Dual test (TST T-SPOT.TB) decision tree

Nathan Green, Imperial College London

11/12/2019

We will replicate the Excel TB TST-T-SPOT.TB cost-effectiveness model. (File name `latesttree.xls`.)

Set-up

```
library(readr)
library(dplyr)
library(tibble)
library(reshape2)
library(treeSimR)
library(assertthat)
library(CEdecisiontree)
library(purrr)
library(treeSimR)
```

Load in the data.

```
load(here::here("data", "params.RData"))
load(here::here("data", "trees.RData"))
```

This included the tree structure variables, the cost and probability arrays, the mapping arrays that inform which nodes have which label with have what cost.

To demonstrate, let us look at the TST and T-SPOT scenario. The decision tree is defined in terms of parents and children in a list.

```
head(TST_TSPOT_tree, 5)
```

```
## $`1`
## [1]  2 41
##
## $`2`
## [1]  3 38
##
## $`3`
## [1]  4 33
##
## $`4`
## [1]  5 30
##
## $`5`
## [1]  6 25
```

In order to assign values to the tree we first transform this to a (sparse) matrix format.

```
# probs <- child_list_to_transmat(TST_tree)
# probs <- child_list_to_transmat(QFT_tree)
# probs <- child_list_to_transmat(TSPOT_tree)
# probs <- child_list_to_transmat(TST_QFT_tree)
probs <- child_list_to_transmat(TST_TSPOT_tree)
head(probs)
```

```
##    1   2   3   4   5   6   7  8  9 10 11 12 13 14 15  16 17 18 19 20 21 22 23
## 1 NA 0.5  NA  NA  NA  NA  NA NA NA NA NA NA NA NA NA  NA NA NA NA NA NA NA NA
## 2 NA  NA 0.5  NA  NA  NA  NA NA NA NA NA NA NA NA NA  NA NA NA NA NA NA NA NA
## 3 NA  NA  NA 0.5  NA  NA  NA NA NA NA NA NA NA NA NA  NA NA NA NA NA NA NA NA
## 4 NA  NA  NA  NA 0.5  NA  NA NA NA NA NA NA NA NA NA  NA NA NA NA NA NA NA NA
## 5 NA  NA  NA  NA  NA 0.5  NA NA NA NA NA NA NA NA NA  NA NA NA NA NA NA NA NA
## 6 NA  NA  NA  NA  NA  NA 0.5 NA NA NA NA NA NA NA NA 0.5 NA NA NA NA NA NA NA
##   24  25 26 27 28 29  30 31 32  33 34 35 36 37  38 39 40  41 42 43
## 1 NA  NA NA NA NA NA  NA NA NA  NA NA NA NA NA  NA NA NA 0.5 NA NA
## 2 NA  NA NA NA NA NA  NA NA NA  NA NA NA NA NA 0.5 NA NA  NA NA NA
## 3 NA  NA NA NA NA NA  NA NA NA 0.5 NA NA NA NA  NA NA NA  NA NA NA
## 4 NA  NA NA NA NA NA 0.5 NA NA  NA NA NA NA NA  NA NA NA  NA NA NA
## 5 NA 0.5 NA NA NA NA  NA NA NA  NA NA NA NA NA  NA NA NA  NA NA NA
## 6 NA  NA NA NA NA NA  NA NA NA  NA NA NA NA NA  NA NA NA  NA NA NA
```

```
empty_transmat <-
  as_tibble(matrix(NA_real_,
                   nrow = nrow(probs), ncol = ncol(probs)),
            .name_repair = "minimal")
```

