F1000Research Software Tool Article Template

Ruizhu HUANG [1,2]***, Charlotte Soneson*** [1,2]***, and Mark Robinson*** [1,2]

[1]Institute of Molecular Life Sciences, University of Zurich.
[2]SIB Swiss Institute of Bioinformatics.

12 July 2020

Abstract

The hierarchical structure appears in several biological fields (e.g., phylogenies, cell types), and usually represents different resolutions to view the data. It's practical to have a data container that stores the hierarchical structure with the biological profile data, and provides functions to easily access or manipulate data at different resolutions. Here, we present `TreeSummarizedExperiment`, a Bioconductor package that extend the `SingleCellExperiment` (Lun and Risso 2020) class to combine with the `phylo` class. It follows the convention of the `SummarizedExperment` family class, and provides the link information between the rows or columns of the `assays` and the nodes of a tree (in `phylo` class) to allow easy data manipulation at arbitrary levels of the tree. The package is designed to be extensible, allowing new functions on the tree (`phylo`) to be contributed by ourselves or other researchers in the future to provide more functionalities in the data manipulation. As the work is based on the `SummarizedExperiment` class and the `phylo` class, both of which are popular class with many R packages depending on, it is expected to be able to work closely and easily with many other tools.

Contents

R version: R version 4.0.2 (2020-06-22)

Bioconductor version: 3.11

1 Introduction

Biological data with a hierarchy appears in several fields. A notable example is in the microbial survey studies where the microbiome is profiled with amplicon sequencing or whole genome shotgun sequencing, and microbial taxa share a common evolutionary history that can be encoded as a tree. A hierarchical structure might also be seen in single cell cytometry and RNA-seq datasets where nodes of a tree represent cell sub-populations at different granularities. Currently, `phyloseq` and `SingleCellExperiment` (Lun and Risso 2020) are dominant classes in the microbial data and the single cell data analysis, respectively. The former is not a `SummarizedExperiment` class (*SummarizedExperiment* (Morgan et al. 2020)) that is widely used in Bioconductor, and the latter doesn't provide functionalities on the hierarchical structure. As there are similarities in data structure shared in these fields, we are motivated to develop a S4 class, `TreeSummarizedExperment`, to store hierarchical biological data. It extends the `SingleCellExperiment` (Lun and Risso 2020) class to provide linkages between the `assays` data and the tree objects. Because `TreeSummarizedExperiment` is a member of `SummarizedExperiment` family, it could benefit from many tools in the Bioconductor ecosystem that are developed for this family (e.g., *iSEE*). Given that all slots of the `phyloseq` class have their corresponding slots in `TreeSummarizedExperiment` class, it's quite convenient to do conversion in between.

The `TreeSummarizedExperiment` class is used to store the rectangular data with the hierarchical structure, and establishs the link between the `assays` and the tree structure. Compared to the `SingleCellExperiment` (Lun and Risso 2020) class, `TreeSummarizedExperiment` has four more slots.

- `rowTree`: the hierarchical structure on the rows of the `assays` tables.
- `rowLinks`: the link between rows of the `assays` tables and the `rowTree`.
- `colTree`: the hierarchical structure on the columns of the `assays` tables.
- `colLinks`: the link information between columns of `assays` tables and the `colTree`.

The `rowTree` and `colTree` could be empty (`NULL`) if no trees are available. Correspondingly, the `rowLinks` and `colLinks` would be `NULL`. All the other slots in `TreeSummarizedExperiment` are inherited from `SingleCellExperiment` (Lun and Risso 2020).

The slots `rowTree` and `colTree` only accept the tree data as the `phylo` class. If a tree is available in other formats, one would need to convert it to `phylo` with other R packages (e.g., *treeio* (Wang et al. 2019)).

The *TreeSummarizedExperiment* package provides functions that could be separated into two main types. Functions in the first type directly work on the `TreeSummarizedExperiment` class (e.g., constructors and accessors); and others work on the tree (`phylo`) class. The latter is used mainly to create customized functions on the `TreeSummarizedExperiment` class. Also, users are freely to use functions available in other R packages to manipulate the `phylo` class (e.g., *ape* (Paradis and Schliep 2019)).

