heemod TB population Markov model

Nathan Green (Imperial College London)

17/01/2020

Introduction

In this document we go through an example of TB progression in a cohort using the package `heemod`. See here for more about `heemod`: https://cran.r-project.org/web/packages/heemod/vignettes/d_non_homogeneous.html

Setup

First we attach the packages we'll need and load in the data.

```
library(heemod)
library(purrr)
library(dplyr)

# age-dependent probability of death, TB and QoL weighting
pdeath_QoL <-
  read.csv("data/pdeath_QoL.csv")
head(pdeath_QoL)
```

```
##   age    pDeath pDeath_TB QOL_weight
## 1  35 0.0008065     0.012       0.91
## 2  36 0.0008710     0.012       0.91
## 3  37 0.0009280     0.012       0.91
## 4  38 0.0010540     0.012       0.91
## 5  39 0.0011160     0.012       0.91
## 6  40 0.0012320     0.012       0.91
```

```
# probabilistic realisations of starting state probabilities
load(file = "data/init_states.RData")
head(init_states)
```

```
##   noLTBI completeTx incompleteTx      noTx activeTB dead
## 1  0.643 0.04429927  0.068075534 0.2446252        0    0
## 2  0.683 0.02576789  0.019090692 0.2721414        0    0
## 3  0.672 0.04943117  0.030844729 0.2477241        0    0
## 4  0.742 0.05223975  0.001835084 0.2039252        0    0
## 5  0.726 0.02743526  0.010678055 0.2358867        0    0
## 6  0.650 0.05570397  0.024181029 0.2701150        0    0
```

Next we define the model parameters. We assume that everyone start in the model at age 35. We hardcode the transition probabilities for TB progression, as well as the treatment cost and discounting rate.

We also use `heemod::look_up()` to take the associated probability of death and Quality of Life (QoL) measure for a given age.

```
param <- define_parameters(
  age_init = 34,                    # starting age
  age = age_init + markov_cycle,    # increment age annually

  # transition probabilities
  pReact_comp = 0.0006779,          # TB after completed LTBI treatment
  pReact_incomp = 0.0015301,        # TB after LTBI treatment dropout
  pReact = 0.0019369,               # TB after no treatment

  TB_cost = 4925.76,                # cost of TB treatment (Â£)
  d = 0.035,                        # annual discount factor

  # match prob death to age
  pdeath = look_up(data = pdeath_QoL,
                   value = "pDeath",
                   age = age),
  pdeathTB = look_up(data = pdeath_QoL,
                     value = "pDeath_TB",
                     age = age),

  # match QoL weight to age
  QoL = look_up(data = pdeath_QoL,
                value = "QOL_weight",
                age = age)
)
```

Lets create the named transition matrix. The `C` notation fills in the missing entry with the complement of the other state probabilities.

```
mat_trans <- define_transition(
  state_names = c(
    "noLTBI",
    "completeTx",
    "incompleteTx",
    "noTx",
    "activeTB",
    "dead"
  ),

  # from-to probability matrix
  # C represent complements
  C, 0, 0, 0, 0,             pdeath,
  0, C, 0, 0, pReact_comp,   pdeath,
  0, 0, C, 0, pReact_incomp, pdeath,
  0, 0, 0, C, pReact,        pdeath,
  C, 0, 0, 0, 0,             pdeathTB,
  0, 0, 0, 0, 0,             1
)
```

Define cost and utility values associated with each state. For this simple example the QALY loss and incurred cost are attributable to the `activeTB` state.

```
noLTBI <- define_state(
  cost = 0,
  utility = discount(QoL, d, first = TRUE)
)

completeTx <- define_state(
  cost = 0,
  utility = discount(QoL, d, first = TRUE)
)

incompleteTx <- define_state(
  cost = 0,
  utility = discount(QoL, d, first = TRUE)
)

noTx <- define_state(
  cost = 0,
  utility = discount(QoL, d, first = TRUE)
)

activeTB <- define_state(
  cost = discount(TB_cost, d, first = TRUE),
  utility = discount(QoL - 0.15, d, first = TRUE)
)

dead <- define_state(
  cost = 0,
  utility = 0
)
```

