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

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)
library(purrr)
library(treeSimR)
```

Load in the data.

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

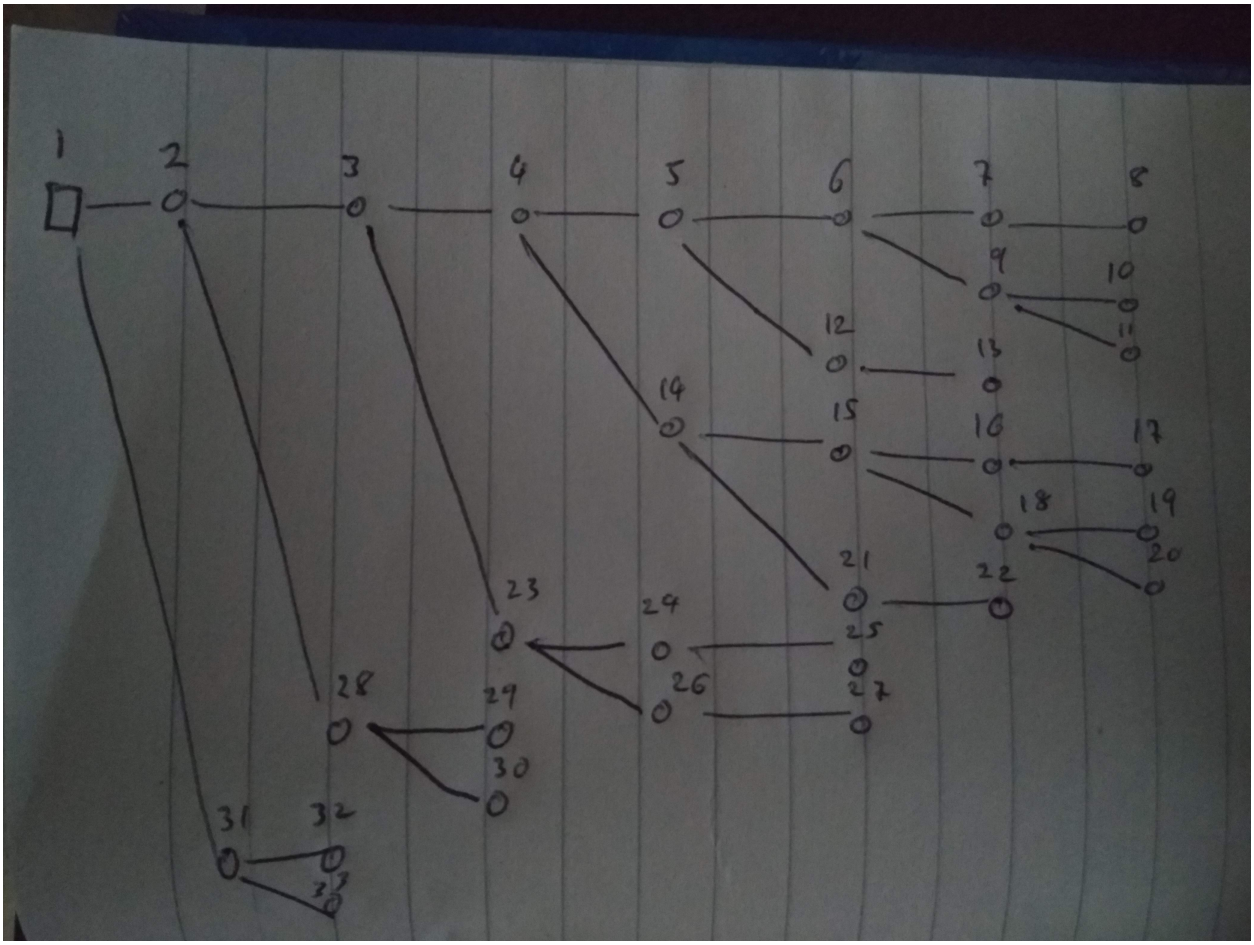
This included the tree structure variables, the cost and probability arrays, the mapping arrays that inform which nodes have which label with have what cost.

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_IGRA_fwd_tree, 5)
```

```
## $`1`
## [1]  2 25
##
## $`2`
## [1]  3 23
##
## $`3`
## [1]  4 21
##
## $`4`
## [1]  5 19
##
## $`5`
## [1]  6 17
```