Next, we specify the labels for each of the edge (or correspondingly to node). We need to do this separately for the probabilities and costs.

```
# pname_from_to <- TST_pname_from_to
# pname_from_to <- QFT_pname_from_to
# pname_from_to <- TSPOT_pname_from_to
# pname_from_to <- TST_QFT_pname_from_to
pname_from_to <- TST_TSPOT_pname_from_to
pname_from_to
```

```
##                  name from to
## 1        pAccept_TST    1  2
## 2           pTSTread    2  3
## 3            TST_pos    3  4
## 4  pAccept_IGRA_TST+    4  5
## 5     TSPOT_pos_TST+    5  6
## 6     PPV_TSPOT_TST+    6  7
## 7      pAccept_chemo    7  8
## 8               pHep    8  9
## 9        pComp_chemo   11 12
## 10     pAccept_chemo   16 17
## 11              pHep   17 18
## 12       pComp_chemo   20 21
## 13    NPV_TSPOT_TST+   25 28
## 14           PPV_TST   30 31
## 15           NPV_TST   33 36
## 16             pLTBI   38 39
## 17             pLTBI   41 42
```

```
# cname_from_to <- TST_cname_from_to
# cname_from_to <- QFT_cname_from_to
# cname_from_to <- TSPOT_cname_from_to
# cname_from_to <- TST_QFT_cname_from_to
cname_from_to <- TST_TSPOT_cname_from_to
cname_from_to
```

```
##                                name from to
## 1                               TST    1  2
## 2           TB special nurse visit    2  3
## 3                             TSPOT    4  5
## 4  Total Cost of positive screening    5  6
## 5                               Hep    8  9
## 6               Total (incomplete)    9 10
## 7                 Total (complete)   11 12
## 8               Total (incomplete)   11 13
## 9               Total (incomplete)   18 19
## 10                Total (complete)   20 21
## 11              Total (incomplete)   20 22
```

Insert probabilities into decision tree

Now that we've set-up the framework for the decision tree, we can assign the input data to it. The probability data are in the form of a list (this is useful for when we want to sample from a distribution later). Lets transform to an array.

```
label_probs_long <-
  as_tibble(label_probs) %>%
  melt(value.name = "prob",
       variable.name = "name")
```

Insert the appropriate probabilities by converting the transition matrix to long format, matching branches to labels, matching labels to probabilities and then filling in missing probabilities so that pairs of branches sum to one.

```
probs_new <-
  probs %>%
  transmat_to_long() %>%
  match_branch_to_label(pname_from_to) %>%
  match_branchlabel_to_prob(label_probs_long) %>%
  fill_complementary_probs()

probs_new
```

```
## # A tibble: 42 x 4
##     from    to name           prob
##    <dbl> <dbl> <fct>         <dbl>
##  1     1     2 pAccept_TST   0.982
##  2     1    41 <NA>          0.018
##  3    11    12 pComp_chemo   0.8
##  4    11    13 <NA>          0.200
##  5    14    15 <NA>          1
##  6    16    17 pAccept_chemo 0.95
##  7    16    23 <NA>          0.05
##  8    17    18 pHep          0.002
##  9    17    20 <NA>          0.998
```

```
## 10    18    19 <NA>          1
## # ... with 32 more rows
```

Finally, we insert these new probabilities in to the decision tree.

```
probs <- insert_to_probmat(dat = probs_new,
                           mat = empty_transmat)
head(probs)
```

```
## # A tibble: 6 x 43
##      ``     ``     ``     ``     ``     ``     ``     ``     ``     ``     ``     ``
##   <dbl>  <dbl>  <dbl>  <dbl>  <dbl> <dbl>  <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1    NA  0.982 NA     NA     NA     NA    NA        NA    NA    NA    NA    NA
## 2    NA NA      0.979 NA     NA     NA    NA        NA    NA    NA    NA    NA
## 3    NA NA     NA      0.534 NA     NA    NA        NA    NA    NA    NA    NA
## 4    NA NA     NA     NA      0.995 NA    NA        NA    NA    NA    NA    NA
## 5    NA NA     NA     NA     NA      0.46 NA        NA    NA    NA    NA    NA
## 6    NA NA     NA     NA     NA     NA     0.944    NA    NA    NA    NA    NA
## # ... with 31 more variables: `` <dbl>, `` <dbl>, `` <dbl>, `` <dbl>, `` <dbl>,
## #   `` <dbl>, `` <dbl>, `` <dbl>, `` <dbl>, `` <dbl>, `` <dbl>, `` <dbl>,
## #   `` <dbl>, `` <dbl>, `` <dbl>, `` <dbl>, `` <dbl>, `` <dbl>, `` <dbl>,
## #   `` <dbl>, `` <dbl>, `` <dbl>, `` <dbl>, `` <dbl>, `` <dbl>, `` <dbl>,
## #   `` <dbl>, `` <dbl>, `` <dbl>, `` <dbl>, `` <dbl>
```