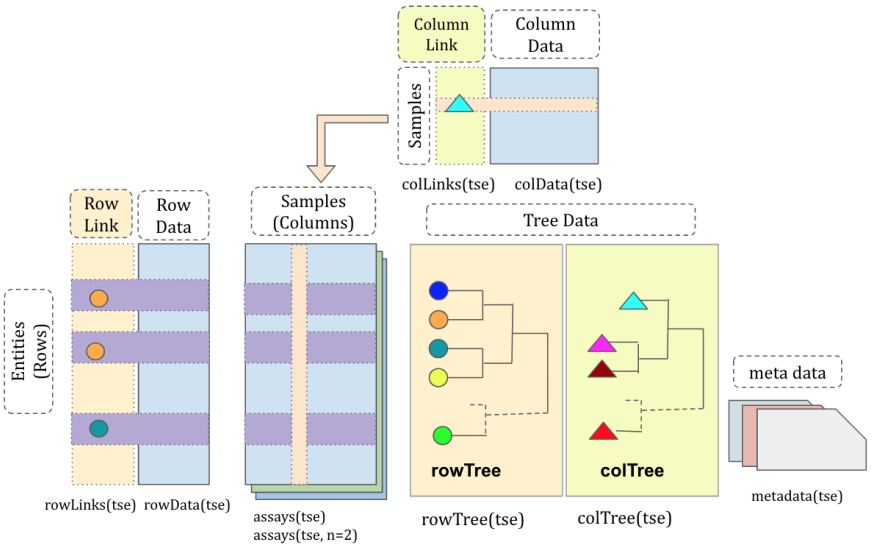


Figure 1: **The structure of the TreeSummarizedExperiment class**

2 Methods

2.1 Implementation

```
library(TreeSummarizedExperiment)
```

We generate a toy datset that has observations of 6 entities collected from 4 samples as an example to show how to construct the `TreeSummarizedExperment` object.

```
# assays data
assay_data <- rbind(rep(0, 4), matrix(1:20, nrow = 5))
colnames(assay_data) <- paste(rep(LETTERS[1:2], each = 2),
                              rep(1:2, 2), sep = "_")
rownames(assay_data) <- paste("entity", seq_len(6), sep = "")
assay_data
##         A_1 A_2 B_1 B_2
## entity1   0   0   0   0
## entity2   1   6  11  16
## entity3   2   7  12  17
## entity4   3   8  13  18
## entity5   4   9  14  19
## entity6   5  10  15  20
```

The information of entities and samples are given in the **row_data** and **col_data**, respectively.

```
# row data
row_data <- data.frame(Kindom = rep("A", 6),
                       Phylum = c("B1", "B1", rep("B2", 4)),
                       Class = c("C1", "C1", "C2", "C2", "C3", "C3"),
```

```
                      OTU = c("D1", "D2", "D3", "D4", "D5", "D6"),
                      row.names = rownames(assay_data),
                      stringsAsFactors = FALSE)

row_data
##         Kindom Phylum Class OTU
## entity1      A     B1    C1  D1
## entity2      A     B1    C1  D2
## entity3      A     B2    C2  D3
## entity4      A     B2    C2  D4
## entity5      A     B2    C3  D5
## entity6      A     B2    C3  D6
# column data
col_data <- data.frame(gg = c(1, 2, 3, 3),
                    group = rep(LETTERS[1:2], each = 2),
                    row.names = colnames(assay_data),
                     stringsAsFactors = FALSE)
col_data
##     gg group
## A_1  1     A
## A_2  2     A
## B_1  3     B
## B_2  3     B
```

The hierarchical structure of the 6 entities and 4 samples are denoted as **row_tree** and **col_tree**, respectively. The two trees are `phylo` objects randomly created with `rtree` from the package *ape*.

```
library(ape)

# The first toy tree
set.seed(12)
row_tree <- rtree(5)

# The second toy tree
set.seed(12)
col_tree <- rtree(4)

# change node labels
col_tree$tip.label <- colnames(assay_data)
col_tree$node.label <- c("All", "GroupA", "GroupB")
```

We visualize the tree using the package *ggtree* (Yu et al. 2017). The node labels and the node numbers are in blue and orange texts, respectively. Here, the internal nodes of the **row_tree** have no labels.

```
library(ggtree)
library(ggplot2)

# Visualize the row tree
ggtree(row_tree, size = 2, branch.length = "none") +
geom_text2(aes(label = node), color = "darkblue",
```

```
                  hjust = -0.5, vjust = 0.7, size = 4) +
geom_text2(aes(label = label), color = "darkorange",
           hjust = -0.1, vjust = -0.7, size = 4) +
  ylim(c(0.8, 5.5))
```

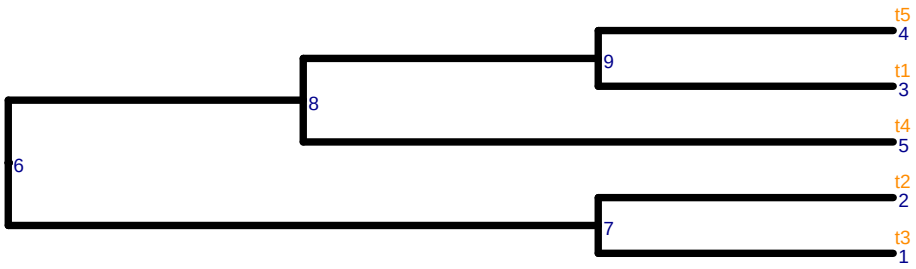


Figure 2: **The structure of the row tree**

The **col_tree** has labels for internal nodes.

```
# Visualize the column tree
ggtree(col_tree, size = 2, branch.length = "none") +
geom_text2(aes(label = node), color = "darkblue",
                hjust = -0.5, vjust = 0.7, size = 4) +
geom_text2(aes(label = label), color = "darkorange",
           hjust = -0.1, vjust = -0.7, size = 4)+
  ylim(c(0.8, 4.5)) +
  xlim(c(0, 2.2))
```

