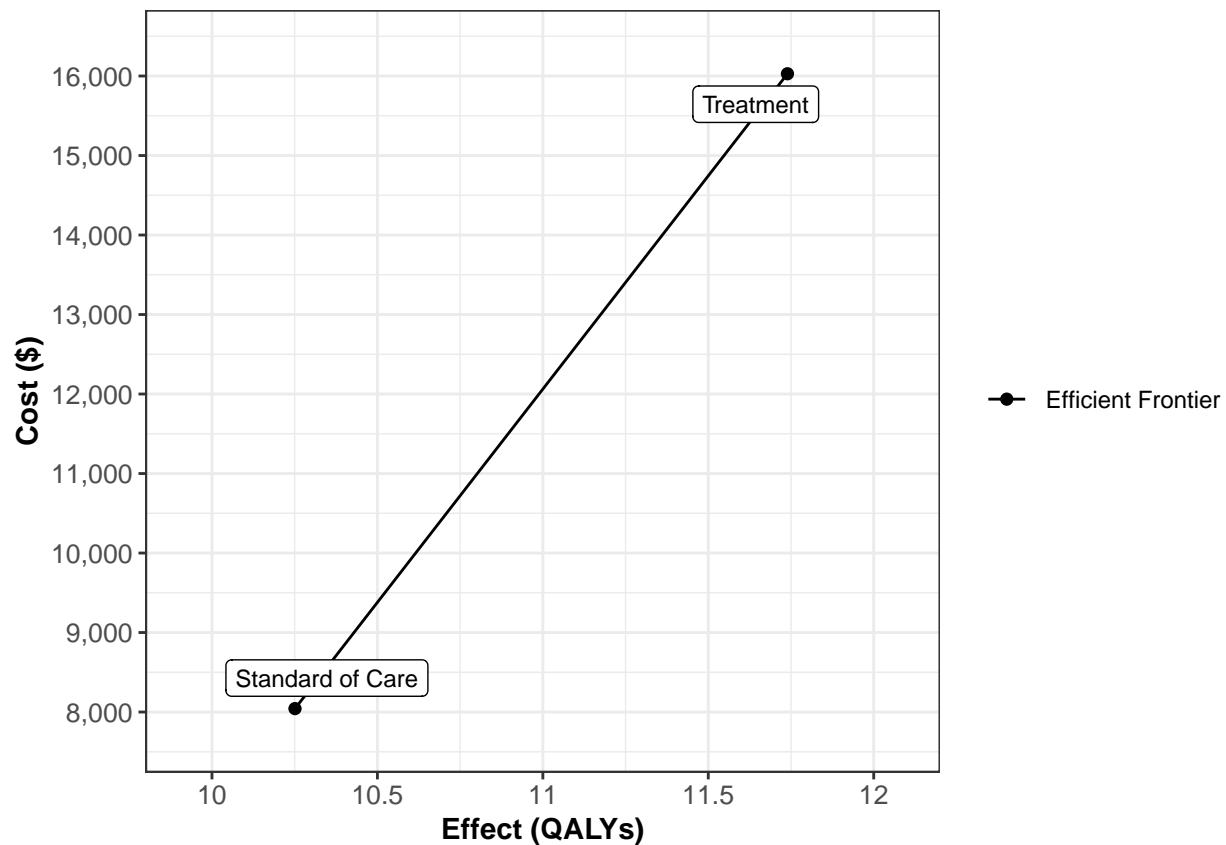
```

07.4 Plot frontier of the Markov model

```

plot(df_cea, effect_units = "QALYs", xlim = c(10, 12))

```



note: you need to adjust the xlim values to values that are covering the range of effect values in your analysis

08 Probabilistic Sensitivity Analysis (PSA)

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    = 0.02, # probability of dying when healthy
  p_HS    = 0.05, # probability of becoming sick when healthy, conditioned on not dying
  p_HS_trt = 0.03, # probability of becoming sick when healthy, conditioned on not dying
  p_SD    = 0.1,  # probability of dying when sick
  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_trt   = 8000, # one-time cost of treatment (at first cycle)
  u_H     = 0.8,  # utility when healthy
  u_S     = 0.5,  # utility when sick
  u_D     = 0,    # utility when dead
  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_3state.R")
# Test function
calculate_ce_out(l_params_all)
```

```
##           Strategy      Cost   Effect      NMB
## 1 Standard of Care 8043.139 10.25087 94465.51
## 2           Treatment 16028.490 11.73928 101364.32
```