Combine all of the model elements to form a 'strategy' consisting of a transition matrix and states with properties attached.

```
strat_6mo <- define_strategy(
  transition = mat_trans,
  noLTBI = noLTBI,
  completeTx = completeTx,
  incompleteTx = incompleteTx,
  noTx = noTx,
  activeTB = activeTB,
  dead = dead
)
```

For an initial population sensitivity analysis, define starting state populations from the data.

```
init_states <- select(.data = init_states,
                      noLTBI,
                      completeTx,
                      incompleteTx,
                      noTx)

init_states <- data.frame(init_states, activeTB = 0, dead = 0)
```

Now we are ready to run a single simulation.

```
res_mod <-
  run_model(
    init = 1000 * init_states[1, ], # initial population sizes
    method = "end",
    strat_6mo,
    parameters = param,
    cycles = 66,                    # number of time steps
    cost = cost,
    effect = utility
  )
```

```
## No named model -> generating names.
```

Note that transitions happen at the beginning of each year (equivalent to transition happening at the end ignoring the first year) with `method = "beginning"`. Since with this method the first year is actually the second, costs should be discounted from the start with the argument `first = TRUE` in `discount()`.

Next run multiple simulations using the sample of starting state probabilities.

```
res_mod <- list()

for (i in 1:nrow(init_states)) {

  res_mod[[i]] <-
    run_model(
      # init = c(674.0588764, # hard-code values
      #          168.0253748,
      #          42.42724895,
      #          115.4884998,
      #          0,0),
      init = 1000 * init_states[i, ],
      method = "end",
      strat_6mo,
      parameters = param,
      cycles = 66,
      cost = cost,
      effect = utility
    )
}
```

Results

Extract the cost and utility values using functions from the 'purrr' package. This is an excellent package for manipulating lists.

```
res_mod[[1]]
```

```
## 1 strategy run for 66 cycles.
##
## Initial state counts:
```

```
##
## noLTBI = 643
## completeTx = 44.2992721299233
## incompleteTx = 68.0755335064717
## noTx = 244.625194363605
## activeTB = 0
## dead = 0
##
## Counting method: 'end'.
##
## Values:
##
##      cost  utility
## I 62739.3 18900.67
```

```
c1 <- map_df(res_mod, "run_model")$cost
h1 <- map_df(res_mod, "run_model")$utility

get_counts(res_mod[[1]]) %>% head()
```

```
## # A tibble: 6 x 4
##   .strategy_names markov_cycle state_names count
##   <chr>                  <int> <chr>       <dbl>
## 1 I                          1 noLTBI       643
## 2 I                          2 noLTBI       642.
## 3 I                          3 noLTBI       643.
## 4 I                          4 noLTBI       643.
## 5 I                          5 noLTBI       642.
## 6 I                          6 noLTBI       642.
```

```
get_values(res_mod[[1]]) %>% head()
```