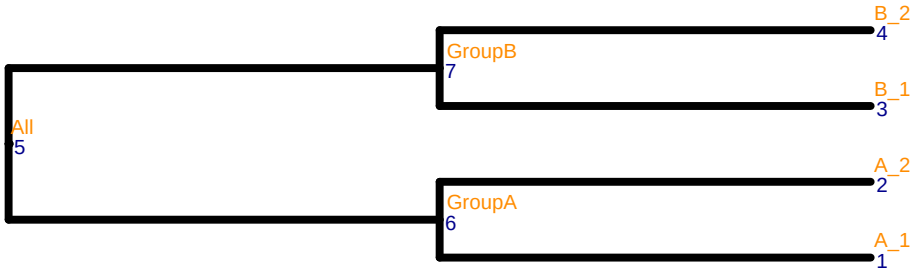


Figure 3: **The structure of the column tree**

2.1.1 The construction of `TreeSummarizedExperiment`

The `TreeSummarizedExperiment` class is used to store the toy data: **assay_data**, **row_data**, **col_data**, **col_tree** and **row_tree**, To correctly store data, the link information between the rows (or columns) of **assay_data** and the nodes of the **row_tree** (or **col_tree**) is required to provide via a character vector `rowNodeLab` (or `colNodeLab`). Those columns or rows that mismatch with nodes of the tree are removed with warnings. The link data between the `assays` tables and the tree data is automatically generated in the construction.

The row and the column trees are allowed to be included simultaneously in the construction. As the column names of the `assays` table and the node labels of the column tree are consistent, we omit the step of providing `colNodeLab`.

```
# provide the node labels in rowNodeLab
tip_lab <- row_tree$tip.label
row_lab <- tip_lab[c(1, 1:5)]
all(colnames(assay_data) %in% c(col_tree$tip.label, col_tree$node.label))
## [1] TRUE

both_tse <- TreeSummarizedExperiment(assays = list(assay_data),
                                     rowData = row_data,
                                     colData = col_data,
                                     rowTree = row_tree,
                                     rowNodeLab = row_lab,
                                     colTree = col_tree)
```

```
both_tse
## class: TreeSummarizedExperiment
## dim: 6 4
## metadata(0):
## assays(1): ''
## rownames(6): entity1 entity2 ... entity5 entity6
## rowData names(4): Kindom Phylum Class OTU
## colnames(4): A_1 A_2 B_1 B_2
## colData names(2): gg group
## reducedDimNames(0):
## altExpNames(0):
## rowLinks: a LinkDataFrame (6 rows)
## rowTree: a phylo (5 leaves)
## colLinks: a LinkDataFrame (4 rows)
## colTree: a phylo (4 leaves)
```

When printing out **both_tse**, we see a similar message as `SingleCellExperiment` (Lun and Risso 2020) with four additional lines about `rowLinks`, `rowTree`, `colLinks` and `colTree`.

2.1.2 The accessor functions

For slots inherited from the family of `SummarizedExperiment` class, they could be accessed in the traditional way.

```
# access assays
assays(both_tse)

# the row data
rowData(both_tse)

# the column data
colData(both_tse)

# the metadata: it's empty here
metadata(both_tse)
```

For new slots, we provide `rowTree` (`colTree`) to access the row (column) trees, and `rowLinks` (`colLinks`) to access the link information between `assays` and nodes of the row (column) tree. If the tree is not available, the corresponding link data is `NULL`.

```
# access trees
rowTree(both_tse)
##
## Phylogenetic tree with 5 tips and 4 internal nodes.
##
## Tip labels:
## [1] "t3" "t2" "t1" "t5" "t4"
##
## Rooted; includes branch lengths.
colTree(both_tse)
##
## Phylogenetic tree with 4 tips and 3 internal nodes.
##
## Tip labels:
## [1] "A_1" "A_2" "B_1" "B_2"
## Node labels:
## [1] "All"    "GroupA" "GroupB"
##
## Rooted; includes branch lengths.
```

```
# access the link data
(rLink <- rowLinks(both_tse))
## LinkDataFrame with 6 rows and 4 columns
##             nodeLab nodeLab_alias   nodeNum    isLeaf
##         <character>   <character> <integer> <logical>
## entity1          t3       alias_1         1      TRUE
## entity2          t3       alias_1         1      TRUE
## entity3          t2       alias_2         2      TRUE
## entity4          t1       alias_3         3      TRUE
## entity5          t5       alias_4         4      TRUE
## entity6          t4       alias_5         5      TRUE
(cLink <- colLinks(both_tse))
## LinkDataFrame with 4 rows and 4 columns
##         nodeLab nodeLab_alias   nodeNum    isLeaf
##     <character>   <character> <integer> <logical>
## A_1         A_1       alias_1         1      TRUE
## A_2         A_2       alias_2         2      TRUE
## B_1         B_1       alias_3         3      TRUE
## B_2         B_2       alias_4         4      TRUE
```

The link data has the `LinkDataFrame` class that is extended from the `DataFrame` class with the restriction that it has at least four columns: **nodeLab**, **nodeLab_alias**, **nodeNum**, and **isLeaf**. More details about the `DataFrame` class could be found in the *S4Vectors* package.