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_IGRA_fwd_tree)
print(as_tibble(probs), n = 100)
```

```
## # A tibble: 49 x 49
##      `1`   `2`   `3`   `4`   `5`   `6`   `7`   `8`   `9`  `10`  `11`  `12`
##    <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
##  1    NA   0.5  NA    NA    NA    NA    NA    NA    NA      NA  NA    NA
##  2    NA  NA     0.5  NA    NA    NA    NA    NA    NA      NA  NA    NA
##  3    NA  NA    NA     0.5  NA    NA    NA    NA    NA      NA  NA    NA
##  4    NA  NA    NA    NA     0.5  NA    NA    NA    NA      NA  NA    NA
##  5    NA  NA    NA    NA    NA     0.5  NA    NA    NA      NA  NA    NA
##  6    NA  NA    NA    NA    NA    NA     0.5  NA    NA      NA  NA    NA
##  7    NA  NA    NA    NA    NA    NA    NA     0.5  NA      NA  NA    NA
##  8    NA  NA    NA    NA    NA    NA    NA    NA     0.5    NA   0.5  NA
##  9    NA  NA    NA    NA    NA    NA    NA    NA    NA       1  NA    NA
## 10    NA  NA    NA    NA    NA    NA    NA    NA    NA      NA  NA    NA
## 11    NA  NA    NA    NA    NA    NA    NA    NA    NA      NA  NA     0.5
## 12    NA  NA    NA    NA    NA    NA    NA    NA    NA      NA  NA    NA
## 13    NA  NA    NA    NA    NA    NA    NA    NA    NA      NA  NA    NA
## 14    NA  NA    NA    NA    NA    NA    NA    NA    NA      NA  NA    NA
```