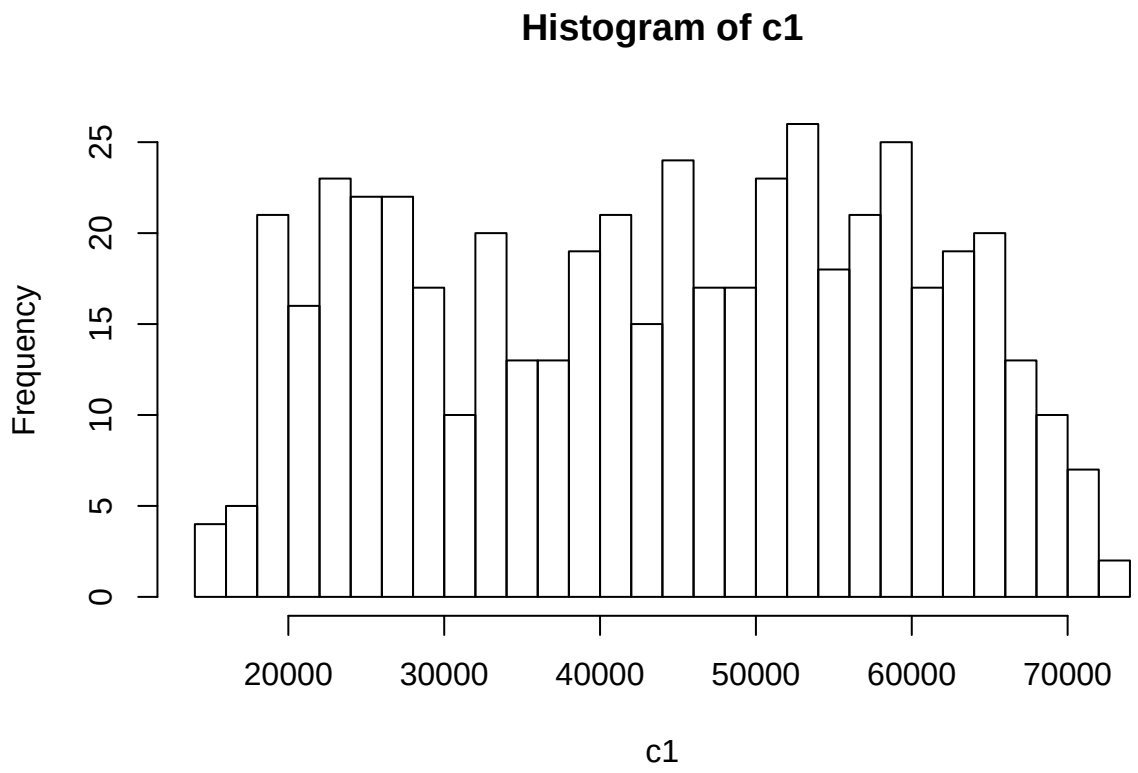
```
##   markov_cycle .strategy_names value_names    value
## 1            1               I        cost    0.000
## 2            2               I        cost 2795.770
## 3            3               I        cost 2694.173
## 4            4               I        cost 2596.100
## 5            5               I        cost 2501.454
## 6            6               I        cost 2409.955
```

```
summary(res_mod[[4]])
```

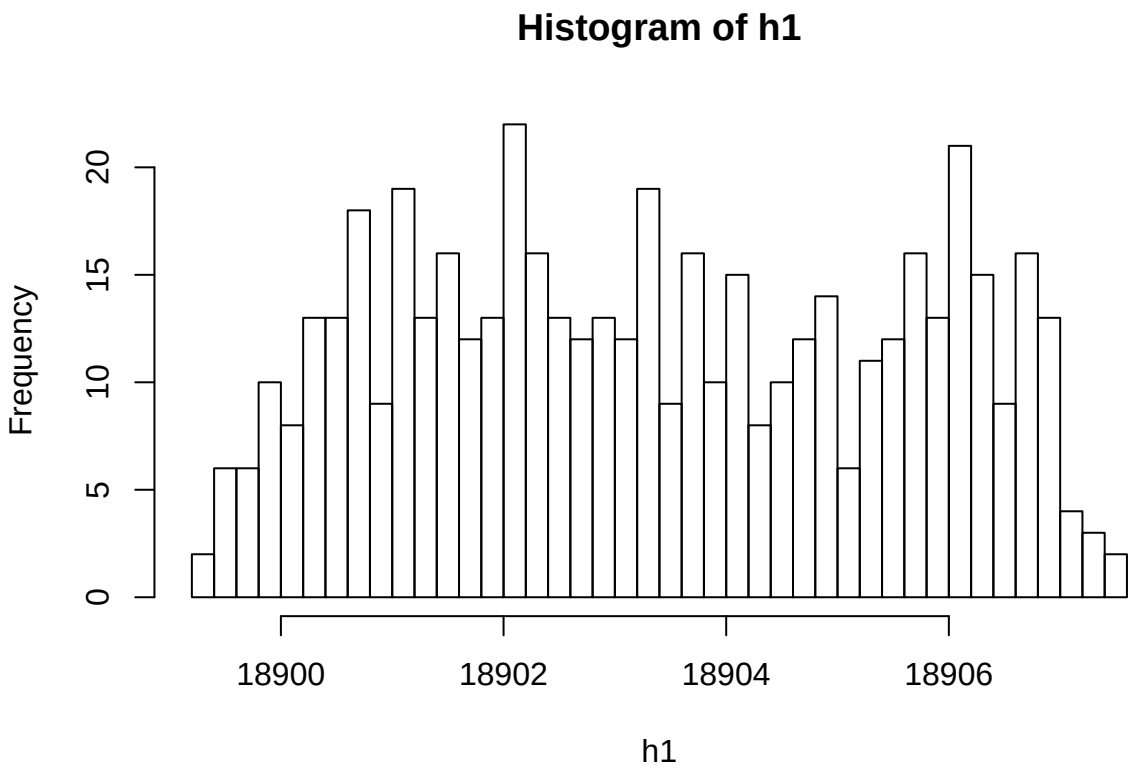
```
## 1 strategy run for 66 cycles.
##
## Initial state counts:
##
## noLTBI = 742
## completeTx = 52.2397525531171
## incompleteTx = 1.83508364108649
## noTx = 203.925163805796
## activeTB = 0
## dead = 0
```

```
##
## Counting method: 'end'.
##
## Values:
##
##    cost  utility
## I 44682 18903.23
```