08.3 Generate PSA datasets

```
# 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)
    # probability to become sick when healthy
    p_HS = rbeta(n_sim, shape1 = 24, shape2 = 450),
    p_HS_trt = rbeta(n_sim, shape1 = 9, shape2 = 281), # under treatment
    # probability of dying when healthy
    p_HD = rbeta(n_sim, shape1 = 16, shape2 = 767),
    # probability of dying when sick
    p_SD = rbeta(n_sim, shape1 = 22.4, shape2 = 201.6),

    # Cost vectors with length n_sim
    # cost of remaining one cycle in state H
    c_H = rgamma(n_sim, shape = 16, scale = 25),
    # cost of remaining one cycle in state S1
    c_S = rgamma(n_sim, shape = 100, scale = 10),
    # cost of being in the death state
    c_D = 0,
    # cost of treatment (per cycle)
    c_trt = rgamma(n_sim, shape = 64, scale = 125),

    # Utility vectors with length n_sim
    # utility when healthy
    u_H = rbeta(n_sim, shape1 = 50.4, shape2 = 12.6),
    # utility when sick
    u_S = rbeta(n_sim, shape1 = 49.5, shape2 = 49.5),
    # utility when dead
    u_D = 0
  )
  return(df_psa)
}
# Try it
gen_psa(10)
```

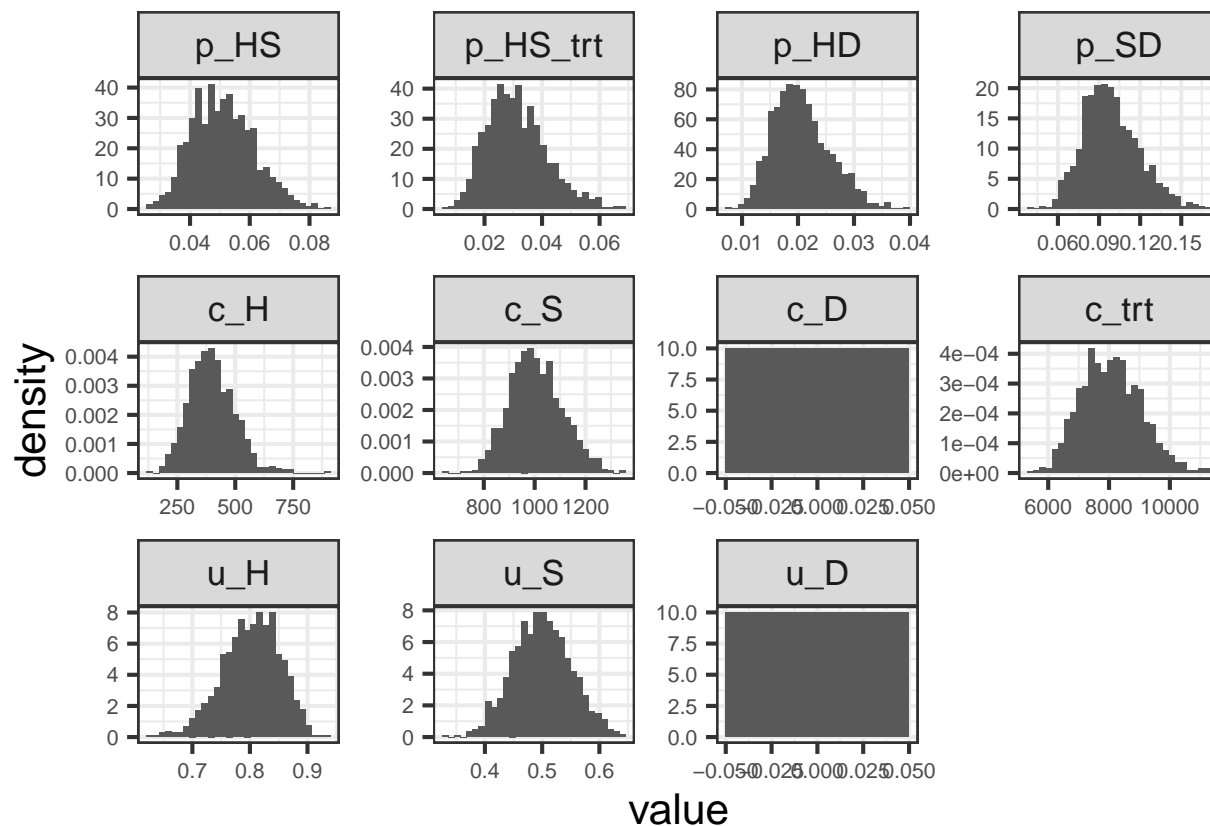
```
##           p_HS   p_HS_trt       p_HD       p_SD       c_H       c_S c_D       c_trt
## 1  0.05116365 0.02717469 0.02650780 0.09350365 331.7253 1001.3478 0 6217.035
## 2  0.04095451 0.01181545 0.02787410 0.15823599 535.2239 995.9279 0 9377.706
## 3  0.05334926 0.01972404 0.01561002 0.12357583 334.2034 955.0140 0 6591.331
## 4  0.03627100 0.03901602 0.02620191 0.08771020 352.7561 990.4764 0 8652.337
## 5  0.04782122 0.02128019 0.01998096 0.10799984 260.4642 943.7245 0 8555.065
## 6  0.07206924 0.02272066 0.01521342 0.12685251 291.1215 983.1453 0 8604.577
## 7  0.04803791 0.04457119 0.01499307 0.08911761 483.8274 1050.5295 0 9942.503
## 8  0.04335723 0.02882206 0.02131098 0.05814786 541.4731 864.3322 0 6018.085
## 9  0.05302297 0.03409499 0.01709267 0.15922549 551.1535 1080.4473 0 7571.225
## 10 0.03096171 0.01553948 0.01983932 0.10406523 315.7027 984.6521 0 7875.281
##           u_H       u_S u_D
## 1  0.8559721 0.4815761 0
## 2  0.7783084 0.5349337 0
## 3  0.8628224 0.5227023 0
## 4  0.8968823 0.4871518 0
## 5  0.8179546 0.5294793 0
## 6  0.6717953 0.5249675 0
## 7  0.8177386 0.5600215 0
## 8  0.8145828 0.5243320 0
## 9  0.8153981 0.5503887 0
## 10 0.8835447 0.5837644 0
```

```
# Number of simulations
n_sim <- 1000

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