- nodeLab: the labels of nodes on the tree
- nodeLab_alias: the alias labels of nodes on the tree
- nodeNum: the numbers of nodes on the tree
- isLeaf: whether the node is a leaf node

2.1.3 The subseting function

Two ways are available to subset a `TreeSummarizedExperiment` object: `[` to subset by rows or columns, and `subsetByNode` to subset by nodes of a tree. To keep track of the original data, the `rowTree` and `colTree` stay the same after subseting.

```
sub_tse <- both_tse[1:2, 1]
sub_tse
## class: TreeSummarizedExperiment
## dim: 2 1
## metadata(0):
## assays(1): ''
## rownames(2): entity1 entity2
## rowData names(4): Kindom Phylum Class OTU
## colnames(1): A_1
## colData names(2): gg group
## reducedDimNames(0):
## altExpNames(0):
## rowLinks: a LinkDataFrame (2 rows)
## rowTree: a phylo (5 leaves)
## colLinks: a LinkDataFrame (1 rows)
## colTree: a phylo (4 leaves)
```

`rowLinks` and `rowData` are updated accordingly.

```
# The first four columns are from rowLinks data and the others from the rowData
cbind(rowLinks(sub_tse), rowData(sub_tse))
## DataFrame with 2 rows and 8 columns
##             nodeLab nodeLab_alias   nodeNum    isLeaf      Kindom      Phylum
##         <character>   <character> <integer> <logical> <character> <character>
## entity1          t3       alias_1         1      TRUE           A          B1
## entity2          t3       alias_1         1      TRUE           A          B1
##               Class         OTU
##         <character> <character>
## entity1          C1          D1
## entity2          C1          D2
```

```
# The first four columns are from colLinks data and the others from colData
cbind(colLinks(sub_tse), colData(sub_tse))
## DataFrame with 1 row and 6 columns
##         nodeLab nodeLab_alias   nodeNum    isLeaf        gg       group
##     <character>   <character> <integer> <logical> <numeric> <character>
## A_1         A_1       alias_1         1      TRUE         1           A
```

To subset by nodes, we specify the node by its node label or node number. Here, *entity1* and *entity2* are both mapped to the same node `t3`, so both of them are obtained.

```
node_tse <- subsetByNode(x = both_tse, rowNode = "t3")

rowLinks(node_tse)
## LinkDataFrame with 2 rows and 4 columns
##             nodeLab nodeLab_alias   nodeNum    isLeaf
```

```
##         <character>   <character> <integer> <logical>
## entity1          t3       alias_1         1      TRUE
## entity2          t3       alias_1         1      TRUE
```

It is allowed to subset simultaneously in both dimensions.

```
node_tse <- subsetByNode(x = both_tse, rowNode = "t3",
                         colNode = c("A_1", "A_2"))
assays(node_tse)[[1]]
##         A_1 A_2
## entity1   0   0
## entity2   1   6
```

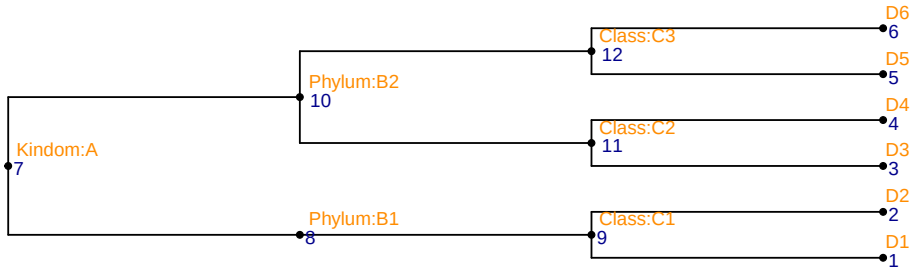
2.1.4 Change the tree

The current tree is allowed to be replaced by a new one by using the `changeTree`. If the hierarchical information is available as a `data.frame` with each column representing a taxonomic level (e.g., *row_data*), we provide `toTree` to convert it into a `phylo` object.

```
# The toy taxonomic table
taxa <- rowData(both_tse)

# convert it to a phylo tree
taxa_tree <- toTree(data = taxa)

# Viz the new tree
ggtree(taxa_tree)+
geom_text2(aes(label = node), color = "darkblue",
               hjust = -0.5, vjust = 0.7, size = 4) +
geom_text2(aes(label = label), color = "darkorange",
           hjust = -0.1, vjust = -0.7, size = 4) +
    geom_point2()
```



The information to match nodes of two trees are required if nodes are labeled differently.

```
taxa_tse <- changeTree(x = both_tse, rowTree = taxa_tree,
                       rowNodeLab = taxa[["OTU"]])

taxa_tse
## class: TreeSummarizedExperiment
## dim: 6 4
## metadata(0):
```

```
## assays(1): ''
## rownames(6): entity1 entity2 ... entity5 entity6
## rowData names(4): Kindom Phylum Class OTU
## colnames(4): A_1 A_2 B_1 B_2
## colData names(2): gg group
## reducedDimNames(0):
## altExpNames(0):
## rowLinks: a LinkDataFrame (6 rows)
## rowTree: a phylo (6 leaves)
## colLinks: a LinkDataFrame (4 rows)
## colTree: a phylo (4 leaves)
rowLinks(taxa_tse)
## LinkDataFrame with 6 rows and 4 columns
##             nodeLab nodeLab_alias   nodeNum    isLeaf
##         <character>   <character> <integer> <logical>
## entity1          D1       alias_1         1      TRUE
## entity2          D2       alias_2         2      TRUE
## entity3          D3       alias_3         3      TRUE
## entity4          D4       alias_4         4      TRUE
## entity5          D5       alias_5         5      TRUE
## entity6          D6       alias_6         6      TRUE
```

2.1.5 Aggregation

Data can be flexibly aggregated to different levels of the tree.