```
## 15    NA  NA    NA    NA    NA    NA    NA    NA    NA      NA  NA    NA
## 16    NA  NA    NA    NA    NA    NA    NA    NA    NA      NA  NA    NA
## 17    NA  NA    NA    NA    NA    NA    NA    NA    NA      NA  NA    NA
## 18    NA  NA    NA    NA    NA    NA    NA    NA    NA      NA  NA    NA
## 19    NA  NA    NA    NA    NA    NA    NA    NA    NA      NA  NA    NA
## 20    NA  NA    NA    NA    NA    NA    NA    NA    NA      NA  NA    NA
## 21    NA  NA    NA    NA    NA    NA    NA    NA    NA      NA  NA    NA
## 22    NA  NA    NA    NA    NA    NA    NA    NA    NA      NA  NA    NA
## 23    NA  NA    NA    NA    NA    NA    NA    NA    NA      NA  NA    NA
## 24    NA  NA    NA    NA    NA    NA    NA    NA    NA      NA  NA    NA
## 25    NA  NA    NA    NA    NA    NA    NA    NA    NA      NA  NA    NA
## 26    NA  NA    NA    NA    NA    NA    NA    NA    NA      NA  NA    NA
## 27    NA  NA    NA    NA    NA    NA    NA    NA    NA      NA  NA    NA
## 28    NA  NA    NA    NA    NA    NA    NA    NA    NA      NA  NA    NA
## 29    NA  NA    NA    NA    NA    NA    NA    NA    NA      NA  NA    NA
## 30    NA  NA    NA    NA    NA    NA    NA    NA    NA      NA  NA    NA
## 31    NA  NA    NA    NA    NA    NA    NA    NA    NA      NA  NA    NA
## 32    NA  NA    NA    NA    NA    NA    NA    NA    NA      NA  NA    NA
## 33    NA  NA    NA    NA    NA    NA    NA    NA    NA      NA  NA    NA
## 34    NA  NA    NA    NA    NA    NA    NA    NA    NA      NA  NA    NA
## 35    NA  NA    NA    NA    NA    NA    NA    NA    NA      NA  NA    NA
## 36    NA  NA    NA    NA    NA    NA    NA    NA    NA      NA  NA    NA
## 37    NA  NA    NA    NA    NA    NA    NA    NA    NA      NA  NA    NA
## 38    NA  NA    NA    NA    NA    NA    NA    NA    NA      NA  NA    NA
## 39    NA  NA    NA    NA    NA    NA    NA    NA    NA      NA  NA    NA
## 40    NA  NA    NA    NA    NA    NA    NA    NA    NA      NA  NA    NA
## 41    NA  NA    NA    NA    NA    NA    NA    NA    NA      NA  NA    NA
## 42    NA  NA    NA    NA    NA    NA    NA    NA    NA      NA  NA    NA
## 43    NA  NA    NA    NA    NA    NA    NA    NA    NA      NA  NA    NA
## 44    NA  NA    NA    NA    NA    NA    NA    NA    NA      NA  NA    NA
## 45    NA  NA    NA    NA    NA    NA    NA    NA    NA      NA  NA    NA
## 46    NA  NA    NA    NA    NA    NA    NA    NA    NA      NA  NA    NA
## 47    NA  NA    NA    NA    NA    NA    NA    NA    NA      NA  NA    NA
## 48    NA  NA    NA    NA    NA    NA    NA    NA    NA      NA  NA    NA
## 49    NA  NA    NA    NA    NA    NA    NA    NA    NA      NA  NA    NA
## # ... with 37 more variables: `13` <dbl>, `14` <dbl>, `15` <dbl>,
## #   `16` <dbl>, `17` <dbl>, `18` <dbl>, `19` <dbl>, `20` <dbl>,
## #   `21` <dbl>, `22` <dbl>, `23` <dbl>, `24` <dbl>, `25` <dbl>,
## #   `26` <dbl>, `27` <dbl>, `28` <dbl>, `29` <dbl>, `30` <dbl>,
## #   `31` <dbl>, `32` <dbl>, `33` <dbl>, `34` <dbl>, `35` <dbl>,
## #   `36` <dbl>, `37` <dbl>, `38` <dbl>, `39` <dbl>, `40` <dbl>,
## #   `41` <dbl>, `42` <dbl>, `43` <dbl>, `44` <dbl>, `45` <dbl>,
## #   `46` <dbl>, `47` <dbl>, `48` <dbl>, `49` <dbl>
```

```
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_QFT_fwd_pname_from_to
pname_from_to
```

```
##             name from to
## 1          pLTBI    1  2
## 2    pAccept_TST    2  3
## 3    pAccept_TST   25 26
## 4       pTSTread    3  4
## 5       pTSTread   26 27
## 6       TST_sens    4  5
## 7       TST_spec   27 42
## 8   pAccept_IGRA    5  6
## 9   pAccept_IGRA   28 29
## 10      QFT_sens    6  7
## 11      QFT_spec   29 38
## 12 pAccept_chemo    7  8
## 13 pAccept_chemo   30 31
## 14          pHep    8  9
## 15          pHep   31 32
## 16   pComp_chemo   11 12
## 17   pComp_chemo   34 35
```

```
cname_from_to <- TST_QFT_fwd_cname_from_to
cname_from_to
```

```
##                                name from to
## 1                               TST    2  3
## 2                               TST   25 26
## 3           TB special nurse visit    3  4
## 4           TB special nurse visit   26 27
## 5                               QFT    5  6
## 6                               QFT   28 29
## 7  Total Cost of positive screening    6  7
## 8  Total Cost of positive screening   29 30
## 9                               Hep    8  9
## 10                              Hep   31 32
## 11              Total (incomplete)    9 10
## 12              Total (incomplete)   32 33
## 13                Total (complete)   11 12
## 14                Total (complete)   34 35
```

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.