```
##           p_HS   p_HS_trt       p_HD       p_SD       c_H       c_S c_D       c_trt
## 1 0.05116365 0.01954245 0.02621456 0.13853135 307.8380 899.5153 0 6875.154
## 2 0.04095451 0.04346716 0.01760834 0.13075427 466.9600 1066.1617 0 8755.292
## 3 0.05334926 0.03357981 0.02404573 0.08454761 235.2890 955.0145 0 6582.810
## 4 0.03627100 0.03806155 0.02716409 0.10457346 306.5100 789.6969 0 7027.446
## 5 0.04782122 0.03737024 0.01716697 0.14474739 535.1188 868.0570 0 5661.605
## 6 0.07206924 0.04257367 0.01710648 0.12289879 353.6715 991.6205 0 10960.742
##           u_H       u_S u_D
## 1 0.7965014 0.4473598 0
## 2 0.7659582 0.4644020 0
## 3 0.7593770 0.4215045 0
## 4 0.8338376 0.5052819 0
## 5 0.8798517 0.5181703 0
## 6 0.7306958 0.5905503 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..)) +
  theme_bw(base_size = 16) +
  theme(axis.text = element_text(size=8))
```



```
# 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
```

08.4 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
```

08.4.1 Create PSA object for dampack

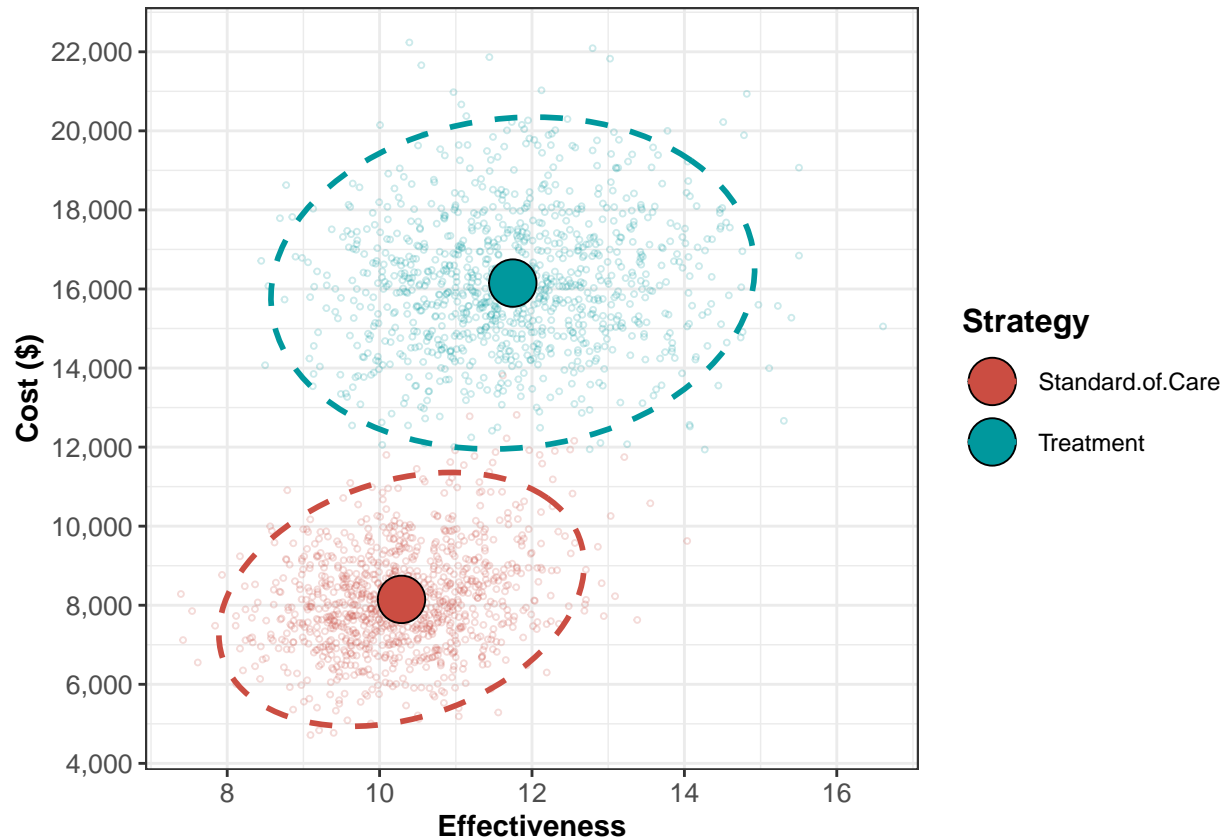
```
l_psa <- make_psa_obj(cost      = df_c,  
                      effectiveness = df_e,  
                      parameters  = df_psa_input,  
                      strategies  = v_names_str)
```