2.1.5.1 The column dimension Here, we show the aggregation on the column dimension. The `TreeSummarizedExperiment` object is assigned to the argument `x`. The desired aggregation level is given in `colLevel`. The level could be specified via the node label (the orange texts in Figure 3) or the node number (the blue texts in Figure 3). We could further decide how to aggregate via the argument `FUN`.

```
# use node labels to specify colLevel
aggCol <- aggValue(x = taxa_tse,
                   colLevel = c("GroupA", "GroupB"),
                   FUN = sum)
# or use node numbers to specify colLevel
aggCol <- aggValue(x = taxa_tse, colLevel = c(6, 7), FUN = sum)
```

```
assays(aggCol)[[1]]
##         alias_6 alias_7
## entity1       0       0
## entity2       7      27
## entity3       9      29
## entity4      11      31
## entity5      13      33
## entity6      15      35
```

The `rowData` does not change, but the `colData` adjusts with the change of the `assays` table. For example, the column **group** has the `A` value for `GroupA` because the descendant nodes of `GroupA` all have the value `A`; the column **gg** has the `NA` value for `GroupA` because the descendant nodes of `GroupA` have different values, (1 and 2).

```
# before aggregation
colData(taxa_tse)
## DataFrame with 4 rows and 2 columns
##            gg       group
##     <numeric> <character>
## A_1         1           A
## A_2         2           A
## B_1         3           B
## B_2         3           B
# after aggregation
colData(aggCol)
## DataFrame with 2 rows and 2 columns
##                gg       group
##         <numeric> <character>
## alias_6        NA           A
## alias_7         3           B
```

The `colLinks` is updated to link the new rows of `assays` tables and the column tree.

```
# the link data is updated
colLinks(aggCol)
## LinkDataFrame with 2 rows and 4 columns
##             nodeLab nodeLab_alias   nodeNum    isLeaf
##         <character>   <character> <integer> <logical>
## alias_6      GroupA       alias_6         6     FALSE
## alias_7      GroupB       alias_7         7     FALSE
```

From the Figure 2, we could see that the nodes 6 and 7 are labeled with `GroupA` and `GroupB`, respectively. This agrees with the column link data.

2.1.5.2 The row dimension

Similarly, we could aggregate the data to the phylum level by providing the internal nodes that represent the phylum level, `taxa_one`.

```
# the phylum level
taxa <- c(taxa_tree$tip.label, taxa_tree$node.label)
(taxa_one <- taxa[startsWith(taxa, "Phylum:")])
## [1] "Phylum:B1" "Phylum:B2"

# aggregation
agg_taxa <- aggValue(x = taxa_tse, rowLevel = taxa_one, FUN = sum)
agg_taxa
## class: TreeSummarizedExperiment
## dim: 2 4
## metadata(0):
## assays(1): ''
## rownames(2): alias_8 alias_10
## rowData names(4): Kindom Phylum Class OTU
```

```
## colnames(4): A_1 A_2 B_1 B_2
## colData names(2): gg group
## reducedDimNames(0):
## altExpNames(0):
## rowLinks: a LinkDataFrame (2 rows)
## rowTree: a phylo (6 leaves)
## colLinks: a LinkDataFrame (4 rows)
## colTree: a phylo (4 leaves)
```

It's free to choose nodes from different taxonomic ranks.

```
# A mixed level
taxa_mix <- c("Class:C3", "Phylum:B1")
agg_any <- aggValue(x = taxa_tse, rowLevel = taxa_mix, FUN = sum)
rowData(agg_any)
## DataFrame with 2 rows and 4 columns
##               Kindom      Phylum       Class       OTU
##          <character> <character> <character> <logical>
## alias_12           A          B2          C3        NA
## alias_8            A          B1          C1        NA
```

2.1.5.3 Both dimensions

The aggregation on both dimensions could be performed in one step using the same function specified via `FUN`. If different functions are required for different dimensions, the aggregation should be performed in two steps because the aggregation order matters.

```
agg_both <- aggValue(x = both_tse, colLevel = c(6, 7),
                     rowLevel = 7:9, FUN = sum)
```

As expected, we obtain a table with 3 rows (`rowLevel = 7:9`) and 2 columns (`colLevel = c(6, 7)`).

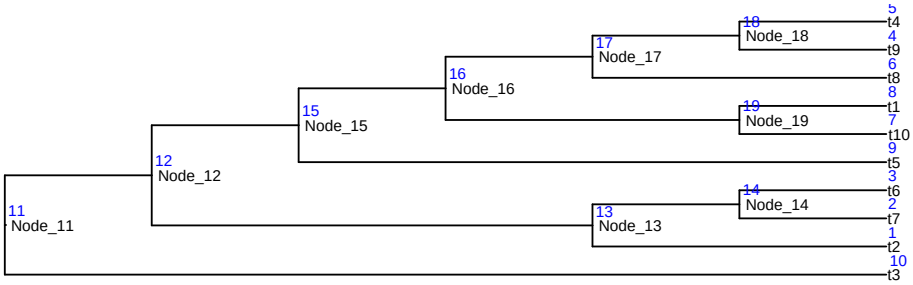
```
assays(agg_both)[[1]]
##         alias_6 alias_7
## alias_7      16      56
## alias_8      39      99
## alias_9      24      64
```

2.1.6 Functions working on the `phylo` object.

We create some functions to manipulate or to extract information from the `phylo` object. More functions could be found in other packages, such as *ape* (**???**), *tidytree*. These functions are useful when users want to customize functions for the `TreeSummarizedExperiment` class.