```r
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: 48 x 4
##     from    to name          prob
##    <dbl> <dbl> <chr>        <dbl>
##  1     1     2 pLTBI        0.326
##  2     1    25 <NA>         0.674
##  3    11    12 pComp_chemo  0.8
##  4    11    48 <NA>         0.200
##  5    13    14 <NA>         1
##  6    15    16 <NA>         1
##  7    17    18 <NA>         1
##  8    19    20 <NA>         1
##  9     2     3 pAccept_TST  0.982
## 10     2    23 <NA>         0.018
## # ... with 38 more rows
```

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

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

```
## # A tibble: 6 x 49
##       ``     ``     ``     ``    ``     ``    ``    ``    ``    ``    ``
##    <dbl>  <dbl>  <dbl>  <dbl> <dbl>  <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1     NA  0.326 NA     NA      NA   NA      NA      NA    NA    NA    NA
## 2     NA NA      0.982 NA      NA   NA      NA      NA    NA    NA    NA
## 3     NA NA     NA      0.979  NA   NA      NA      NA    NA    NA    NA
## 4     NA NA     NA     NA       0.8 NA      NA      NA    NA    NA    NA
## 5     NA NA     NA     NA      NA    0.992  NA      NA    NA    NA    NA
## 6     NA NA     NA     NA      NA   NA       0.9    NA    NA    NA    NA
## # ... with 38 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>, `` <dbl>,
## #   `` <dbl>, `` <dbl>, `` <dbl>, `` <dbl>, `` <dbl>, `` <dbl>
```

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                             Hep   31 32 732.13
## 3                             QFT    5  6  23.65
## 4                             QFT   28 29  23.65
## 5           TB special nurse visit    3  4  44.31
## 6           TB special nurse visit   26 27  44.31
## 7                 Total (complete)   11 12 169.68
## 8                 Total (complete)   34 35 169.68
## 9               Total (incomplete)    9 10  84.84
## 10              Total (incomplete)   32 33  84.84
## 11 Total Cost of positive screening    6  7 241.23
## 12 Total Cost of positive screening   29 30 241.23
## 13                            TST    2  3  18.62
## 14                            TST   25 26  18.62
```

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

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

```
## # A tibble: 6 x 49
##      ``    ``    ``    ``    ``    ``    ``    ``    ``    ``    ``    ``
##   <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1    NA    NA    NA    NA    NA    NA    NA    NA    NA    NA    NA    NA
## 2    NA    NA  18.6    NA    NA    NA    NA    NA    NA    NA    NA    NA
## 3    NA    NA    NA  44.3    NA    NA    NA    NA    NA    NA    NA    NA
```

```
## 4    NA    NA  NA    NA      NA  NA     NA     NA    NA    NA    NA    NA
## 5    NA    NA  NA    NA      NA  23.6   NA     NA    NA    NA    NA    NA
## 6    NA    NA  NA    NA      NA  NA    241.    NA    NA    NA    NA    NA
## # ... with 37 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>, `` <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.

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


res[1] + label_costs$`TB special nurse visit` #196.9457
```

```
##
## 196.9457
```

Create a single cost-effectiveness decision tree data frame.

```
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                   <NA>    NA       pLTBI 0.326
## 2    1 25                   <NA>    NA        <NA> 0.674
## 3    2  3                    TST 18.62 pAccept_TST 0.982
## 4    2 23                   <NA>    NA        <NA> 0.018
## 5    3  4 TB special nurse visit 44.31    pTSTread 0.979
## 6    3 21                   <NA>    NA        <NA> 0.021
```