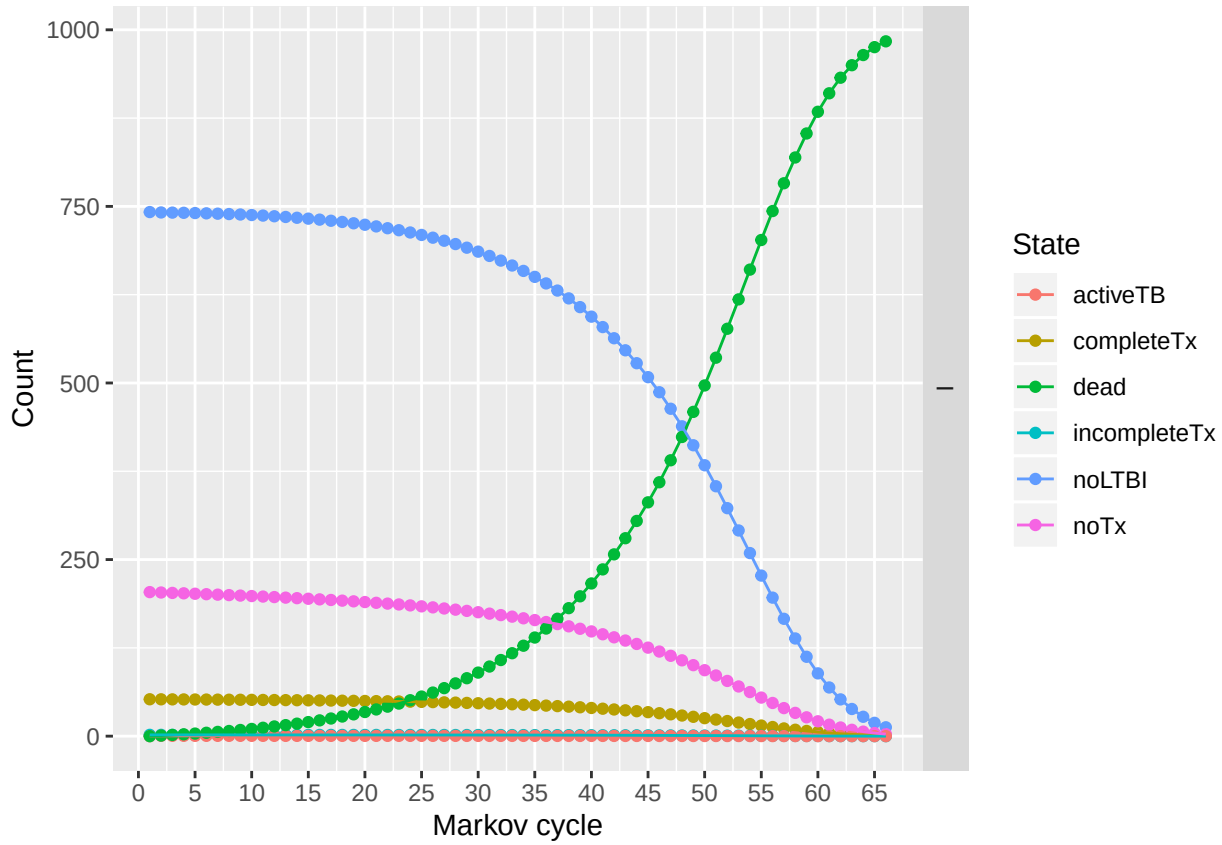
```r
# plots
hist(c1, breaks = 30)
```




```r
hist(h1, breaks = 30)
```