To show these functions, we use an example tree that has its node labels (black texts) and node numbers (blue texts) shown as below.

```
ggtree(tinyTree, branch.length = "none") +
    geom_text2(aes(label = label), hjust = -0.1, size = 3) +
    geom_text2(aes(label = node), vjust = -0.8,
               hjust = -0.2, color = 'blue', size = 3)
```



2.1.6.1 print out nodes of the tree

We could print out all nodes (`type = "all"`), the leaves (`type = "leaf"`) or the internal nodes (`type = "internal"`) with `printNode`.

```
printNode(tree = tinyTree, type = "all")
##    nodeLab nodeLab_alias nodeNum isLeaf
## 1       t2        alias1       1   TRUE
## 2       t7        alias2       2   TRUE
## 3       t6        alias3       3   TRUE
## 4       t9        alias4       4   TRUE
## 5       t4        alias5       5   TRUE
## 6       t8        alias6       6   TRUE
## 7      t10        alias7       7   TRUE
## 8       t1        alias8       8   TRUE
## 9       t5        alias9       9   TRUE
## 10      t3       alias10      10   TRUE
## 11 Node_11      alias_11      11  FALSE
## 12 Node_12      alias_12      12  FALSE
## 13 Node_13      alias_13      13  FALSE
## 14 Node_14      alias_14      14  FALSE
## 15 Node_15      alias_15      15  FALSE
## 16 Node_16      alias_16      16  FALSE
## 17 Node_17      alias_17      17  FALSE
## 18 Node_18      alias_18      18  FALSE
## 19 Node_19      alias_19      19  FALSE
```

2.1.6.2 Count the number of nodes

```
# The number of leaves
countLeaf(tree = tinyTree)
## [1] 10

# The number of nodes (leaf nodes and internal nodes)
countNode(tree = tinyTree)
## [1] 19
```

2.1.6.3 Conversion of the node label and the node number

The translation between the labels and the numbers of nodes could be achieved by the function `transNode`.

```
transNode(tree = tinyTree, node = c(12, 1, 4))
## [1] "Node_12" "t2"      "t9"
```

```
transNode(tree = tinyTree, node = c("t4", "Node_18"))
##      t4 Node_18
##       5      18
```

2.1.6.4 Find the descendants

To get descendants that are at the leaf level, we could set the argument `only.leaf = TRUE`.

```
# only the leaf nodes
findOS(tree = tinyTree, node = 17, only.leaf = TRUE)
## $Node_17
## [1] 4 5 6
```

The argument `only.leaf = FALSE` is set to get all descendants

```
# all descendant nodes
findOS(tree = tinyTree, node = 17, only.leaf = FALSE)
## $Node_17
## [1]  4  5  6 18
```

2.1.6.5 Find the sibling node

The input `node` could be either the node label or the node number.

```
# node = 5, node = "t4" are the same node
findSibling(tree = tinyTree, node = 5)
## t9
##  4
findSibling(tree = tinyTree, node = "t4")
## t9
##  4
```

2.1.6.6 Find the share node

This would find the first node that joined by the specified nodes (`node`) in their paths to the root.

```
shareNode(tree = tinyTree, node = c(5, 6))
## Node_17
##      17
```

```
isLeaf(tree = tinyTree, node = 5)
## [1] TRUE
isLeaf(tree = tinyTree, node = 17)
## [1] FALSE
```

2.1.6.7 Identify leaf nodes

2.1.6.8 The distance between two nodes

The distance between two nodes is obtained using `distNode`.

```
distNode(tree = tinyTree, node = c(1, 5))
## [1] 2.699212
```

2.1.6.9 Convert a `phylo` object to a matrix

Each row gives a path that connects a leaf and the root. Each entry value is a node reprsented by its node number.

```
matTree(tree = tinyTree)
##       L1 L2 L3 L4 L5 L6 L7
##  [1,]  1 13 12 11 NA NA NA
##  [2,]  2 14 13 12 11 NA NA
##  [3,]  3 14 13 12 11 NA NA
##  [4,]  4 18 17 16 15 12 11
##  [5,]  5 18 17 16 15 12 11
##  [6,]  6 17 16 15 12 11 NA
##  [7,]  7 19 16 15 12 11 NA
##  [8,]  8 19 16 15 12 11 NA
##  [9,]  9 15 12 11 NA NA NA
## [10,] 10 11 NA NA NA NA NA
```

2.1.7 Customize functions for the `TreeSummarizedExperiment` class

We show examples about how to create functions for the `TreeSummarizedExperiment`. Here, the function `rmRows` is to remove entities (on rows) that have zero in all samples (on columns) in the first `assays` table.

```
# dat: a TreeSummarizedExperiment
rmRows <- function(dat) {
    # calculate the total counts of each row
    count <- assays(dat)[[1]]
    tot <- apply(count, 1, sum)

    # find the row with zero in all columns
    ind <- which(tot == 0)

    # remove those rows
    out <- dat[-ind, ]
    return(out)

}
(rte <- rmRows(dat = both_tse))
## class: TreeSummarizedExperiment
## dim: 5 4
## metadata(0):
## assays(1): ''
## rownames(5): entity2 entity3 entity4 entity5 entity6
## rowData names(4): Kindom Phylum Class OTU
## colnames(4): A_1 A_2 B_1 B_2
```

```
## colData names(2): gg group
## reducedDimNames(0):
## altExpNames(0):
## rowLinks: a LinkDataFrame (5 rows)
## rowTree: a phylo (5 leaves)
## colLinks: a LinkDataFrame (4 rows)
## colTree: a phylo (4 leaves)
rowLinks(rte)
## LinkDataFrame with 5 rows and 4 columns
##             nodeLab nodeLab_alias   nodeNum    isLeaf
##         <character>   <character> <integer> <logical>
## entity2          t3       alias_1         1      TRUE
## entity3          t2       alias_2         2      TRUE
## entity4          t1       alias_3         3      TRUE
## entity5          t5       alias_4         4      TRUE
## entity6          t4       alias_5         5      TRUE
```