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 *forwards* model i.e. with sensitivity and specificity:

```r
costs_SA <-
  tibble::tribble(~name.cost,                ~vals,
                  "TST",                     list(distn = "unif", params = c(min=9.31, max=37.24)),
                  "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)),
                  "QFT",                     list(distn = "unif", params = c(min=35, max=80)),
                  "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,
                  "pComp_chemo",  list(distn = "unif", params = c(min=0.28, max=0.97)), #Alsdurf (2016)
                  "pHep",         list(distn = "unif", params = c(min=0.001, max=0.003)),
                  "pAccept_chemo",list(distn = "unif", params = c(min=0.53, max=0.98)), #Alsdurf (2016)
                  "pAccept_TST",  list(distn = "unif", params = c(min=0.5, max=1)), #Campbell (2017)
                  "pTSTread",     list(distn = "unif", params = c(min=0.90, max=0.99)), #Alsdurf (2016)
                  "pAccept_IGRA", list(distn = "unif", params = c(min=0.5, max=1)),
                  "pLTBI",        list(distn = "unif", params = c(min=0.10, max=0.40)), #Pooran (2010)
                  # "TSPOT_sens",   list(distn = "unif", params = c(min=0.85, max=0.93)), #NICE appendi
                  # "TSPOT_spec",   list(distn = "unif", params = c(min=0.86, max=1.0)),
                  "QFT_sens",     list(distn = "unif", params = c(min=0.81, max=0.87)), #Diel (2010)
                  "QFT_spec",     list(distn = "unif", params = c(min=0.98, max=1.0)),
                  "TST_sens",     list(distn = "unif", params = c(min=0.73, max=1.0)), #Warwick evidenc
                  "TST_spec",     list(distn = "unif", params = c(min=0.12, max=0.53))) #Kik (2010) low
probs_SA
```

```
## # A tibble: 11 x 2
##    name.prob     prob
##    <chr>         <list>
##  1 pComp_chemo   <named list [2]>
##  2 pHep          <named list [2]>
##  3 pAccept_chemo <named list [2]>
##  4 pAccept_TST   <named list [2]>
##  5 pTSTread      <named list [2]>
##  6 pAccept_IGRA  <named list [2]>
##  7 pLTBI         <named list [2]>
##  8 QFT_sens      <named list [2]>
##  9 QFT_spec      <named list [2]>
## 10 TST_sens      <named list [2]>
## 11 TST_spec      <named list [2]>
```

Combine the distributions with the original tree specification data frame.

```r
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()
```

```
## Warning: Column `name.cost` joining factor and character vector, coercing
## into character vector
```

```
input_SA
```

```
## # A tibble: 48 x 6
##     from    to name.cost           name.prob    prob          vals
##    <dbl> <dbl> <chr>               <chr>        <list>        <list>
##  1     1     2 <NA>                pLTBI        <named list ~ <NULL>
##  2     1    25 <NA>                <NA>         <NULL>        <NULL>
##  3     2     3 TST                 pAccept_TST  <named list ~ <named list ~
##  4     2    23 <NA>                <NA>         <NULL>        <NULL>
##  5     3     4 TB special nurse v~ pTSTread     <named list ~ <named list ~
##  6     3    21 <NA>                <NA>         <NULL>        <NULL>
##  7     4     5 <NA>                TST_sens     <named list ~ <NULL>
##  8     4    19 <NA>                <NA>         <NULL>        <NULL>
##  9     5     6 QFT                 pAccept_IGRA <named list ~ <named list ~
## 10     5    17 <NA>                <NA>         <NULL>        <NULL>
## # ... with 38 more rows
```

We can now loop over this tree and generate samples of values for the given distributions. Lets use the `sample_distributions()` function from my `treeSimR` package.

```
tree_dat_sa <- list()

for (i in 1:500) {

  tree_dat_sa[[i]] <-
    define_model(dat_long =
                   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())) %>%
    select(-missing)
}
```

