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 `lasttree.xls`.)

Set-up

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

Load in the data.

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

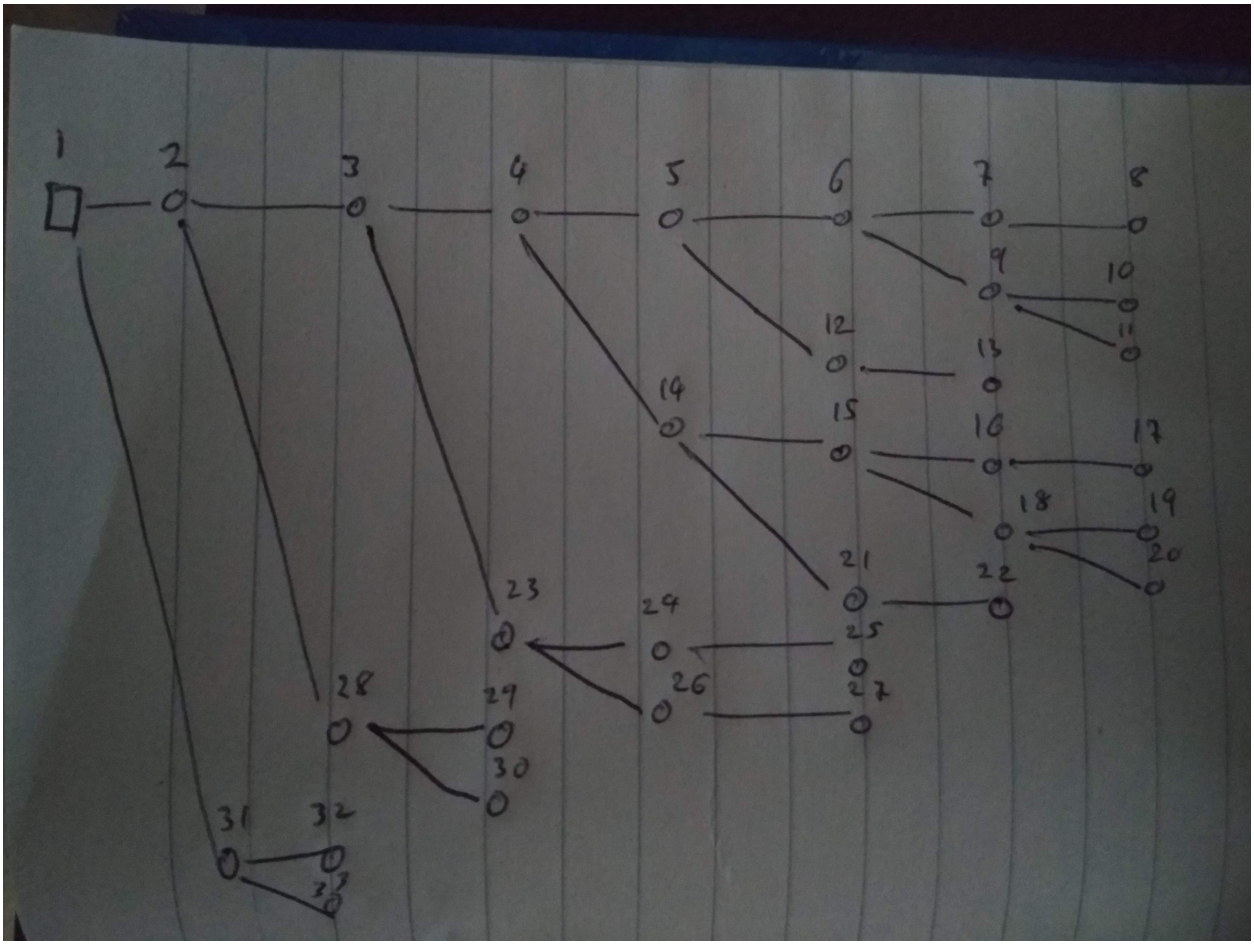
This included the tree structure variables, the cost and probability arrays, the mapping arrays that inform which nodes have which label with have what cost.

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
```

It looks like this for the single test decision tree



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
## 1 NA 0.5  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
## 3 NA  NA  NA 0.5  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
## 5 NA  NA  NA  NA  NA 0.5  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
##   23 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 NA 0.5 NA NA
## 2 NA 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 NA 0.5 NA NA NA NA  NA NA NA  NA NA NA
## 4 NA NA  NA NA NA NA NA 0.5 NA NA  NA NA NA NA NA  NA NA NA  NA NA NA
## 5 NA 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 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 is in the form of a list (this is useful for when we want to sample from a distributuion 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> <chr>         <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 <- CEdecisiontree:::insert_to_probmat(dat = probs_new,
                                            mat = empty_transmat)
```

Insert costs into decision tree

We essentially do the same thing now for costs.

```
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 <- CEdecisiontree:::insert_to_costmat(dat = costs_names,
                                            mat = empty_transmat)
```

Run model

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

```
res <-
  dectree_expected_values(vals = costs,
                          p = probs)


res[1] + label_costs$`TB special nurse visit` #44.31
```

```
##
## 214.1826
```

Deterministic sensitivity analysis

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

```
# list of deterministic scenarios

all_long <-
  merge(costs_names, probs_new,
        all = TRUE, by = c("from", "to")) %>%
  rename(vals = cost) %>%
  select(-contains("name"))

dat <-
  list(all_long,
       all_long)

# wrapper for sampling in SA
dectree_expected_values(define_model(dat_long = all_long))
```

```
##        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
```

```
lapply(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
```