The function `rmRows` doesn't update the tree. Leaves that are mapped to the removed rows could be dropped with `ape::drop.tip`.

```
updateRowTree <- function(tse, dropLeaf) {
    ## -------------- new tree: drop leaves ----------
    oldTree <- rowTree(tse)
    newTree <- ape::drop.tip(phy = oldTree, tip = dropLeaf)

    ## -------------- update the row link ----------
    # track the tree
    track <- trackNode(oldTree)
    track <- ape::drop.tip(phy = track, tip = dropLeaf)

    # row links
    rowL <- rowLinks(tse)
    rowL <- DataFrame(rowL)

    # update the row links:
    #   1. use the alias label to track and updates the nodeNum
    #   2. the nodeLab should be updated based on the new tree using the new
    #      nodeNum
    #   3. lastly, update the nodeLab_alias
    rowL$nodeNum <- transNode(tree = track, node = rowL$nodeLab_alias,
                              message = FALSE)
    rowL$nodeLab <- transNode(tree = newTree, node = rowL$nodeNum,
                              use.alias = FALSE, message = FALSE)
    rowL$nodeLab_alias <- transNode(tree = newTree, node = rowL$nodeNum,
                                    use.alias = TRUE, message = FALSE)
    rowL$isLeaf <- isLeaf(tree = newTree, node = rowL$nodeNum)

    rowNL <- new("LinkDataFrame", rowL)

    ## update the row tree and links
    newDat <- BiocGenerics:::replaceSlots(tse,
```

```
                                            rowLinks = rowNL,
                                            rowTree = list(phylo = newTree))
    return(newDat)

}
```

Now the row tree has four leaves.

```
# find the mismatch between the rows of the 'assays' table and the leaves of the
# tree
row_tree <- rowTree(rte)
row_link <- rowLinks(rte)
leaf_tree <- showNode(tree = row_tree, only.leaf = TRUE)
leaf_data <- row_link$nodeNum[row_link$isLeaf]
leaf_rm <- setdiff(leaf_tree, leaf_data)
ntse <- updateRowTree(tse = rte, dropLeaf = leaf_rm)
```

```
ntse
## class: TreeSummarizedExperiment
## dim: 5 4
## metadata(0):
## assays(1): ''
## rownames(5): entity2 entity3 entity4 entity5 entity6
## rowData names(4): Kindom Phylum Class OTU
## colnames(4): A_1 A_2 B_1 B_2
## colData names(2): gg group
## reducedDimNames(0):
## altExpNames(0):
## rowLinks: a LinkDataFrame (5 rows)
## rowTree: a phylo (5 leaves)
## colLinks: a LinkDataFrame (4 rows)
## colTree: a phylo (4 leaves)
rowLinks(ntse)
## LinkDataFrame with 5 rows and 4 columns
##             nodeLab nodeLab_alias   nodeNum    isLeaf
##         <character>   <character> <integer> <logical>
## entity2          t3       alias_1         1      TRUE
## entity3          t2       alias_2         2      TRUE
## entity4          t1       alias_3         3      TRUE
## entity5          t5       alias_4         4      TRUE
## entity6          t4       alias_5         5      TRUE
```

2.2 Operation

The `TreeSummarizedExperiment` package can be installed by following the standard installation procedures of Bioconductor package.

```
# install BiocManager
install.packages("BiocManager")
```

```
# install TreeSummarizedExperiment package
BiocManager::install("TreeSummarizedExperiment")
```

Minimum system requirements is to have R with version (3.6 or later) on a Mac, Windows or Linux system.

3 Use Cases

To demonstrate the functionality of `TreeSummarizedExperiment`, we use it to store and manipulate a real microbial dataset. We further show exploratory graphics using available functions that are designed for the `SummarizedExperiment` class in other packages (e.g., *scrater*), or customized functions that are generated together with visualization packages (e.g., *ggplot2* (Wickham et al. 2020)).

```
suppressPackageStartupMessages({
  # Packages provide dataset
  library(HMP16SData)

  # Packages to manipulate data extracted from TreeSummarizedExperiment
  library(tidyr)
  library(dplyr)

  # Packages provide visualization
  library(ggplot2)
  library(TreeHeatmap)
  library(scales)
  library(ggtree)
  library(scater)
  library(cowplot)
  })
```

The Human Microbiome Project (HMP) 16S rRNA sequencing data is downloaded from the R package *HMP16SData*. It is collected from the variable regions 3–5 and is provided as a SummarizedExperiment object via the `ExperimentHub`.

```
v35 <- V35()
v35
## class: SummarizedExperiment
## dim: 45383 4743
## metadata(2): experimentData phylogeneticTree
## assays(1): 16SrRNA
## rownames(45383): OTU_97.1 OTU_97.10 ... OTU_97.9998 OTU_97.9999
## rowData names(7): CONSENSUS_LINEAGE SUPERKINGDOM ... FAMILY GENUS
## colnames(4743): 700013549 700014386 ... 700114717 700114750
## colData names(7): RSID VISITNO ... HMP_BODY_SUBSITE SRS_SAMPLE_ID
```