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

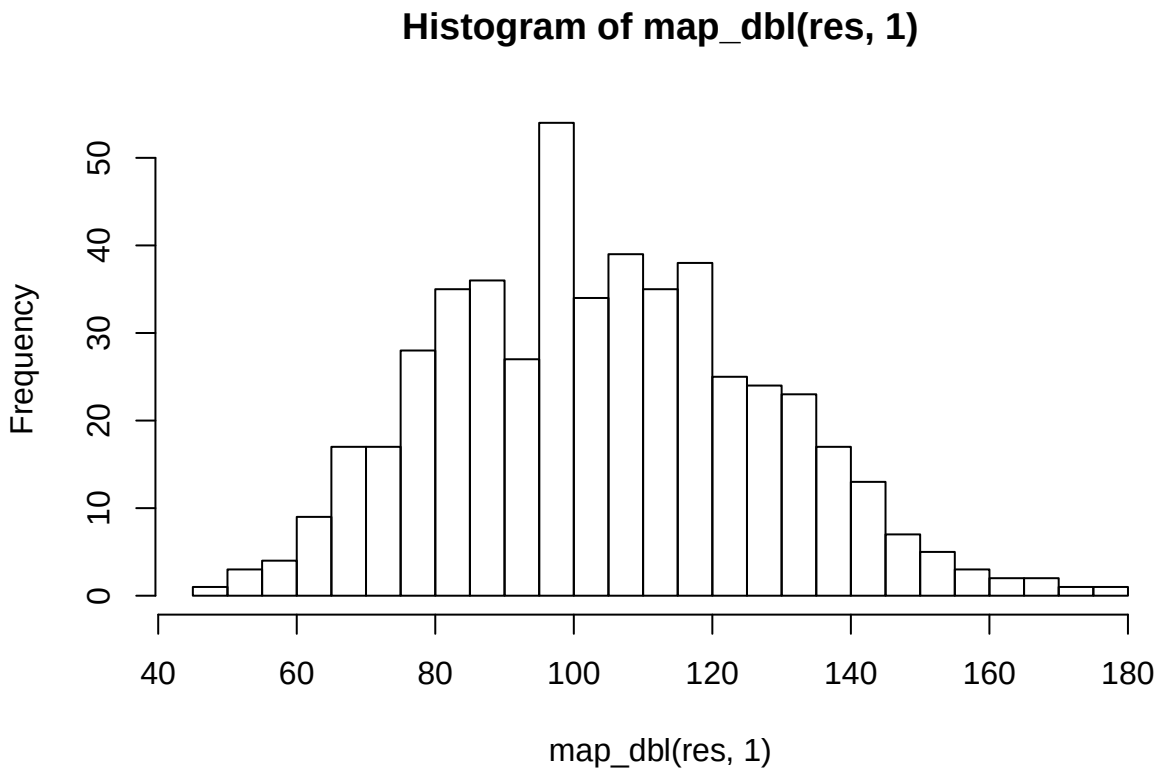
```
## [[1]]
##          1          2          3          4          5          6
##  110.11066  163.51208  236.97403  222.52612  229.97485  296.74174
##          7          8          9         10         11         12
##  305.92219   67.84412  477.28300   84.84000   67.02360  169.68000
##         13         14         15         16         17         18
##    0.00000    0.00000    0.00000    0.00000    0.00000    0.00000
##         19         20         21         22         23         24
```

```
##    0.00000    0.00000    0.00000    0.00000    0.00000    0.00000
##         25         26         27         28         29         30
##   80.46166   82.69441   63.96634   37.34776   54.28454  326.56098
##         31         32         33         34         35         36
##  116.95976 1412.08600   84.84000  114.36432  169.68000    0.00000
##         37         38         39         40         41         42
##    0.00000    0.00000    0.00000    0.00000    0.00000    0.00000
##         43         44         45         46         47         48
##    0.00000    0.00000    0.00000    0.00000    0.00000    0.00000
##         49
##    0.00000
##
## [[2]]
##          1          2          3          4          5          6
##   93.39040  130.06290  192.68578  192.55713  174.54591  306.22090
##          7          8          9         10         11         12
##  291.21826   98.49125 1022.81400   84.84000   97.56600  169.68000
##         13         14         15         16         17         18
##    0.00000    0.00000    0.00000    0.00000    0.00000    0.00000
##         19         20         21         22         23         24
##    0.00000    0.00000    0.00000    0.00000    0.00000    0.00000
##         25         26         27         28         29         30
##   76.36963   83.64692   70.28989   42.07652   49.91283  362.42582
##         31         32         33         34         35         36
##  137.55825  534.84500   84.84000  136.76208  169.68000    0.00000
##         37         38         39         40         41         42
##    0.00000    0.00000    0.00000    0.00000    0.00000    0.00000
##         43         44         45         46         47         48
##    0.00000    0.00000    0.00000    0.00000    0.00000    0.00000
##         49
##    0.00000
##
## [[3]]
##          1          2          3          4          5          6
##   86.37256  154.40046  246.25272  234.54045  214.47167  338.28340
##          7          8          9         10         11         12
##  332.68927  107.10160 1309.21500   84.84000  104.69256  169.68000
##         13         14         15         16         17         18
##    0.00000    0.00000    0.00000    0.00000    0.00000    0.00000
##         19         20         21         22         23         24
##    0.00000    0.00000    0.00000    0.00000    0.00000    0.00000
##         25         26         27         28         29         30
##   53.16846   70.14309   53.71997   25.55533   42.24021  348.22180
##         31         32         33         34         35         36
##  117.13186 1074.78300   84.84000  115.21272  169.68000    0.00000
##         37         38         39         40         41         42
##    0.00000    0.00000    0.00000    0.00000    0.00000    0.00000
##         43         44         45         46         47         48
##    0.00000    0.00000    0.00000    0.00000    0.00000    0.00000
##         49
##    0.00000
##
## [[4]]
##          1          2          3          4          5          6
```

