HEOR 533
Health State Transition in R Homework
Julia Fox
1/29/2024

A new end-of-life care was approved based on the clinical trial result that it can reduce the excess mortality due to progressive disease by 50%. It costs $500.

Decision makers (e.g. clinicians) requested information on whether they should provide this end-of-life care in addition to the original treatment to reduce the disease progression.

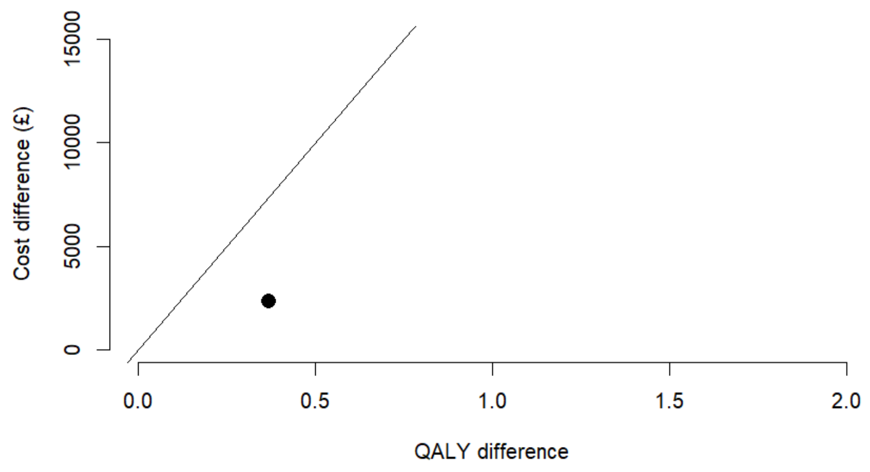
Using the same health state transition model, conduct a cost-effectiveness analysis considering three strategies: 1) without treatment 2) with treatment but no end-of-life care, 3) with both treatment and end-of-life care

Provide a table of cost and QALY of three strategies and report ICER. It is optional to generate ICER graph.

Treatment	Total Cost	Total QALYs
Without drug	$0	7755.952
With drug	$1000	8624.738
With drug and EOL care	$1500	8993.777

ICER of without drug vs drug: $7918.14

ICER of with drug vs with drug and EOL care: $7446.27



APPENDIX - R CODE

~~~
---
title: "Markov_model_realworld_explain"
author: "Kyu Lee edited by Julia Fox"
date: "2024-01-04"
output: pdf_document
---

```{r setup, include=FALSE}
knitr::opts_chunk$set(echo = TRUE)
```

## In this document, I provided line-by-line explanation of the code, 'Markov_model_realworld.R'
by Briggs et al

Overall structure of the code
< Base-case analysis>
1. Define fixed variables and parameters for the model
2. Prepare state-specific per-cycle cost, qaly, and transitional cost matrix
3. Prepare transition probability matrix (this will change in simulation given age-specific
mortality)
4. Prepare population matrix that keeps track of health state transition by cycle
5. Prepare a matrix ('trans') to record the transitional cost
6. Prepare empty matrices to record QALYs and Cost outcomes by cycle and by strategy
7. Define a function to calculate time-dependent transition probability based on age-specific
mortality
8. Simulate a cohort for 'no drug' and 'drug' scenario
9. Calculate cost and qaly of each scenario
10. Calculate ICER
11. Plot


<PSA>
1. Make the base-case analysis code as a function so that we can repeat the simulation by the
sampled parameter set
2. Random sampling of parameters
3. Define cost and qaly matrices to record outcomes by parameter sample
4. Run PSA
5. Plot


### Basecase analysis ###
1. Define parameters
~~~

~~~
```{r}
t_names <- c("without_drug", "with_drug", "with_drug_EOL")#t_names : strategy/scenario label
n_treatments <- length(t_names)#n_treatment: number of strategies

s_names  <- c("Asymptomatic_disease", "Progressive_disease", "Dead")#s_names: vector of
health states
n_states <- length(s_names)#n_states: number of health states

n_cohort <- 1000#n_cohort: cohort size
cycle <- 1#cycle: cycle length
n_cycles <- 46#n_cylces: total number of cycles