Insert costs into decision tree

We essentially do the same thing now for costs. Convert from list to dataframe.

```
label_cost_long <-
  as_tibble(label_costs) %>%
  melt(value.name = "cost",
       variable.name = "name")

head(label_cost_long)
```

```
##                                                 name    cost
## 1                          Contact tracing per contact  368.90
## 2     Mean number of contacts examined per primary case    6.50
## 3                                Total Contact tracing 2397.85
## 4               Cost of inpatient episode for acute TB 3325.15
## 5 Proportion of patients with acute TB who are admitted    0.53
## 6                                 Total Inpatient care 1762.33
```

Join the cost names and their associated branches in to a single array.

```
costs_names <-
  merge(cname_from_to, label_cost_long,
        by = "name", all.x = TRUE) %>%
  mutate(from = as.numeric(as.character(from)),
         to = as.numeric(as.character(to)))
costs_names
```

```
##                              name from to   cost
## 1                             Hep    8  9 732.13
## 2           TB special nurse visit    2  3  44.31
## 3                 Total (complete)   20 21 169.68
## 4                 Total (complete)   11 12 169.68
## 5               Total (incomplete)    9 10  84.84
```

```
## 6                Total (incomplete)   20 22  84.84
## 7                Total (incomplete)   11 13  84.84
## 8                Total (incomplete)   18 19  84.84
## 9  Total Cost of positive screening    5  6 241.23
## 10                            TSPOT    4  5  35.12
## 11                              TST    1  2  18.62
```

Finally, we insert these costs in to the decision tree.

```
costs <- insert_to_costmat(dat = costs_names,
                           mat = empty_transmat)
head(costs)
```

```
## # A tibble: 6 x 43
##      ``    ``    ``    ``    ``    ``    ``    ``    ``    ``    ``    ``    ``
##   <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1    NA  18.6  NA      NA  NA     NA     NA    NA    NA    NA    NA    NA    NA
## 2    NA  NA    44.3    NA  NA     NA     NA    NA    NA    NA    NA    NA    NA
## 3    NA  NA    NA      NA  NA     NA     NA    NA    NA    NA    NA    NA    NA
## 4    NA  NA    NA      NA  35.1   NA     NA    NA    NA    NA    NA    NA    NA
## 5    NA  NA    NA      NA  NA    241.    NA    NA    NA    NA    NA    NA    NA
## 6    NA  NA    NA      NA  NA     NA     NA    NA    NA    NA    NA    NA    NA
## # ... with 30 more variables: `` <dbl>, `` <dbl>, `` <dbl>, `` <dbl>, `` <dbl>,
## #   `` <dbl>, `` <dbl>, `` <dbl>, `` <dbl>, `` <dbl>, `` <dbl>, `` <dbl>,
## #   `` <dbl>, `` <dbl>, `` <dbl>, `` <dbl>, `` <dbl>, `` <dbl>, `` <dbl>,
## #   `` <dbl>, `` <dbl>, `` <dbl>, `` <dbl>, `` <dbl>, `` <dbl>, `` <dbl>,
## #   `` <dbl>, `` <dbl>, `` <dbl>, `` <dbl>
```

Run model

See the `CEdecisiontree` package for how to use the `dectree_expected_value()` function. Here we provide the matrix format arguments.

```
TST_model <-
  define_model(transmat = list(prob = probs,
                               vals = costs))

res <-
  dectree_expected_values(TST_model)

res[1] + label_costs$`TB special nurse visit`
```

```
##
## 214.1826
```

```
# 214.054 in xls
```