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

```
v_wtp <- seq(0, 5000, by = 1000)
```

08.4.2 Cost-Effectiveness Scatter plot

```
plot(l_psa)
```



08.4.3 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)
```

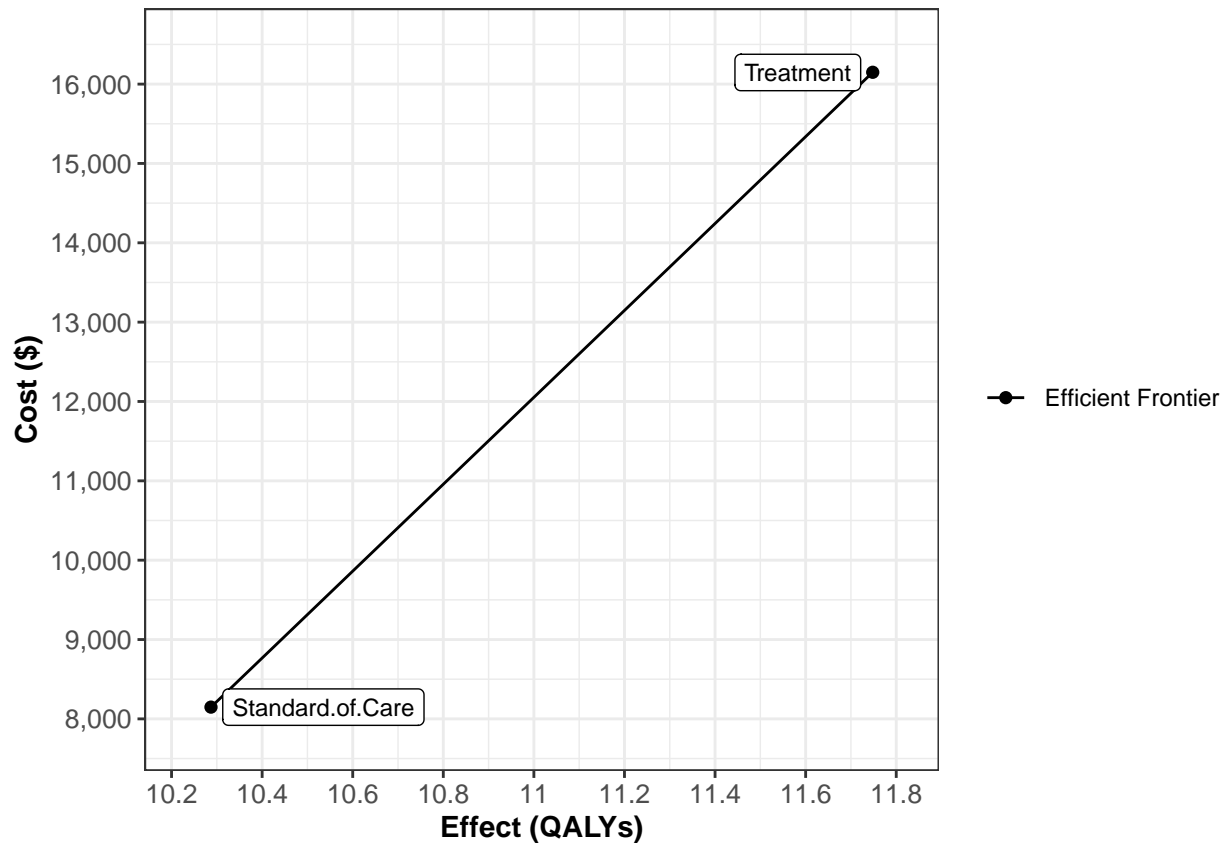