```
##   85.30670  170.84520  225.68718  198.91040  228.57823  355.48713
##          7          8          9         10         11         12
##  348.53682  166.00265 1040.45700   84.84000  164.25024  169.68000
##         13         14         15         16         17         18
##    0.00000    0.00000    0.00000    0.00000    0.00000    0.00000
##         19         20         21         22         23         24
##    0.00000    0.00000    0.00000    0.00000    0.00000    0.00000
##         25         26         27         28         29         30
##   55.56420   61.66948   47.38173   27.13587   41.61943  322.49391
##         31         32         33         34         35         36
##  102.96775 1334.24700   84.84000   99.26280  169.68000    0.00000
##         37         38         39         40         41         42
##    0.00000    0.00000    0.00000    0.00000    0.00000    0.00000
##         43         44         45         46         47         48
##    0.00000    0.00000    0.00000    0.00000    0.00000    0.00000
##         49
##    0.00000
##
## [[5]]
##          1          2          3          4          5          6
##   97.69351  145.66416  202.87488  202.55104  201.80192  329.74169
##          7          8          9         10         11         12
##  314.99601  125.88009 1246.21800   84.84000  122.50896  169.68000
##         13         14         15         16         17         18
##    0.00000    0.00000    0.00000    0.00000    0.00000    0.00000
##         19         20         21         22         23         24
##    0.00000    0.00000    0.00000    0.00000    0.00000    0.00000
##         25         26         27         28         29         30
##   79.58889   84.94012   62.28239   41.60304   71.60592  360.40991
##         31         32         33         34         35         36
##  144.95857  895.26300   84.84000  142.70088  169.68000    0.00000
##         37         38         39         40         41         42
##    0.00000    0.00000    0.00000    0.00000    0.00000    0.00000
##         43         44         45         46         47         48
##    0.00000    0.00000    0.00000    0.00000    0.00000    0.00000
##         49
##    0.00000
##
## [[6]]
##          1          2          3          4          5          6
##   83.42142  147.13418  284.59223  274.35604  314.35300  341.31705
##          7          8          9         10         11         12
##  343.19769  119.51390 1195.69800   84.84000  118.43664  169.68000
##         13         14         15         16         17         18
##    0.00000    0.00000    0.00000    0.00000    0.00000    0.00000
##         19         20         21         22         23         24
##    0.00000    0.00000    0.00000    0.00000    0.00000    0.00000
##         25         26         27         28         29         30
##   49.11454   91.12160   67.91126   54.51268   70.06771  294.71411
##         31         32         33         34         35         36
##   96.44923 1317.25600   84.84000   94.00272  169.68000    0.00000
##         37         38         39         40         41         42
##    0.00000    0.00000    0.00000    0.00000    0.00000    0.00000
##         43         44         45         46         47         48
```

```
##    0.00000    0.00000    0.00000    0.00000    0.00000    0.00000
##         49
##    0.00000
```

```r
hist(map_dbl(res, 1), breaks = 20)
```