Deterministic sensitivity analysis Simply repeating the same set of values, we can demonstrate running multiple tree calculations. We show how to use the long format to specify the tree as the input argument to `dectree_expected_value()`.

```
# list of deterministic scenarios

# create combined tree long format dataframe
all_long <-
  merge(costs_names, probs_new,
        all = TRUE, by = c("from", "to"),
```

```
        suffixes = c(".cost", ".prob")) %>%
  rename(vals = cost)

head(all_long)
```

```
##   from to              name.cost  vals   name.prob  prob
## 1    1  2                    TST 18.62 pAccept_TST 0.982
## 2    1 41                   <NA>    NA        <NA> 0.018
## 3    2  3 TB special nurse visit 44.31    pTSTread 0.979
## 4    2 38                   <NA>    NA        <NA> 0.021
## 5    3  4                   <NA>    NA     TST_pos 0.534
## 6    3 33                   <NA>    NA        <NA> 0.466
```

```
dat <-
  all_long %>%
  select(-contains("name"))

dat <-
  list(dat,
       dat)

map(dat,
    function(x) dectree_expected_values(define_model(dat_long = x)))
```

```
## [[1]]
##        1        2        3        4        5        6        7        8
## 169.8726 172.9864 157.6776 212.2988 213.3657 387.4906 146.3385 154.0405
##        9       10       11       12       13       14       15       16
## 816.9700  84.8400 152.7120 169.6800  84.8400   0.0000   0.0000 144.9474
##       17       18       19       20       21       22       23       24
## 152.5763  84.8400  84.8400 152.7120 169.6800  84.8400   0.0000   0.0000
##       25       26       27       28       29       30       31       32
##   0.0000   0.0000   0.0000   0.0000   0.0000   0.0000   0.0000   0.0000
##       33       34       35       36       37       38       39       40
##   0.0000   0.0000   0.0000   0.0000   0.0000   0.0000   0.0000   0.0000
##       41       42       43
##   0.0000   0.0000   0.0000
##
## [[2]]
##        1        2        3        4        5        6        7        8
## 169.8726 172.9864 157.6776 212.2988 213.3657 387.4906 146.3385 154.0405
##        9       10       11       12       13       14       15       16
## 816.9700  84.8400 152.7120 169.6800  84.8400   0.0000   0.0000 144.9474
##       17       18       19       20       21       22       23       24
## 152.5763  84.8400  84.8400 152.7120 169.6800  84.8400   0.0000   0.0000
##       25       26       27       28       29       30       31       32
##   0.0000   0.0000   0.0000   0.0000   0.0000   0.0000   0.0000   0.0000
##       33       34       35       36       37       38       39       40
##   0.0000   0.0000   0.0000   0.0000   0.0000   0.0000   0.0000   0.0000
##       41       42       43
##   0.0000   0.0000   0.0000
```

Probabilistic sensitivity analysis We first define the decision tree. The difference to previous trees is that we now use the *list-column* feature to define distributions rather than point values.

For the *backwards* model i.e. with PPV and NPV:

```
costs_SA <-
  tibble::tribble(~name.cost,                ~vals,
                  "TST",                     list(distn = "unif", params = c(min=9.31,  max=37.24)), # P
                  "TB special nurse visit", list(distn = "unif", params = c(min=22.15, max=66.23)),
                  "TSPOT",                   list(distn = "unif", params = c(min=18.06, max=68.23)),   #
                  "Total Cost of positive screening",  list(distn = "unif", params = c(min=233.17, max=
                  "Hep",                     list(distn = "unif", params = c(min=366.06, max=1464.25)),
                  #
                  "Total (complete)",        list(distn = "unif", params = c(min=169.68, max=169.68)),
                  "Total (incomplete)",      list(distn = "unif", params = c(min=84.84,  max=84.84)))

probs_SA <-
  tibble::tribble(~name.prob,        ~prob,
                  "pAccept_chemo",  list(distn = "unif", params = c(min=0.5,   max=1)), # Pareek 2013
                  "pHep",           list(distn = "unif", params = c(min=0.001, max=0.003)), # Kunst, Pa
                  "pComp_chemo",    list(distn = "unif", params = c(min=0.5,   max=0.9)), # Kowada 2013
                  #
                  "pAccept_TST",    list(distn = "unif", params = c(min=0.5,   max=1)),         #Campbe
                  "pTSTread",       list(distn = "unif", params = c(min=0.979, max=0.979)),
                  "TST_pos",        list(distn = "unif", params = c(min=0.534, max=0.534)),
                  "pAccept_IGRA_TST+", list(distn = "unif", params = c(min=0.995, max=0.995)),

                  "pLTBI",          list(distn = "unif", params = c(min=0.326, max=0.326)),
                  "TSPOT_pos_TST+", list(distn = "unif", params = c(min=0.46,  max=0.46)),
                  "PPV_TSPOT_TST+", list(distn = "unif", params = c(min=0.944, max=0.944)),
                  "NPV_TSPOT_TST+", list(distn = "unif", params = c(min=0.911, max=0.911)),
                  "PPV_TST",        list(distn = "unif", params = c(min=0.482, max=0.482)),
                  "NPV_TST",        list(distn = "unif", params = c(min=0.853, max=0.853)))
probs_SA
```

```
## # A tibble: 13 x 2
##    name.prob         prob
##    <chr>             <list>
##  1 pAccept_chemo     <named list [2]>
##  2 pHep              <named list [2]>
##  3 pComp_chemo       <named list [2]>
##  4 pAccept_TST       <named list [2]>
##  5 pTSTread          <named list [2]>
##  6 TST_pos           <named list [2]>
##  7 pAccept_IGRA_TST+ <named list [2]>
##  8 pLTBI             <named list [2]>
##  9 TSPOT_pos_TST+    <named list [2]>
## 10 PPV_TSPOT_TST+    <named list [2]>
## 11 NPV_TSPOT_TST+    <named list [2]>
## 12 PPV_TST           <named list [2]>
## 13 NPV_TST           <named list [2]>
```