```
df_cea_psa <- calculate_icers(cost      = df_out_ce_psa$meanCost,
                             effect    = df_out_ce_psa$meanEffect,
                             strategies = df_out_ce_psa$Strategy)
df_cea_psa
```

```
##           Strategy      Cost    Effect Inc_Cost Inc_Effect      ICER Status
## 1 Standard.of.Care 8148.004 10.28706      NA      NA      NA      ND
## 2      Treatment 16149.044 11.74808 8001.04  1.461028 5476.307      ND
```

08.4.4 Plot cost-effectiveness frontier

```
plot(df_cea_psa)
```

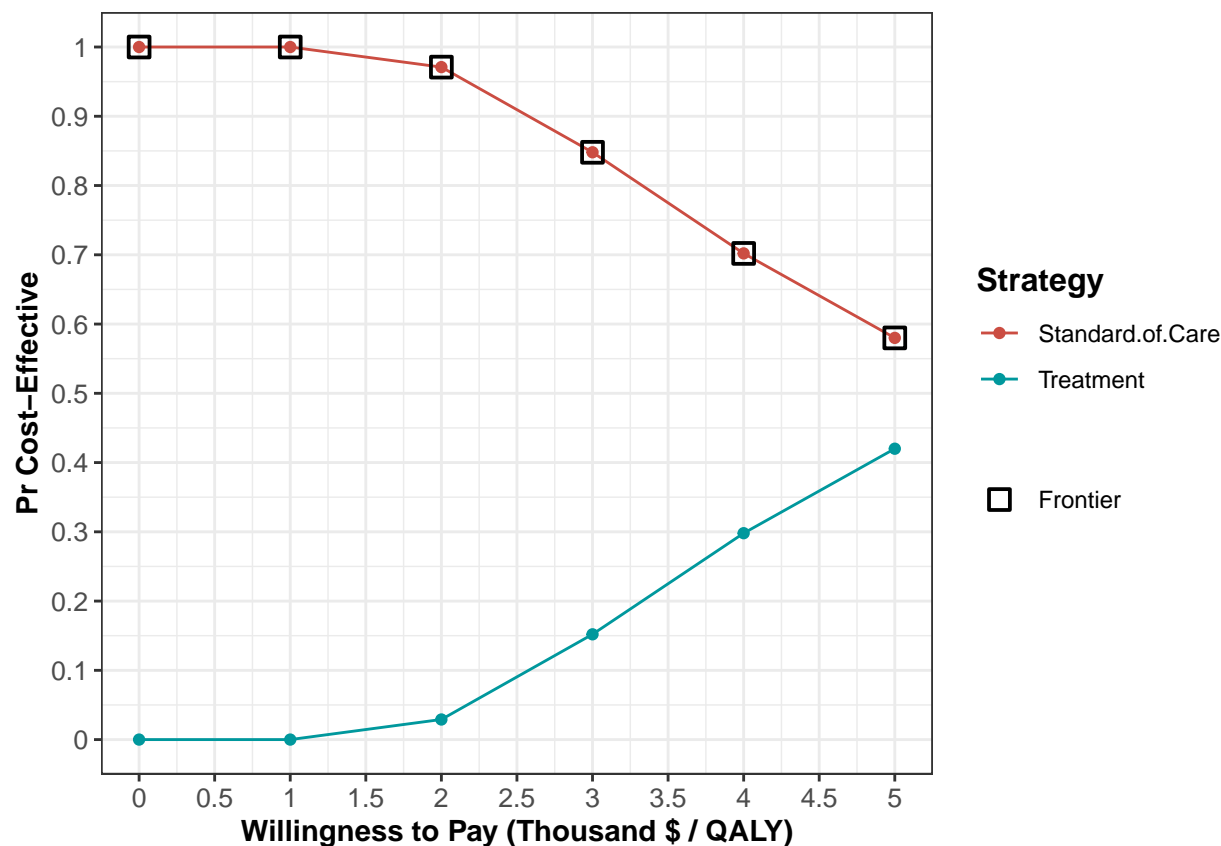


08.4.5 Cost-effectiveness acceptability curves (CEACs) and frontier (CEAF)