Initial_age <- 55#Initial_age: age at the beginning of simulation

cAsymp <- 500#cAsymp: cost of having asymptomatic disease (per-cycle cost)
cDeath <- 1000#cDeath: cost of death (transitional cost), only applies to the death among
progressive dx
cDrug <- 1000#cDrug: cost of drug
cProg <- 3000#cProg: cost of having progressive disease (per-cycle cost)
cEOL <- 500 #cEOL: cost of end-of-life care

uAsymp <- 0.95#uAsymp: utility of having asymptomatic disease
uProg <- 0.75#uProg: utility of having progressive disease

oDr <- 0.06#oDr: discount rate for qaly
cDr <- 0.06#cDr: discount rate for cost

tpDcm <- 0.15#tpDcm: excess mortality with progressive disease
tpEOL <- 0.15*0.5 #tpEOL: excess mortality with progressive disease under end-of-life care
tpProg <- 0.01#tpProg: transition probability from asymptomatic to progressive disease
tpDn <- 0.0379  #pDn: baseline mortality <- mortality 0.0379  # over 65 year old

effect <- 0.5#effect: drug efficacy in decreasing the risk of progressing from asymptomatic to
progressive disease

```
~~~

2. Prepare cost, qaly, and transitional cost matrix. Note that the parameters defined in the previous block are used/called here
cost and qaly matrix has the following structure
  row: strategies
  column: health states
  value: cost, qaly payoffs

Transitional cost matrix has the following structure
  row: health states (departure state)
  column: health states(arrival state)
  value: transitional cost(toll)

```{r}
# cost of staying in state
state_c_matrix <-
  matrix(c(cAsymp, cProg, 0,
           cAsymp + cDrug, cProg, 0,
           cAsymp + cDrug, cProg + cEOL, 0),
         byrow = TRUE,
         nrow = n_treatments,
         dimnames = list(t_names,
                         s_names))
state_c_matrix
# qaly when staying in state
state_q_matrix <-
  matrix(c(uAsymp, uProg, 0,
           uAsymp, uProg, 0,
           uAsymp, uProg, 0),
         byrow = TRUE,
         nrow = n_treatments,
         dimnames = list(t_names,
                         s_names))
state_q_matrix
# cost of moving to a state
# same for both treatments
trans_c_matrix <-
  matrix(c(0, 0, 0,
           0, 0, cDeath,
           0, 0, 0),
         byrow = TRUE,
         nrow = n_states,
         dimnames = list(from = s_names,
                         to = s_names))
trans_c_matrix
```

3. Prepare transition probability matrix
p_matrix is a time-dependent transition probability matrix when incorporating age-specific mortality, meaning that transition probability will depend on the age/cycle. Here we define p_matrix with a set of parameters.

Note: array is useful to create a multidimensional matrix.
Here p_matrix has 3-dimensional matrix with (health states x health states) x strategies

```{r}
# Transition probabilities ----

# time-homogeneous
p_matrix <- array(data = c(1 - tpProg - tpDn, 0, 0,
                           tpProg, 1 - tpDcm - tpDn, 0,
                           tpDn, tpDcm + tpDn, 1,
                           1 - tpProg*(1-effect) - tpDn, 0, 0,
                           tpProg*(1-effect), 1 - tpDcm - tpDn, 0,
                           tpDn, tpDcm + tpDn, 1,
                           1 - tpProg*(1-effect) - tpDn, 0, 0,
                           tpProg*(1-effect), 1 - tpEOL - tpDn, 0,
                           tpDn, tpEOL + tpDn, 1),
                  dim = c(n_states, n_states, n_treatments),
                  dimnames = list(from = s_names,
                                  to = s_names,
                                  t_names))
p_matrix
```

4. Prepare population matrix that keeps track of health state transition by cycle
'pop' matrix will record health state distribution in the population for each cycle by strategy
(3-dimensional matrix: (n_states x n_cycles) x n_treatments)

In cycle = 1, everyone is in the asymptomatic disease state. The third dimension is not specified to apply the same operation to both treatment strategies

```{r}
# Store population output for each cycle

# state populations
pop <- array(data = NA,
             dim = c(n_states, n_cycles, n_treatments),
             dimnames = list(state = s_names,
                             cycle = NULL,
                             treatment = t_names))

pop["Asymptomatic_disease", cycle = 1, ] <- n_cohort
pop["Progressive_disease", cycle = 1, ] <- 0
pop["Dead", cycle = 1, ] <- 0

head(pop)
```

~~~
```
~~~

5. Prepare a matrix ('trans') to record total transitional cost per cycle by state
'pop' matrix records total number of people in each health state, whereas trans records the number of people who 'enter' the state and the cost imposed to those who newly enter the state

dimension: n_states x n_cycles x n_treatments (here n_states indicate 'destination state')
For example, trans["asymptomatic", cycle=10, treatment='without drug'] indicates the "toll" or "transitional costs" imposed to those who arrived in the asymptomatic state at cycle 10 under the 'without drug' scenario