Combine the distributions with the original tree specification.

```
input_SA <-
  all_long %>%
  select(-prob, -vals) %>%
  dplyr::full_join(probs_SA, by = "name.prob") %>%
  dplyr::full_join(costs_SA, by = "name.cost") %>%
```

```
  as_tibble()

input_SA
```

```
## # A tibble: 42 x 6
##     from    to name.cost                name.prob      prob         vals
##    <dbl> <dbl> <chr>                    <chr>          <list>       <list>
##  1     1     2 TST                      pAccept_TST    <named list~ <named list~
##  2     1    41 <NA>                     <NA>           <NULL>       <NULL>
##  3     2     3 TB special nurse visit   pTSTread       <named list~ <named list~
##  4     2    38 <NA>                     <NA>           <NULL>       <NULL>
##  5     3     4 <NA>                     TST_pos        <named list~ <NULL>
##  6     3    33 <NA>                     <NA>           <NULL>       <NULL>
##  7     4     5 TSPOT                    pAccept_IGRA_~ <named list~ <named list~
##  8     4    30 <NA>                     <NA>           <NULL>       <NULL>
##  9     5     6 Total Cost of positive ~ TSPOT_pos_TST+ <named list~ <named list~
## 10     5    25 <NA>                     <NA>           <NULL>       <NULL>
## # ... with 32 more rows
```

We can now loop over this tree and generate samples of values for the given distributions. We could do this within the model but having a record of the inputs may be useful for reproducibility and testing. Use the `sample_distributions()` function from my `treeSimR` package.

```
tree_dat_sa <- list()

for (i in 1:400) {

  tree_dat_sa[[i]] <-
    data.frame(from = input_SA$from,
               to   = input_SA$to,
               prob = lapply(input_SA$prob, sample_distributions) %>% unlist(),
               vals = lapply(input_SA$vals, sample_distributions) %>% unlist()) %>%
    fill_complementary_probs() %>%
    define_model(dat_long =  .)
}
```