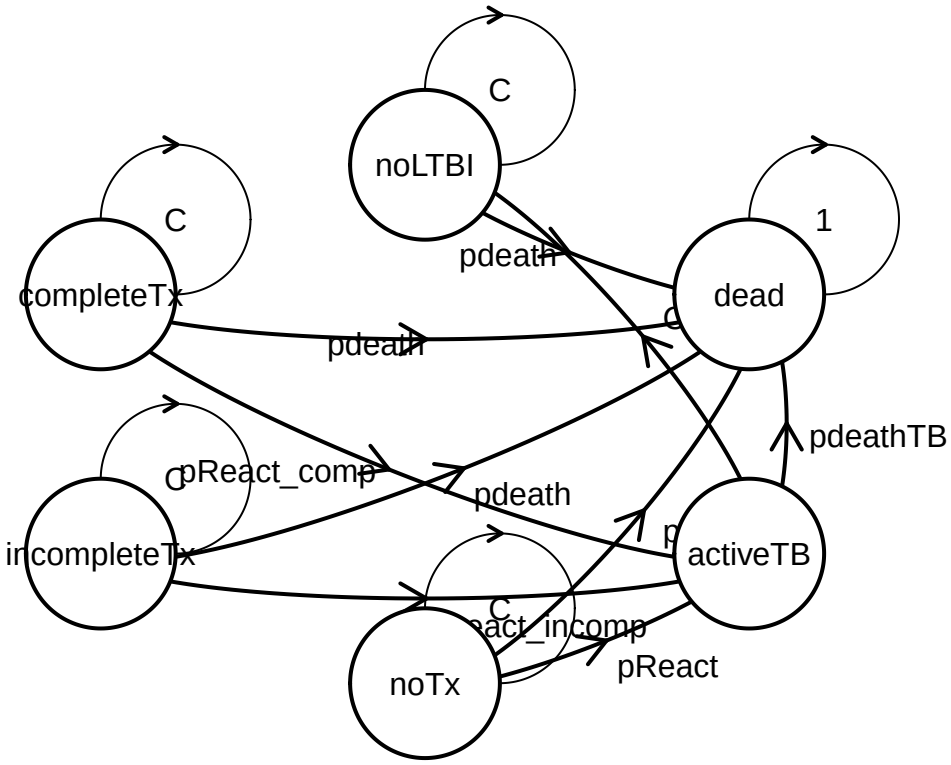



```
plot(res_mod[[4]])
```




```
# state-edge graph
plot(mat_trans, arr.type = "simple")
```

```
## Loading required namespace: diagram
```



Cost-utility PSA

Define a second itnervention. Lets assume that a 3 month active TB treatment cost half as much but has three times the health impact.

```
activeTB_3mo <- define_state(
  cost = discount(TB_cost/2, d, first = TRUE),
  utility = discount(QoL - 0.15*3, d, first = TRUE)
)
```

This new state is then used as part of a new 'strategy'.

```
strat_3mo <- define_strategy(
  transition = mat_trans,
  noLTBI = noLTBI,
  completeTx = completeTx,
  incompleteTx = incompleteTx,
  noTx = noTx,
  activeTB = activeTB_3mo,
  dead = dead
)
```

We repeat running `run_model()` but with two strategies this time (and a single starting population).

```
mod_psa <-
  run_model(
    init = 100 * init_states[1, ],
    method = "end",
    strat_6mo = strat_6mo,
    strat_3mo = strat_3mo,
    parameters = param,
    cycles = 66,
    cost = cost,
    effect = utility
  )
```

