Tree aggregation

Contents

0.1 Data preparation . . . 2
    0.1.1 The tree . . . 2
    0.1.2 The count table . . . 2
0.2 Data analysis . . . 3
0.3 Tree aggregation . . . 4
0.4 Result visualisation . . . 4

In this Vignette, I use the data generated by Lukas as an example to show how to use the **treeAGG** package.

0.1 Data preparation

0.1.1 The tree

The tree data created by Lukas from the cytometry data is a hclust object. To use treeAGG, we need to firstly transform it into a phylo object.

```
data("hclust_out")
class(hclust_out)
```

```
## [1] "hclust"
```

```
phylo_out <- as.phylo(hclust_out)
class(phylo_out)
```

```
## [1] "phylo"
```

We further prune the tree at each internal node into subtrees. The results are output as a list of phylo objects, each representing a subtree. This step is not obligatory. The list of subtrees is required later for tree aggregation. It could be generated automatically in the aggregation R functions, but the cutting could take a lot of time if the tree has huge size. To run it once and save it for later use would save a lot of time.

```
# prune tree
system.time(
phylo_small <- pruneTree(phylo_out)
)
```

```
##    user  system elapsed
##   1.450   0.013   1.463
```

0.1.2 The count table

A count table with each row representing a node in the tree and each column representing a sample is required. If data only provides the count at tree leaf nodes, we could use the following code to generate a count table for the whole tree.

```
data("res_table")
head(res_table)
```

```
##   cluster    p_vals     p_adj n_cells n_spikein prop_spikein  true
## 1       1 0.1318399 0.7094024     133         0            0 FALSE
## 2       2 0.2645481 0.7586307     157         0            0 FALSE
## 3       3 0.1719556 0.7456668     117         0            0 FALSE
## 4       4 0.1288124 0.7094024      96         0            0 FALSE
## 5       5        NA        NA     119         0            0 FALSE
## 6       6        NA        NA     140         0            0 FALSE
##   healthy_H1 healthy_H2 healthy_H3 healthy_H4 healthy_H5 CN_H1 CN_H2 CN_H3
## 1          6          6         17         14         30     6     7     6
## 2          3          2          2         53          9     4     3     5
## 3         10          3          0         42          9     9     0     2
## 4          3          1          5          7         23    11     0     1
## 5          3          0          0          4         44     4     0     2
## 6          1          0          0          4         63     1     0     0
##   CN_H4 CN_H5
## 1    12    29
## 2    62    14
## 3    37     5
## 4    12    33
## 5     6    56
## 6     3    68
```

```
# generate the count table for the leaf nodes
tipCount <- res_table[, c(grep("healthy_", colnames(res_table)),
                         grep("CN_H", colnames(res_table)))]
rownames(tipCount) <- res_table$cluster
dim(tipCount)
```

```
## [1] 900  10
```

```
# generate the count table for the whole tree (including leaves and internal nodes)
# the count of an internal node is the sum of counts on its descendant leaves.
allCount <- nodeCount(tipTable = tipCount, wtree = phylo_out,
                      stree = phylo_small, fun = sum)
dim(allCount)
```

```
## [1] 1799   10
```

0.2 Data analysis

We arrange the hypothesis in a tree-like structure and test hypotheses H_0 : There is not differential abundance at each node (interal node and leaf node) of the tree.

R package **edgeR** is used here to investigate whether an entity has differential abundance among conditions. Users are free to choose other R packages to do the analysis. Finally, we obtain a p value at each node of the tree and further use them to do tree aggregation.

```
library(edgeR)
```

```
## Loading required package: limma
```

```
isNode <- grepl("Node", rownames(allCount))
mod_edgeR <- Redge(countTab = allCount, nSam = c(5,5),
                   isTip = !isNode, isAnalyze = rep(TRUE, nrow(allCount)),
                   prior.count = 0, normalize = TRUE)
head(mod_edgeR)
```

```
##        logFC    logCPM          LR    PValue       FDR     predLFC
## 1 -0.3952743 10.493679 0.292249653 0.5887825 0.9999643 -0.39895823
## 2  0.6634443 10.027185 0.424348571 0.5147751 0.9999643  0.36535617
## 3 -0.2653306 10.171643 0.059809621 0.8067971 0.9999643 -0.24281463
## 4  0.5385744 10.010648 0.269474577 0.6036844 0.9999643  0.54627632
## 5  0.7591419  9.595170 0.244376790 0.6210627 0.9999643  0.45126260
## 6 -0.1471498  9.139289 0.006391183 0.9362811 0.9999643  0.08176307
##      waldAP   std.err   estimate  tag.disp
## 1 0.9998938 0.5118648 -0.2739833 0.5678814
## 2 0.9998938 0.6740882  0.4598645 1.0032028
## 3 0.9998938 0.7586387 -0.1839132 1.3257010
## 4 0.9998938 0.7305720  0.3733113 1.2017075
## 5 0.9998938 1.0226228  0.5261971 2.4144973
## 6 0.9998938 1.2255563 -0.1019965 3.4647141
```

The output has multiple columns, one of which is the adjusted p value named *FDR*. The adjusted p value is obtained via Benjamin-Hochberg method. We could directly use the *mod_edgeR* to do the tree aggregation in the next section or extract only the column *FDR*.