~~~
```{r}
# _arrived_ state populations
trans <- array(data = NA,
               dim = c(n_states, n_cycles, n_treatments),
               dimnames = list(state = s_names,
                               cycle = NULL,
                               treatment = t_names))

trans[, cycle = 1, ] <- 0
```
~~~

6. Prepare empty matrices to record QALYs and Costs outcomes by cycle and by strategy
by cycle: cycle_costs, cycle_QALYs, cycle_QALE, LE, LYs (dimension: n_treatments, cycles)
total: total_costs, total_QALYs (1xn_treatments)

~~~
```{r}
# Sum costs and QALYs for each cycle at a time for each drug
cycle_empty_array <-
  array(NA,
        dim = c(n_treatments, n_cycles),
        dimnames = list(treatment = t_names,
                        cycle = NULL)) # per-cycle outcome template matrix

cycle_empty_array

cycle_state_costs <- cycle_trans_costs <- cycle_empty_array
cycle_costs <- cycle_QALYs <- cycle_empty_array
LE <- LYs <- cycle_empty_array    # life expectancy; life-years
cycle_QALE <- cycle_empty_array   # quality-adjusted life expectancy

total_costs <- setNames(c(NA, NA, NA), t_names)
total_QALYs <- setNames(c(NA, NA, NA), t_names)
~~~

~~~
total_costs
```
~~~

7. Define a function to calculate time-dependent transition probability based on age-specific mortality
Because non-mortality transition probabilities can change based on mortality, we will update the transition probability matrix given age by calling the following function.

~~~
```{r}

# Time-dependent probability matrix ----

p_matrix_cycle <- function(p_matrix, age, cycle,
                           tpProg = 0.01,
                           tpDcm = 0.15,
                           effect = 0.5) {
  tpDn_lookup <-
    c("(34,44]" = 0.0017,
      "(44,54]" = 0.0044,
      "(54,64]" = 0.0138,
      "(64,74]" = 0.0379,
      "(74,84]" = 0.0912,
      "(84,100]" = 0.1958)

  age_grp <- cut(age, breaks = c(34,44,54,64,74,84,100)) # find the age group that this age falls
into
  tpDn <- tpDn_lookup[age_grp] # look up mortality table using age_grp label

  # Matrix containing transition probabilities for without_drug
  p_matrix["Asymptomatic_disease", "Progressive_disease", "without_drug"] <- tpProg*cycle
  p_matrix["Asymptomatic_disease", "Dead", "without_drug"] <- tpDn
  p_matrix["Asymptomatic_disease", "Asymptomatic_disease", "without_drug"] <- 1 -
tpProg*cycle - tpDn
  p_matrix["Progressive_disease", "Dead", "without_drug"] <- tpDcm + tpDn
  p_matrix["Progressive_disease", "Progressive_disease", "without_drug"] <- 1 - tpDcm - tpDn
  p_matrix["Dead", "Dead", "without_drug"] <- 1

  # Matrix containing transition probabilities for with_drug
  p_matrix["Asymptomatic_disease", "Progressive_disease", "with_drug"] <- tpProg*(1 -
effect)*cycle
  p_matrix["Asymptomatic_disease", "Dead", "with_drug"] <- tpDn
  p_matrix["Asymptomatic_disease", "Asymptomatic_disease", "with_drug"] <-
    1 - tpProg*(1 - effect)*cycle - tpDn
~~~

~~~
  p_matrix["Progressive_disease", "Dead", "with_drug"] <- tpDcm + tpDn
  p_matrix["Progressive_disease", "Progressive_disease", "with_drug"] <- 1 - tpDcm - tpDn
  p_matrix["Dead", "Dead", "with_drug"] <- 1

    # Matrix containing transition probabilities for with_drug_EOL
  p_matrix["Asymptomatic_disease", "Progressive_disease", "with_drug_EOL"] <- tpProg*(1 -
effect)*cycle
  p_matrix["Asymptomatic_disease", "Dead", "with_drug_EOL"] <- tpDn
  p_matrix["Asymptomatic_disease", "Asymptomatic_disease", "with_drug_EOL"] <-
    1 - tpProg*(1 - effect)*cycle - tpDn
  p_matrix["Progressive_disease", "Dead", "with_drug_EOL"] <- tpEOL + tpDn
  p_matrix["Progressive_disease", "Progressive_disease", "with_drug_EOL"] <- 1 - tpEOL - tpDn
  p_matrix["Dead", "Dead", "with_drug"] <- 1

  return(p_matrix)
}
```
~~~

8. Simulate a cohort for 'no drug' and 'drug' scenario
two for loops are implemented in this block
(1) given strategy (outer loop)
(2) given cycle # (inner loop)
  - calculate age-specific transition probability matrix
  - update pop matrix in the current cycle given pop matrix in last cycle and transition probability matrix (matrix multiplication)
  - calculate trans matrix to count the number of people move from one to another state within a cycle

A * B: element-wise multiplication
A %*% B: matrix multiplication (take the row of A, column of B and sum the product)

To calculate the transitional cost per cycle to arrive in each state,
For example, the transitional cost of arriving in "progressive disease (state #2)" is
N_1*P_12*C_12 + N_2*P_22*C_22 + N3*P_32*C_32
 = N_1*C_12*P_12 + N_2*C_22*P_22 + N3*C_32*P_32
 = c(N_1, N_2, N_3) %*% t(c(C_12*P_12, C_22*P_22, C_32*P_32))
 = c(N_1, N_2, N_3) %*% t(c(C_12,C_22,C_32) * c(P_12,P_22,P_32))

trans_c_matrix * p_matrix -> transition probability-weighted transitional cost
c(1,1,1) %*% column vector() -> sum of elements in the column vector