`heemod` has in-built functions to perform PSA. We specify the distributions in the `define_psa()` definition.

```
rsp <- define_psa(
  TB_cost ~ gamma(mean = 4925.76, sd = sqrt(4925.76)),
  pReact_comp ~ binomial(prob = 0.14, size = 10),
  pReact_incomp ~ binomial(prob = 0.15301, size = 10),
  pReact ~ binomial(prob = 0.09369, size = 10)
)
```

Finally we run the PSA.

```
pm <- run_psa(
  model = mod_psa,
  psa = rsp,
  N = 200
)
```

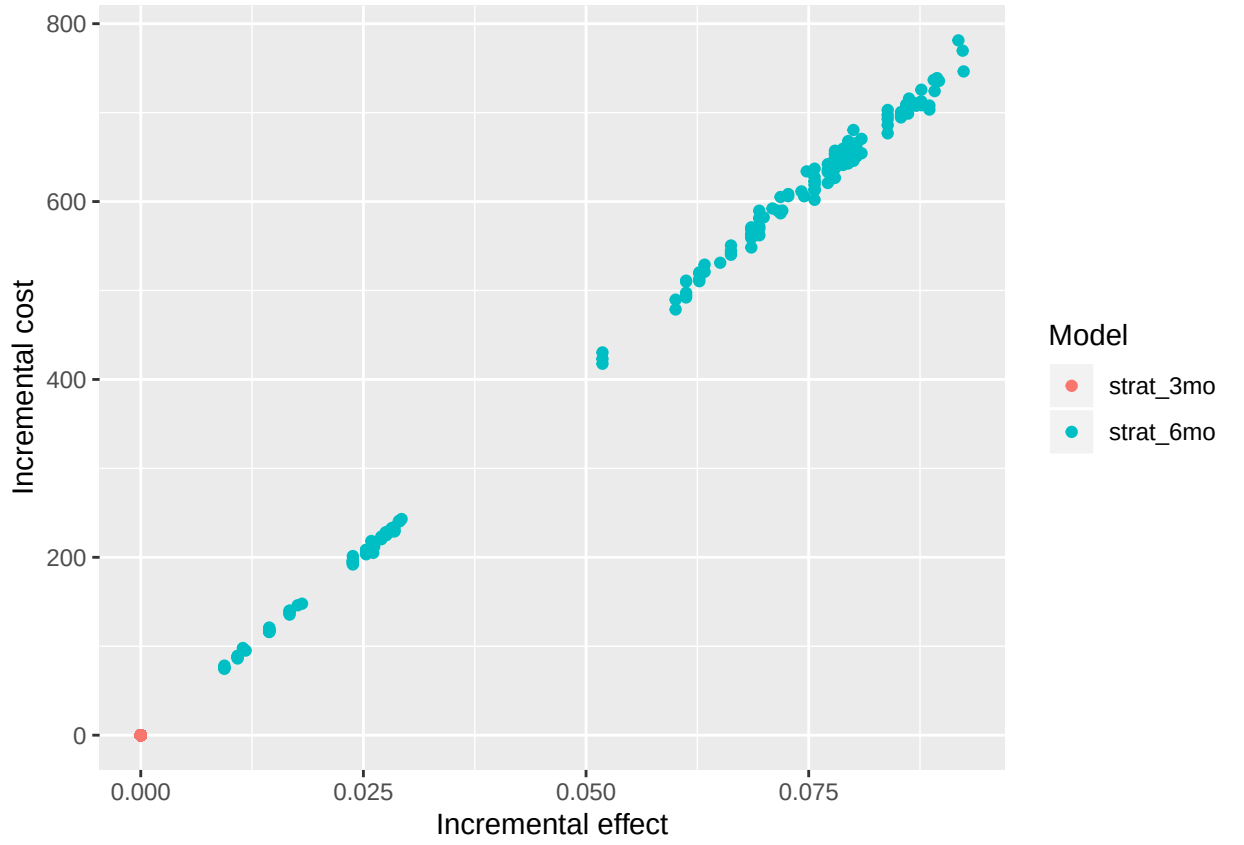
```
## Resampling strategy 'strat_6mo'...

## Resampling strategy 'strat_3mo'...
```

```
summary(pm)
```