```
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      5000 Standard.of.Care
```

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



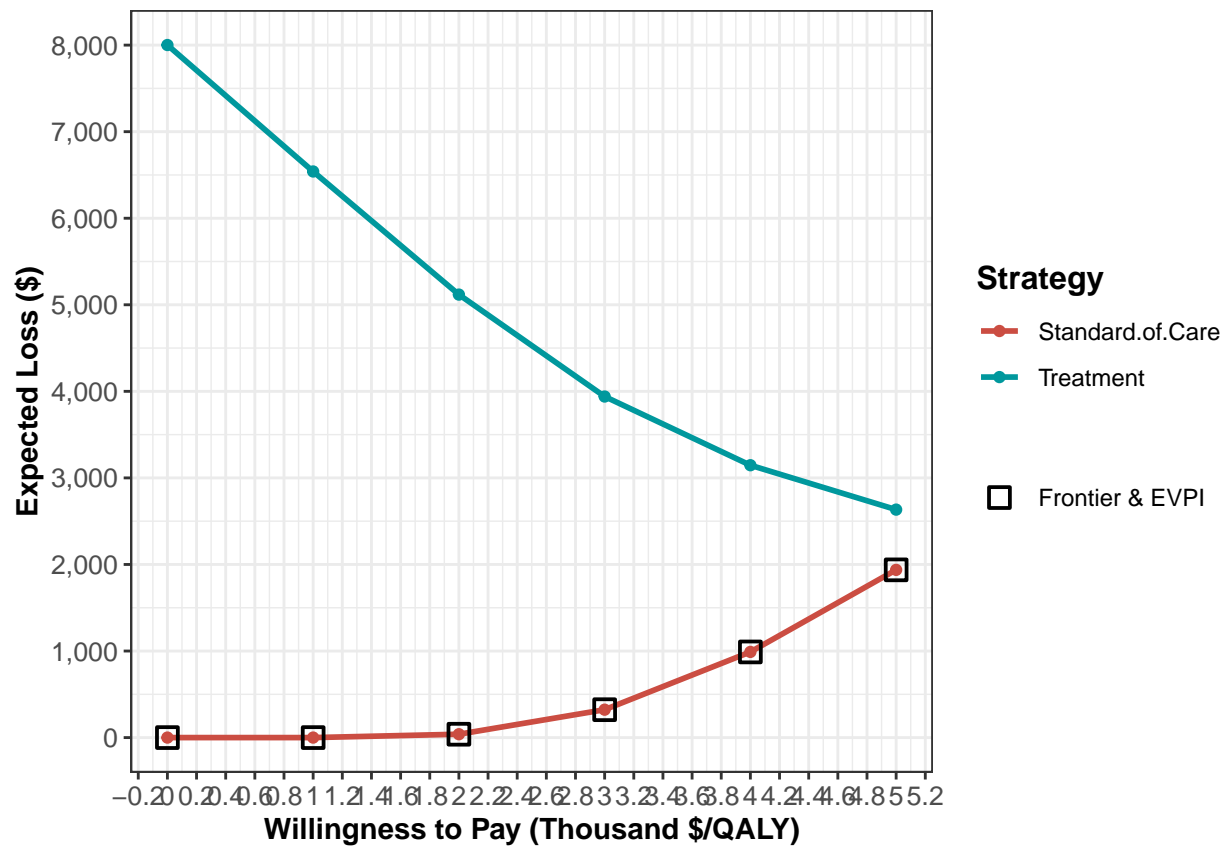
08.4.6 Expected Loss Curves (ELCs)

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
```