This results in a list of trees. Now it is straightforward to map over each of these trees to obtain the total expected values.

```
res <- map(tree_dat_sa, .f = dectree_expected_values)
head(res)
```

```
## [[1]]
##         1         2         3         4         5         6         7         8
##  114.3898  177.7557  166.0449  190.7801  191.7388  361.7451  129.5010  143.5644
##         9        10        11        12        13        14        15        16
## 1055.9509   84.8400  142.0206  169.6800   84.8400    0.0000    0.0000   69.3514
##        17        18        19        20        21        22        23        24
##  134.0299   84.8400   84.8400  134.1495  169.6800   84.8400    0.0000    0.0000
##        25        26        27        28        29        30        31        32
##    0.0000    0.0000    0.0000    0.0000    0.0000    0.0000    0.0000    0.0000
##        33        34        35        36        37        38        39        40
##    0.0000    0.0000    0.0000    0.0000    0.0000    0.0000    0.0000    0.0000
##        41        42        43
##    0.0000    0.0000    0.0000
##
```

```
## [[2]]
##         1         2         3         4         5         6         7         8
##   81.8264  156.4553  147.6610  200.3588  201.3656  380.7296  142.1984  148.2213
##         9        10        11        12        13        14        15        16
## 1425.5345   84.8400  145.0940  169.6800   84.8400    0.0000    0.0000  126.2746
##        17        18        19        20        21        22        23        24
##  128.0555   84.8400   84.8400  128.1431  169.6800   84.8400    0.0000    0.0000
##        25        26        27        28        29        30        31        32
##    0.0000    0.0000    0.0000    0.0000    0.0000    0.0000    0.0000    0.0000
##        33        34        35        36        37        38        39        40
##    0.0000    0.0000    0.0000    0.0000    0.0000    0.0000    0.0000    0.0000
##        41        42        43
##    0.0000    0.0000    0.0000
##
## [[3]]
##         1         2         3         4         5         6         7         8
##  91.61563 164.76054 152.28861 216.63209 217.72070 328.80593  81.67809 144.63227
##         9        10        11        12        13        14        15        16
## 966.44708  84.84000 143.57761 169.68000  84.84000   0.00000   0.00000  97.35357
##        17        18        19        20        21        22        23        24
## 159.08327  84.84000  84.84000 159.16810 169.68000  84.84000   0.00000   0.00000
##        25        26        27        28        29        30        31        32
##   0.00000   0.00000   0.00000   0.00000   0.00000   0.00000   0.00000   0.00000
##        33        34        35        36        37        38        39        40
##   0.00000   0.00000   0.00000   0.00000   0.00000   0.00000   0.00000   0.00000
##        41        42        43
##   0.00000   0.00000   0.00000
##
## [[4]]
##        1        2        3        4        5        6        7        8
## 147.2780 206.3782 175.2868 216.0177 217.1032 356.7701 123.9545 128.9888
##        9       10       11       12       13       14       15       16
## 674.7361  84.8400 128.0411 169.6800  84.8400   0.0000   0.0000 115.0300
##       17       18       19       20       21       22       23       24
## 149.8588  84.8400  84.8400 149.9378 169.6800  84.8400   0.0000   0.0000
##       25       26       27       28       29       30       31       32
##   0.0000   0.0000   0.0000   0.0000   0.0000   0.0000   0.0000   0.0000
##       33       34       35       36       37       38       39       40
##   0.0000   0.0000   0.0000   0.0000   0.0000   0.0000   0.0000   0.0000
##       41       42       43
##   0.0000   0.0000   0.0000
##
## [[5]]
##         1         2         3         4         5         6         7         8
##  188.1111  199.9870  171.5308  224.7836  225.9132  351.2031  107.6034  154.6990
##         9        10        11        12        13        14        15        16
## 1429.0303   84.8400  153.1076  169.6800   84.8400    0.0000    0.0000  126.0949
##        17        18        19        20        21        22        23        24
##  131.3960   84.8400   84.8400  131.5334  169.6800   84.8400    0.0000    0.0000
##        25        26        27        28        29        30        31        32
##    0.0000    0.0000    0.0000    0.0000    0.0000    0.0000    0.0000    0.0000
##        33        34        35        36        37        38        39        40
##    0.0000    0.0000    0.0000    0.0000    0.0000    0.0000    0.0000    0.0000
##        41        42        43
```

```
##    0.0000    0.0000    0.0000
##
## [[6]]
##         1         2         3         4         5         6         7
##  144.94839 208.68825 175.83026 206.47486 207.51242 335.47464  98.34748
##         8         9        10        11        12        13        14
##  134.62826 1259.97897  84.84000 131.53382 169.68000  84.84000   0.00000
##        15        16        17        18        19        20        21
##    0.00000 103.37395 127.17935  84.84000  84.84000 127.28525 169.68000
##        22        23        24        25        26        27        28
##   84.84000   0.00000   0.00000   0.00000   0.00000   0.00000   0.00000
##        29        30        31        32        33        34        35
##    0.00000   0.00000   0.00000   0.00000   0.00000   0.00000   0.00000
##        36        37        38        39        40        41        42
##    0.00000   0.00000   0.00000   0.00000   0.00000   0.00000   0.00000
##        43
##    0.00000
```

```r
hist(map_dbl(res, 1), breaks = 25, freq = FALSE, xlab = "cost")
lines(density(map_dbl(res, 1)), col = "red")
```