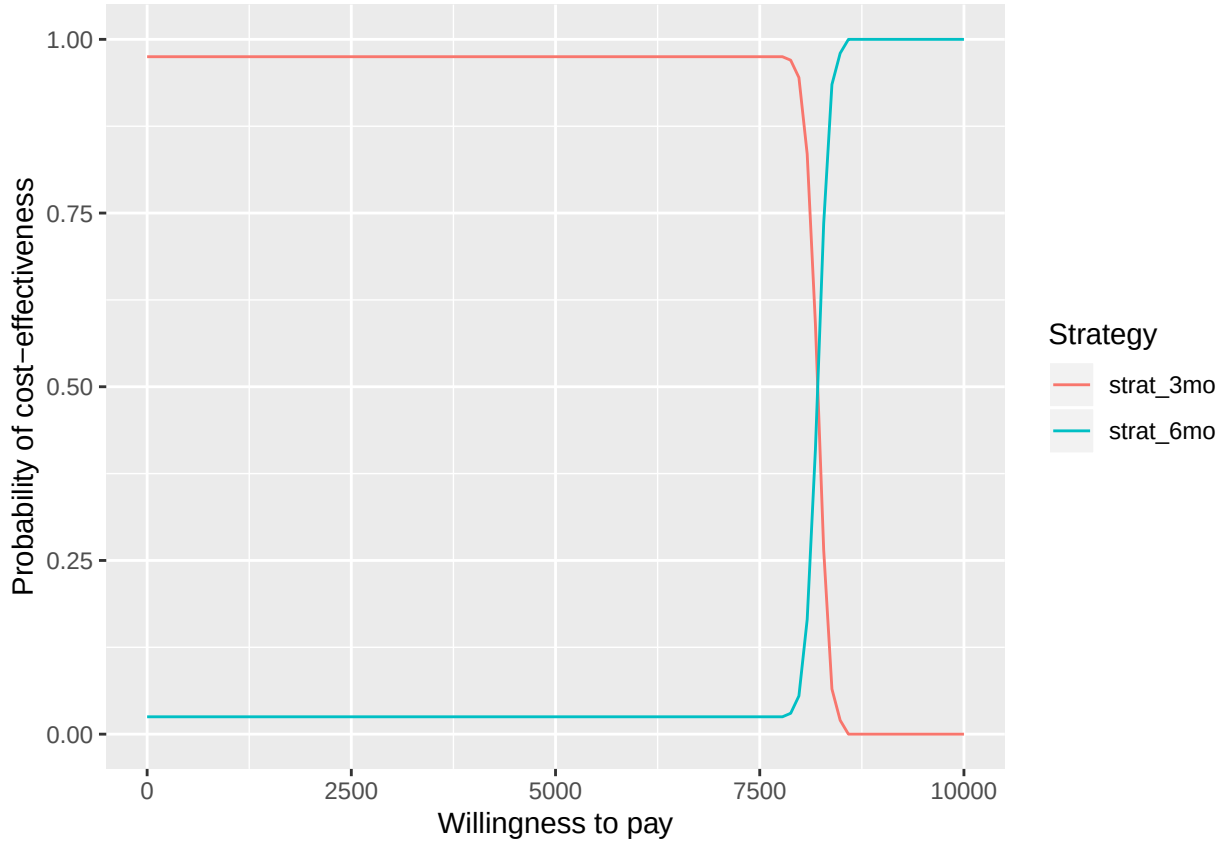
```
## 2 strategies run for 66 cycles.
##
## Initial state counts:
##
## noLTBI = 64.3
## completeTx = 4.42992721299233
## incompleteTx = 6.80755335064717
## noTx = 24.4625194363605
## activeTB = 0
## dead = 0
##
## Counting method: 'end'.
##
## Values:
##
##              cost  utility
```

```
## strat_3mo 45020.4 1877.607
## strat_6mo 90040.8 1883.085
##
## Efficiency frontier:
##
## strat_3mo -> strat_6mo
##
## Differences:
##
##           Cost Diff. Effect Diff.     ICER      Ref.
## strat_6mo    450.204   0.05478248 8218.028 strat_3mo
```

```
plot(pm, type = "ce")
```