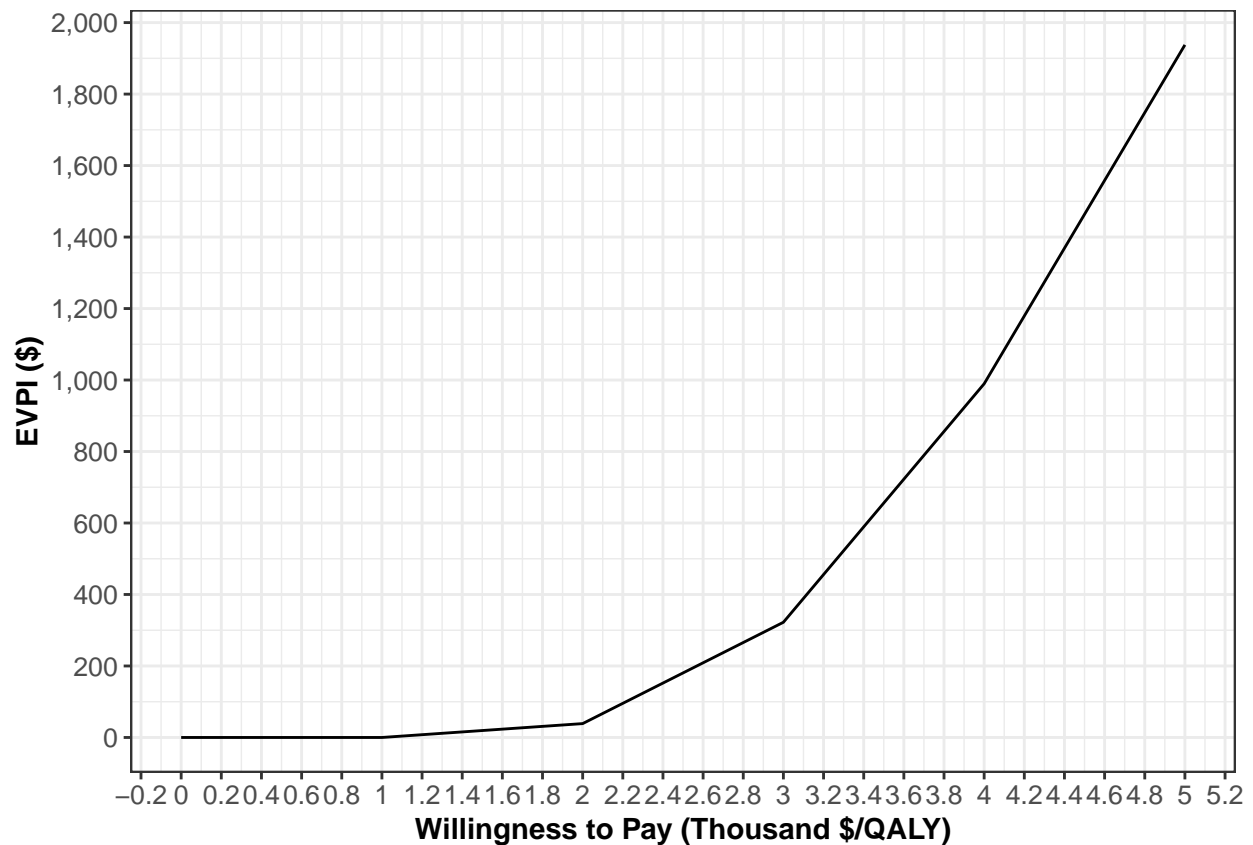
##	WTP	Strategy	Expected_Loss	On_Frontier
## 1	0	Standard.of.Care	0.00000	TRUE
## 2	0	Treatment	8001.03979	FALSE
## 3	1000	Standard.of.Care	0.00000	TRUE
## 4	1000	Treatment	6540.01147	FALSE
## 5	2000	Standard.of.Care	38.76766	TRUE
## 6	2000	Treatment	5117.75081	FALSE
## 7	3000	Standard.of.Care	322.26595	TRUE
## 8	3000	Treatment	3940.22079	FALSE
## 9	4000	Standard.of.Care	989.52884	TRUE
## 10	4000	Treatment	3146.45536	FALSE
## 11	5000	Standard.of.Care	1937.60524	TRUE
## 12	5000	Treatment	2633.50344	FALSE

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



08.4.7 Expected value of perfect information (EVPI)

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



09 Using R package hesim

```
p_load("hesim")
```

09.1 Model setup

Here we define target population and intervention strategies.

We have one representative patient here of age 25, we can think of this as a cohort of homogenous patients instead of one individual patient.

```
# define strategies
strategies <- data.frame(
  strategy_id = 1:n_str,
  strategy_name = v_names_str
)
# define patient cohort
patients <- data.frame(
  patient_id = 1,
  age = 25
)
# create dataset with
hesim_dat <- hesim_data(
```

```

    strategies = strategies,
    patients    = patients
  )
hesim_dat

```

```

## $strategies
##   strategy_id  strategy_name
## 1             1 Standard of Care
## 2             2      Treatment
##
## $patients
##   patient_id age
## 1           1  25
##
## attr(,"class")
## [1] "hesim_data"

```

09.2 parameters

```

params <- list(
  # medical costs
  c_medical    = c(Healthy = c_H, Sick = c_S),
  c_medical_se = c(Healthy = 100, Sick = 100),
  # treatment costs (embedded in medical costs since only those who are sick get treated)
  c_trt        = c_trt,
  # state utilities
  u_mean       = c(Healthy = u_H, Sick = u_S),
  u_se         = c(Healthy = 0.05, Sick = 0.05)
)

```

09.3 PSA setup

```
rng_def()
```

```

rng_def <- define_rng({
  list( # Parameters to return
    # medical costs
    c_medical = gamma_rng(mean = c_medical, sd = c_medical_se),
    c_trt     = gamma_rng(mean = c_trt,    sd = 1000),
    # state utilities
    u         = beta_rng(mean = u_mean, sd = u_se)
  )
}, n = 1000)

```

09.4 Transform parameters

```

input_data <- hesim::expand(hesim_dat, by = c("strategies", "patients"))
head(input_data)

```

```
##      strategy_id patient_id    strategy_name age
## 1:           1           1 Standard of Care  25
## 2:           2           1      Treatment  25
```

The function `define_tparams()` returns:

- `tpmatrix`: The transition probability matrix
- `utility`: Utility assigned to each health state
- `costs`: Costs assigned to each health state or each cost category