Pathway joint probabilities

We calculate the terminal state total probabiliites for each LTBI status to use as inputs for the Markov model.

```r
pathway_joint_prob <- function(df,
                               terminal_node){

  out <- list()

  for (i in seq_along(terminal_node)) {

    to_node <- terminal_node[i]
    p_total <- 1

    while (to_node %in% df$to) {

      p_total <- c(p_total, df$prob[df$to == to_node])
      to_node <- df$from[df$to == to_node]
```

```
    }

    out[[i]] <- p_total
  }

  return(out)
}
```

```
LTBI_complete_Tx <-
  pathway_joint_prob(df = all_long,
                     terminal_node = 12) %>%
  map_dbl(prod) %>%
  sum()

LTBI_incomplete_Tx <-
  pathway_joint_prob(df = all_long,
                     terminal_node = c(10,48)) %>%
  map_dbl(prod) %>%
  sum()

LTBI_no_Tx <-
  pathway_joint_prob(df = all_long,
                     terminal_node = c(14,16,18,20,22,24)) %>%
  map_dbl(prod) %>%
  sum()

no_LTBI <-
  pathway_joint_prob(df = all_long,
                     terminal_node = c(33,35,37,39,41,43,45,47,49)) %>%
  map_dbl(prod) %>%
  sum()
```

Check that they sum to one.

```
all.equal(1 -(LTBI_complete_Tx + LTBI_incomplete_Tx + LTBI_no_Tx), no_LTBI)
```

```
## [1] TRUE
```

Now we can do the same for the SA data to give a distribution of terminal state subpopulations.

```
terminal_pop <- function(df) {

  res <- list()

  res$LTBI_complete_Tx <-
    pathway_joint_prob(df,
                       terminal_node = 12) %>%
    map_dbl(prod) %>%
    sum()

  res$LTBI_incomplete_Tx <-
    pathway_joint_prob(df,
```

```
                        terminal_node = c(10,48)) %>%
    map_dbl(prod) %>%
    sum()

  res$LTBI_no_Tx <-
    pathway_joint_prob(df,
                        terminal_node = c(14,16,18,20,22,24)) %>%
    map_dbl(prod) %>%
    sum()

  res$no_LTBI <-
    pathway_joint_prob(df,
                        terminal_node = c(33,35,37,39,41,43,45,47,49)) %>%
    map_dbl(prod) %>%
    sum()

  return(res)
}
```

Check

```
terminal_pop(all_long)
```

```
## $LTBI_complete_Tx
## [1] 0.1697853
##
## $LTBI_incomplete_Tx
## [1] 0.04287164
##
## $LTBI_no_Tx
## [1] 0.1133431
##
## $no_LTBI
## [1] 0.674
```

Now we can map over all of the sample trees. Note we restrict to the non-NA branches. This is something to fix properly in the future.

```
xx <- map(tree_dat_sa, .f = function(x) terminal_pop(x[1:48, ]))
```

Plot the results with the original scenario indicated by the vertical red line.

```
par(mfrow=c(2,2))

hist(map_dbl(xx, "LTBI_complete_Tx"), breaks = 25)
abline(v=LTBI_complete_Tx, col = "red")

hist(map_dbl(xx, "LTBI_incomplete_Tx"), breaks = 25)
abline(v=LTBI_incomplete_Tx, col = "red")

hist(map_dbl(xx, "LTBI_no_Tx"), breaks = 25)
abline(v=LTBI_no_Tx, col = "red")
```

```r
hist(map_dbl(xx, "no_LTBI"), breaks = 25)
abline(v=no_LTBI, col = "red")
```