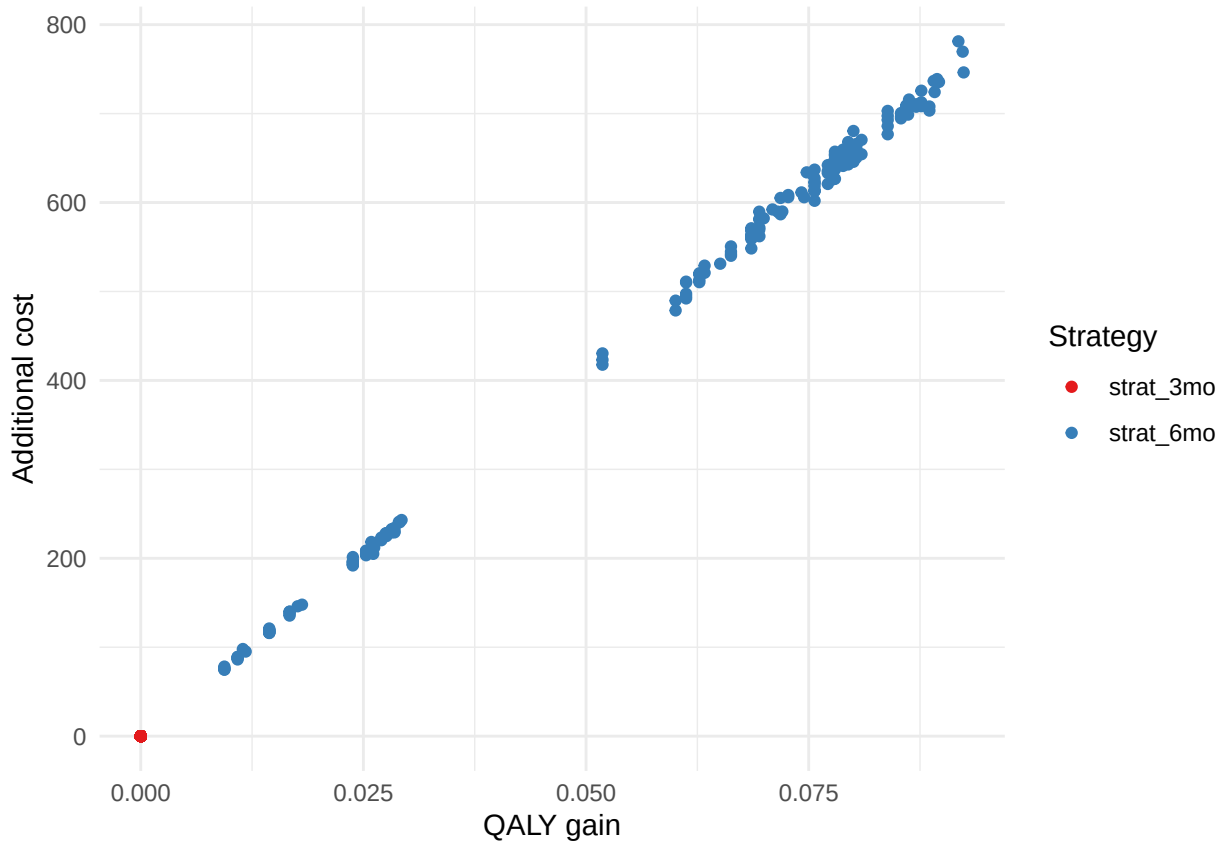

```
plot(pm, type = "ac", max_wtp = 10000, log_scale = FALSE)
```



We can alternatively use `ggplot` to make the figures.

```
library(ggplot2)

plot(pm, type = "ce") +
  xlab("QALY gain") +
  ylab("Additional cost") +
  scale_color_brewer(
    name = "Strategy",
    palette = "Set1"
  ) +
  theme_minimal()
```

```
## Scale for 'colour' is already present. Adding another scale for
## 'colour', which will replace the existing scale.
```


800
600
400
200
0
Additional cost
0.000
0.025
0.050
0.075
QALY gain
Strategy
strat_3mo
strat_6mo