Your task: write mathematical expressions

The function: automatically loops over PSA iterations (running the model on each sampled parameter set)

```
tparams_def <- define_tparams({
  # treatment reduces the risk of getting sick
  rr <- ifelse(strategy_name == "Standard of Care", 1, p_HS_trt / p_HS) # relative risk

  list(
    tpmatrix = tpmatrix(
      (1 - p_HD) * (1 - p_HS * rr), (1 - p_HD) * (p_HS * rr), p_HD,
      0, C, p_SD,
      0, 0, 1
    ),

    utility = u,
    costs = list(
      treatment = ifelse(strategy_name == "Standard of Care", 0, c_trt),
      medical    = c_medical
    )
  )
})
```

09.5 Simulation

Construct model:

```
mod_def <- define_model(tparams_def = tparams_def,
                        rng_def      = rng_def,
                        params       = params)
```

Initialize-model:

```
cost_args <- list(
  treatment = list(method = "starting"),
  medical    = list(method = "wlos")
)
econmod <- create_CohortDtstm(mod_def, input_data, cost_args = cost_args)
```

Simulate outcomes:

```
econmod$sim_stateprobs(n_cycles = n_t)
head(econmod$stateprobs_)
```

```
##      sample strategy_id patient_id grp_id state_id t      prob
## 1:      1          1          1      1          1 0 1.0000000
## 2:      1          1          1      1          1 1 0.9310000
## 3:      1          1          1      1          1 2 0.8667610
## 4:      1          1          1      1          1 3 0.8069545
## 5:      1          1          1      1          1 4 0.7512746
## 6:      1          1          1      1          1 5 0.6994367
```

```
econmod$sim_qalys(dr = d_e, lys = TRUE, integrate_method = "riemann_right")
head(econmod$qalys_)
```

```
##      sample strategy_id patient_id grp_id state_id dr      qalys      lys
## 1:      1          1          1      1          1 0.03 6.484607 9.339850
## 2:      1          1          1      1          2 0.03 1.841271 3.861427
## 3:      1          2          1      1          1 0.03 8.199613 11.809992
## 4:      1          2          1      1          2 0.03 1.361197 2.854639
## 5:      2          1          1      1          1 0.03 8.063655 9.339850
## 6:      2          1          1      1          2 0.03 1.849052 3.861427
```

```
econmod$sim_costs(dr = d_c, integrate_method = "riemann_right")
head(econmod$costs_)
```

```
##      sample strategy_id patient_id grp_id state_id dr category costs
## 1:      1          1          1      1          1 0.03 treatment 0.000
## 2:      1          1          1      1          2 0.03 treatment 0.000
## 3:      1          2          1      1          1 0.03 treatment 7819.843
## 4:      1          2          1      1          2 0.03 treatment 0.000
## 5:      2          1          1      1          1 0.03 treatment 0.000
## 6:      2          1          1      1          2 0.03 treatment 0.000
```

09.6 Cost-effectiveness analysis

```
ce_sim <- econmod$summarize()
cea_pw_out <- cea_pw(ce_sim,
  comparator = 1,
  dr_qalys = 0.03, dr_costs = 0.03,
  k = seq(0, 5000, 1000))

## @knitr icer
icer_tbl(cea_pw_out, colnames = strategies$strategy_name)
```

```
##      Standard of Care Treatment
## Incremental QALYs "-" "1.47 (1.21, 1.72)"
## Incremental costs "-" "7,990 (6,070, 10,233)"
## Incremental NMB "-" "65,685 (52,214, 77,821)"
## ICER "-" "5,422"
## Conclusion "-" "Cost-effective"
```

10 Overview of **hesim**

Advantages:

- Easy to build models without having to program the complete model structure (easier for novice modelers).
- A lot of the modeling code are implemented for you in the back end.
- Suitable for modelers who are not familiar with R programming and functionality.
- Code written in C++ in the back end, which offers enhanced computational speed.

Disadvantages:

- Its rigid function structure inhibits its ability to tweak models or incorporate more complex model components (e.g. tunnel states, transition rewards).
- Does not provide the option for running deterministic analysis or one-way and two-way sensitivity analyses.
- Does not provide the ability to capture information about the specific transitions among health states (transition dynamics).
- Does not provide the ability to easily compute epidemiological outcomes.
- Does not allow costs to be applied to certain health states (at least not easily).

References

```
citation("hesim")
```

```
##
## To cite package 'hesim' in publications use:
##
##   Devin Incerti and Jeroen P. Jansen (2021). hesim: Health Economic
##   Simulation Modeling and Decision Analysis. R package version 0.5.0.
##   https://CRAN.R-project.org/package=hesim
##
## A BibTeX entry for LaTeX users is
##
##   @Manual{,
##     title = {hesim: Health Economic Simulation Modeling and Decision Analysis},
##     author = {Devin Incerti and Jeroen P. Jansen},
##     year = {2021},
##     note = {R package version 0.5.0},
##     url = {https://CRAN.R-project.org/package=hesim},
##   }
```