~~~
```{r}
## Run model ----
~~~

```
for (i in 1:n_treatments) { # outer loop: over strategies

  age <- Initial_age

  for (j in 2:n_cycles) { # inner loop: over cycles
    # update cycle and age specific transition probability matrix
    p_matrix <- p_matrix_cycle(p_matrix, age, j - 1)
    #print(p_matrix[,,'with_drug']) # Uncomment this line to see how transition probability matrix
changes over cycle
    # calculate population health state distribution in the next cycle
    pop[, cycle = j, treatment = i] <-
      pop[, cycle = j - 1, treatment = i] %*% p_matrix[, , treatment = i]
    # calculate the total transitional costs per cycle
    trans[, cycle = j, treatment = i] <-
      pop[, cycle = j - 1, treatment = i] %*% (trans_c_matrix * p_matrix[, , treatment = i])

    age <- age + 1
  }
  # calculate cycle-specific state costs given a treatment strategy
  cycle_state_costs[i, ] <-
    (state_c_matrix[treatment = i, ] %*% pop[, , treatment = i]) * 1/(1 + cDr)^(1:n_cycles - 1)

  # discounting at _previous_ cycle
  cycle_trans_costs[i, ] <-
    (c(1,1,1) %*% trans[, , treatment = i]) * 1/(1 + cDr)^(1:n_cycles - 2) # dot product with c(1,1,1)
sums transitional cost across states and generate per-cycle total transitional cost
  # per-cycle cost is the summ of state cost and transitional cost
  cycle_costs[i, ] <- cycle_state_costs[i, ] + cycle_trans_costs[i, ]

  # life expectancy
  LE[i, ] <- c(1,1,0) %*% pop[, , treatment = i]

  # life-years
  LYs[i, ] <- LE[i, ] * 1/(1 + oDr)^(1:n_cycles - 1)

  # quality-adjusted life expectancy
  cycle_QALE[i, ] <-
    state_q_matrix[treatment = i, ] %*% pop[, , treatment = i]

  # quality-adjusted life-years
  cycle_QALYs[i, ] <- cycle_QALE[i, ] * 1/(1 + oDr)^(1:n_cycles - 1)
  # calculate the total cost and qaly (sum of per-cycle costs and qalys) of each scenario
  total_costs[i] <- sum(cycle_costs[treatment = i, -1])
  total_QALYs[i] <- sum(cycle_QALYs[treatment = i, -1])
}

print(total_QALYs)
```