0.3 Tree aggregation

To do tree aggregation, we need a hiearchical tree (*phylo_out*), and a matrix or data frame with a column of adjusted p vlaues at each node of the tree (*mod_edgeR*). The subtrees of the hiearchical tree (phylo_small) could be provided to save time. If stree is set as NULL, the list of subtrees would be generated automatically.

```
# min - P
loc_edgeR <- treeAGG(wtree = phylo_out, data = mod_edgeR,
                     stree = phylo_small, P.lim = 0.05,
                     varSIG = "FDR", varAGG = "FDR")
```

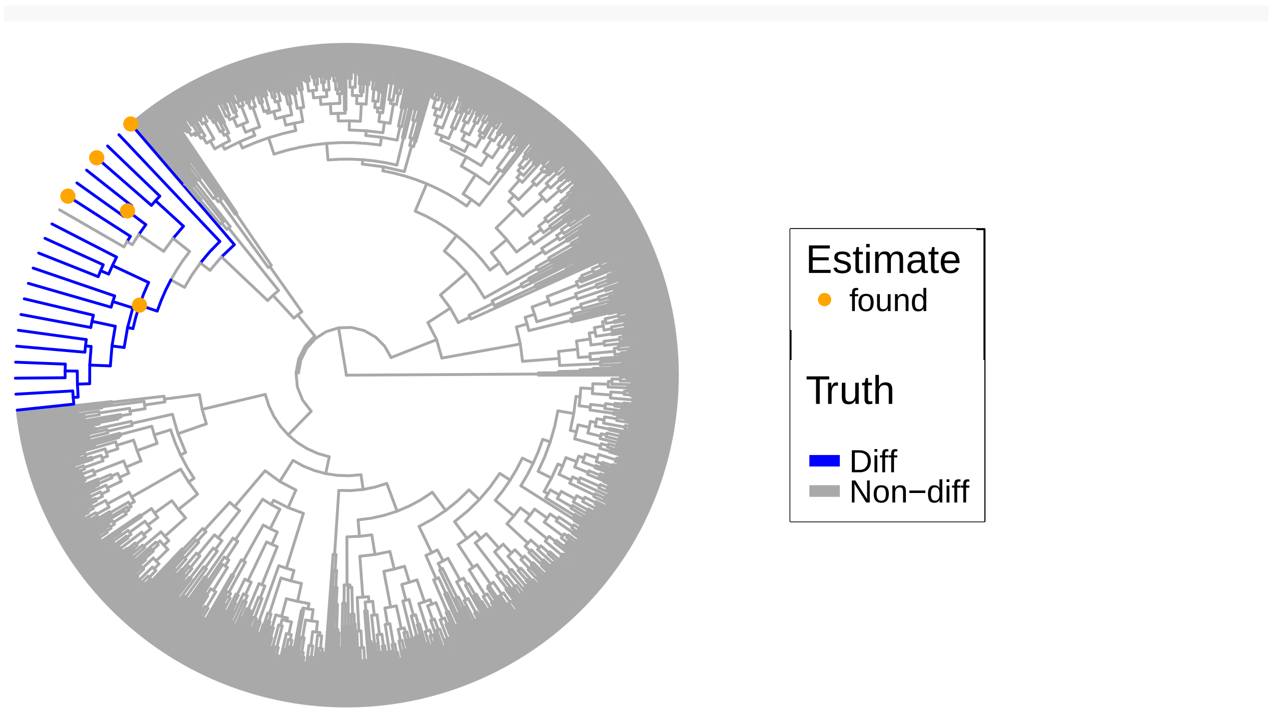
- Item 1 wtree is the hiearchical tree of the entities (such as OTUs or cells).
- Item 2 data is the data frame including at least a column of adjusted p values for all nodes of hiearchical tree.
- Item 3 stree is a list of subtrees of the hiearchical tree.
- Item 4 P.lim is the significant threshold value for the adjusted p value.
- Item 5 varSIG is the column name of the adjusted p value.
- Item 6 varAGG is the column name of the variable the aggregation based on. Here, we also use the adjusted p value.

0.4 Result visualisation

```
trueTip <- as.character(res_table$cluster[res_table$true])
real <- signalFind(node = trueTip, tree = phylo_out, label = TRUE)

p <- treePlot(tree = phylo_out,
              branch = real,
              col.branch = c("Diff" = "blue"),
              col.other = c("Non-diff" = "darkgrey"),
              size.point = 2,
              size.line.legend = 2,
              size.point.legend = 2,
              point = loc_edgeR,
              col.point = c(found = "orange"),
              zoomNode = real,
              zoomScale = 8,
              legend.title = c("point" = "Estimate",
                               "branch" = "Truth"),
              layout = "circular")

p+
  ggplot2::theme(legend.position = "right",
                 legend.text = element_text(size= 12),
                 legend.key.size = unit(4,"cm"),
                 legend.key.height = unit(0.4,"cm"),
                 legend.key.width = unit(0.5, "cm"),
                 legend.title = element_text(size = 15),
                 legend.background = element_rect(),
                 legend.box.background = element_rect(colour = "black"))
```


Estimate
found
Truth
Diff
Non−diff