9. Incremental cost and qaly between two strategies
```{r}
## Plot results ----

# Incremental costs and QALYs of with_drug vs to without_drug
c_incr <- total_costs["with_drug"] - total_costs["without_drug"]
q_incr <- total_QALYs["with_drug"] - total_QALYs["without_drug"]
c_incr_eol <- total_costs["with_drug_EOL"] - total_costs["with_drug"]
q_incr_eol <- total_QALYs["with_drug_EOL"] - total_QALYs["with_drug"]
c_incr_eol_v_notx <- total_costs["with_drug_EOL"] - total_costs["without_drug"]
q_incr_eol_v_notx <- total_QALYs["with_drug_EOL"] - total_QALYs["without_drug"]

print(c_incr_eol)
```

10. Calculate ICER
```{r}
# Incremental cost-effectiveness ratio
ICER_tx_v_notx <- c_incr/q_incr
ICER_weol_v_tx <- c_incr_eol/q_incr_eol
ICER_weol_v_notx <- c_incr_eol_v_notx/q_incr_eol_v_notx

print(ICER_weol_v_notx)
print(ICER_tx_v_notx)
```

11. Plot
```{r}

wtp <- 20000
plot(x = q_incr_eol/n_cohort, y = c_incr_eol/n_cohort,
    xlim = c(0, 2),
    ylim = c(0, 15e3),
    pch = 16, cex = 1.5,
    xlab = "QALY difference",
    ylab = paste0("Cost difference (", enc2utf8("\u00A3"), ")"),
    frame.plot = FALSE)
abline(a = 0, b = wtp) # willingness-to-pay threshold
```

~~~
#png("figures/ceplane_point.png", width = 4, height = 4, units = "in", res = 640)
plot(x = q_incr/n_cohort, y = c_incr/n_cohort,
     xlim = c(0, 2),
     ylim = c(0, 15e3),
     pch = 16, cex = 1.5,
     xlab = "QALY difference",
     ylab = paste0("Cost difference (", enc2utf8("\u00A3"), ")"),
     frame.plot = FALSE)
abline(a = 0, b = wtp) # willingness-to-pay threshold
dev.off()
```
~~~

<Skip to PSA code>
You can also turn the inner-loop of the base-case analysis code into a function so that we can repeat the simulation by the sampled parameter set. Note that sim_pop function takes strategy (i) as a function argument. The function will generate two outcomes: 1) population state distribution table or pop and 2) transitional cost table or trans

~~~
```{r}


############################################
# replace the task with sim_pop()

# simulate state populations
sim_pop <- function(n_cycles, age,
                    trans_c_matrix,
                    p_matrix, pop, trans, i) {

  for (j in 2:n_cycles) {
    p_matrix <- p_matrix_cycle(p_matrix, age, j - 1)
    pop[, cycle = j, i] <-
      pop[, cycle = j - 1, i] %*% p_matrix[, , i]
    trans[, cycle = j, i] <-
      pop[, cycle = j - 1, i] %*% (trans_c_matrix * p_matrix[, , i])
    age <- age + 1
  }

  list(pop = pop[, , i],
       trans = trans[, , i])
}
```
~~~

Implement a for-loop over strategies

~~~
```{r}
for (i in 1:n_treatments) {

  # simulate state populations
  sim_res <-
    sim_pop(n_cycles, Initial_age,
            trans_c_matrix,
            p_matrix, pop, trans, i)

  trans[, , i] <- sim_res$trans
  pop[, , i] <- sim_res$pop

  cycle_state_costs[i, ] <-
    (state_c_matrix[treatment = i, ] %*% pop[, , treatment = i]) * 1/(1 + cDr)^(1:n_cycles - 1)

  # discounting at _previous_ cycle
  cycle_trans_costs[i, ] <-
    (c(1,1,1) %*% trans[, , treatment = i]) * 1/(1 + cDr)^(1:n_cycles - 2)

  cycle_costs[i, ] <- cycle_state_costs[i, ] + cycle_trans_costs[i, ]

  # life expectancy
  LE[i, ] <- c(1,1,0) %*% pop[, , treatment = i]

  # life-years
  LYs[i, ] <- LE[i, ] * 1/(1 + oDr)^(1:n_cycles - 1)

  # quality-adjusted life expectancy
  cycle_QALE[i, ] <-
    state_q_matrix[treatment = i, ] %*% pop[, , treatment = i]

  # quality-adjusted life-years
  cycle_QALYs[i, ] <- cycle_QALE[i, ] * 1/(1 + oDr)^(1:n_cycles - 1)

  total_costs[i] <- sum(cycle_costs[treatment = i, -1])
  total_QALYs[i] <- sum(cycle_QALYs[treatment = i, -1])


}
```
~~~

<PSA code starts here>

Note that how the fixed parameters were replaced with the function to sample random values. We take start_pop, p_matrix, state_c_matrix, trans_c_matrix, and state_q_matrix as function arguments because these will change with sampled value of parameters.

~~~
```{r}
##############################################
# Probability Sensitivity Analysis (PSA)

ce_markov <- function(start_pop, # initial population distribution
                      p_matrix, # transition probability matrix
                      state_c_matrix, # state-specific cost reward matrix
                      trans_c_matrix, # transitional cost matrix
                      state_q_matrix, # state-specific qaly reward matrix
                      n_cycles = 46,
                      init_age = 55,
                      s_names = NULL,
                      t_names = NULL) {

  n_states <- length(start_pop)
  n_treat <- dim(p_matrix)[3]
  # define population matrix (3D)
  pop <- array(data = NA,
               dim = c(n_states, n_cycles, n_treat),
               dimnames = list(state = s_names,
                               cycle = NULL,
                               treatment = t_names))
  # define transitional cost matrix (3D)
  trans <- array(data = NA,
                 dim = c(n_states, n_cycles, n_treat),
                 dimnames = list(state = s_names,
                                 cycle = NULL,
                                 treatment = t_names))
  # populate pop dist at cycle = 1 with start_pop
  for (i in 1:n_states) {
    pop[i, cycle = 1, ] <- start_pop[i]
  }

  cycle_empty_array <-
    array(NA,
          dim = c(n_treat, n_cycles),
          dimnames = list(treatment = t_names,
                          cycle = NULL))

  cycle_state_costs <- cycle_trans_costs <- cycle_empty_array
  cycle_costs <- cycle_QALYs <- cycle_empty_array
~~~

```r
LE <- LYs <- cycle_empty_array    # life expectancy; life-years
cycle_QALE <- cycle_empty_array   # quality-adjusted life expectancy

total_costs <- setNames(rep(NA, n_treat), t_names)
total_QALYs <- setNames(rep(NA, n_treat), t_names)

for (i in 1:n_treat) { #outer loop over strategy

  age <- init_age

  for (j in 2:n_cycles) { # inner loop over cycles

    # difference from point estimate case
    # pass in functions for random sample
    # rather than fixed values
    p_matrix <- p_matrix_cycle(p_matrix, age, j - 1,
                               tpProg = tpProg(),
                               tpDcm = tpDcm(),
                               effect = effect())

    # Matrix multiplication
    pop[, cycle = j, treatment = i] <-
      pop[, cycle = j - 1, treatment = i] %*% p_matrix[, , treatment = i]

    trans[, cycle = j, treatment = i] <-
      pop[, cycle = j - 1, treatment = i] %*% (trans_c_matrix * p_matrix[, , treatment = i])

    age <- age + 1
  }

  cycle_state_costs[i, ] <-
    (state_c_matrix[treatment = i, ] %*% pop[, , treatment = i]) * 1/(1 + cDr)^(1:n_cycles - 1)

  cycle_trans_costs[i, ] <-
    (c(1,1,1) %*% trans[, , treatment = i]) * 1/(1 + cDr)^(1:n_cycles - 2)

  cycle_costs[i, ] <- cycle_state_costs[i, ] + cycle_trans_costs[i, ]

  LE[i, ] <- c(1,1,0) %*% pop[, , treatment = i]

  LYs[i, ] <- LE[i, ] * 1/(1 + oDr)^(1:n_cycles - 1)

  cycle_QALE[i, ] <-
   state_q_matrix[treatment = i, ] %*%  pop[, , treatment = i]
```

~~~
    cycle_QALYs[i, ] <- cycle_QALE[i, ] * 1/(1 + oDr)^(1:n_cycles - 1)

    total_costs[i] <- sum(cycle_costs[treatment = i, -1])
    total_QALYs[i] <- sum(cycle_QALYs[treatment = i, -1])
  }

  list(pop = pop,
       cycle_costs = cycle_costs,
       cycle_QALYs = cycle_QALYs,
       total_costs = total_costs,
       total_QALYs = total_QALYs)
}
```
~~~

2. Random sampling of parameters

~~~
```{r}
# replace point values with functions to random sample
cAsymp <- function() rnorm(1, 500, 127.55)
cDeath <- function() rnorm(1, 1000, 255.11)
cDrug  <- function() rnorm(1, 1000, 102.04)
cProg  <- function() rnorm(1, 3000, 510.21)
effect <- function() rnorm(1, 0.5, 0.051)
tpDcm  <- function() rbeta(1, 29, 167)
tpProg <- function() rbeta(1, 15, 1506)
uAsymp <- function() rbeta(1, 69, 4)
uProg  <- function() rbeta(1, 24, 8)
```
~~~

3. per-cycle reward matrices (state_c_matrix, state_q_matrix) will also depend on the sampled parameter values. So we define the name of these matrices to be functions and call random sampling functions.

~~~
```{r}
# Define cost and QALYs as functions

state_c_matrix <- function() {
  matrix(c(cAsymp(), cProg(), 0,            # without drug
         cAsymp() + cDrug(), cProg(), 0), # with drug
         byrow = TRUE,
         nrow = n_treatments,
         dimnames = list(t_names,
                         s_names))
}
~~~

~~~~
state_q_matrix <- function() {
  matrix(c(uAsymp(), uProg(), 0,  # without drug
           uAsymp(), uProg(), 0), # with drug
         byrow = TRUE,
         nrow = n_treatments,
         dimnames = list(t_names,
                         s_names))
}

trans_c_matrix <- function() {
  matrix(c(0, 0, 0,         # Asymptomatic_disease
           0, 0, cDeath(),  # Progressive_disease
           0, 0, 0),        # Dead
         byrow = TRUE,
         nrow = n_states,
         dimnames = list(from = s_names,
                         to = s_names))
}

```
~~~~

4. Run PSA

~~~~
```{r}
## Run PSA analysis ----

n_trials <- 500

costs <- matrix(NA, nrow = n_trials, ncol = n_treatments,
                dimnames = list(NULL, t_names))
qalys <- matrix(NA, nrow = n_trials, ncol = n_treatments,
                dimnames = list(NULL, t_names))

for (i in 1:n_trials) {
  ce_res <- ce_markov(start_pop = c(n_cohort, 0, 0),
                      p_matrix,
                      state_c_matrix(),
                      trans_c_matrix(),
                      state_q_matrix())

  costs[i, ] <- ce_res$total_costs
  qalys[i, ] <- ce_res$total_QALYs
}
~~~~

~~~
```

5. Plot
```{r}
## Plot results ----

# incremental costs and QALYs of with_drug vs to without_drug
c_incr_psa <- costs[, "with_drug"] - costs[, "without_drug"]
q_incr_psa <- qalys[, "with_drug"] - qalys[, "without_drug"]

plot(x = q_incr_psa/n_cohort, y = c_incr_psa/n_cohort,
     xlim = c(0, 2),
     ylim = c(0, 15e3),
     pch = 16, cex = 1.2,
     col = "grey",
     xlab = "QALY difference",
     ylab = paste0("Cost difference (", enc2utf8("\u00A3"), ")"),
     frame.plot = FALSE)
abline(a = 0, b = wtp, lwd = 2) # Willingness-to-pay threshold
points(x = q_incr/n_cohort, y = c_incr/n_cohort,
       col = "red", pch = 16, cex = 1.5) # base-case point estimate

png("figures/ceplane_psa.png", width = 4, height = 4, units = "in", res = 640)
plot(x = q_incr_psa/n_cohort, y = c_incr_psa/n_cohort,
     xlim = c(0, 2),
     ylim = c(0, 15e3),
     pch = 16, cex = 1.2,
     col = "grey",
     xlab = "QALY difference",
     ylab = paste0("Cost difference (", enc2utf8("\u00A3"), ")"),
     frame.plot = FALSE)
abline(a = 0, b = wtp, lwd = 2) # Willingness-to-pay threshold
points(x = q_incr/n_cohort, y = c_incr/n_cohort,
       col = "red", pch = 16, cex = 1.5)
dev.off()


```
~~~