3.1 The storage of HMP 16S rRNA-seq data

We store the phylogenetic tree as the `rowTree`. Links between nodes of the tree and rows of `assays` are automatically generated in the construction of the `TreeSummarizedExperiment` object, and are stored as `rowLinks`. Rows of `assays` tables that mismatch with nodes of the tree are removed.

```
tse_phy <- TreeSummarizedExperiment(assays = assays(v35),
                                    rowData = rowData(v35),
                                    colData = colData(v35),
                                    rowTree = metadata(v35)$phylogeneticTree,
                                    metadata = metadata(v35)["experimentData"])
tse_phy
## class: TreeSummarizedExperiment
## dim: 45336 4743
## metadata(1): experimentData
## assays(1): 16SrRNA
## rownames(45336): OTU_97.1 OTU_97.10 ... OTU_97.9998 OTU_97.9999
## rowData names(7): CONSENSUS_LINEAGE SUPERKINGDOM ... FAMILY GENUS
## colnames(4743): 700013549 700014386 ... 700114717 700114750
## colData names(7): RSID VISITNO ... HMP_BODY_SUBSITE SRS_SAMPLE_ID
## reducedDimNames(0):
## altExpNames(0):
## rowLinks: a LinkDataFrame (45336 rows)
## rowTree: a phylo (45364 leaves)
## colLinks: NULL
## colTree: NULL

cD <- colData(tse_phy)
dim(table(cD$HMP_BODY_SITE, cD$RUN_CENTER))
## [1]  5 12
```

3.2 Replace the phylogenetic tree with the taxonomic tree

Here, we replace the phylogenetic with the taxonomic tree that is generated from the taxonomic table. Due to the existence of polyphyletic groups, a tree structure can't be really generated. For example, the `Ruminococcus` genus is from different families: `Lachnospiraceae` and `Ruminococcaceae`.

```
# taxonomic tree
# tax_0 <- rowData(tse_phy)[, -1] %>%
#   data.frame() %>%
#   mutate(OTU = rownames(tse_phy))
# reorder columns to have ranks from the Superkingdom to the consensus lineage
tax_0 <- data.frame(rowData(tse_phy))
ord_col <- colnames(tax_0)
tax_0 <- tax_0[, c(ord_col[-1], ord_col[1])]

tax_loop <- detectLoop(tax_tab = tax_0)
```

```
# show loops that are not caused by NA
no_na <- tax_loop$child != "NA" & tax_loop$parent != "NA"
tax_loop[no_na, ]
##                                      parent            child parent_column
## 35                          Alteromonadales Alteromonadaceae         ORDER
## 36                        Oceanospirillales Alteromonadaceae         ORDER
## 84                             Rhizobiales Rhodobacteraceae         ORDER
## 85                          Rhodobacterales Rhodobacteraceae         ORDER
## 86                             Chromatiales  Sinobacteraceae         ORDER
## 87                          Xanthomonadales  Sinobacteraceae         ORDER
## 88                           Bacteroidaceae      Bacteroides        FAMILY
## 89                          Lachnospiraceae      Bacteroides        FAMILY
## 90                          Ruminococcaceae      Bacteroides        FAMILY
## 91                           Clostridiaceae      Clostridium        FAMILY
## 92                       Erysipelotrichaceae      Clostridium        FAMILY
## 93                          Lachnospiraceae      Clostridium        FAMILY
## 94                          Ruminococcaceae      Clostridium        FAMILY
## 95  Clostridiales Family XIII. Incertae Sedis      Eubacterium        FAMILY
## 96                           Eubacteriaceae      Eubacterium        FAMILY
## 97                          Lachnospiraceae      Eubacterium        FAMILY
## 98                          Ruminococcaceae      Eubacterium        FAMILY
## 218                         Lachnospiraceae     Ruminococcus        FAMILY
## 219                         Ruminococcaceae     Ruminococcus        FAMILY
##     child_column
## 35        FAMILY
## 36        FAMILY
## 84        FAMILY
## 85        FAMILY
## 86        FAMILY
## 87        FAMILY
## 88         GENUS
## 89         GENUS
## 90         GENUS
## 91         GENUS
## 92         GENUS
## 93         GENUS
## 94         GENUS
## 95         GENUS
## 96         GENUS
## 97         GENUS
## 98         GENUS
## 218        GENUS
## 219        GENUS
```

To resolve the loops, we add suffix to the polyphyletic genus with `resolveLoop`. For example, `Ruminococcus` belonging to the `Lachnospiraceae` and the `Ruminococcaceae` families become `Ruminococcus_1` and `Ruminococcus_2`, respectively. A `phylo` tree is created afterwards using `toTree`.

```
tax_1 <- resolveLoop(tax_tab = tax_0)
tax_tree <- toTree(data = tax_1)
```

```
tax_tree
##
## Phylogenetic tree with 664 tips and 1079 internal nodes.
##
## Tip labels:
##  Root;p__Acidobacteria;c__Acidobacteria;o__Acidobacteriales;f__Acidobacteriaceae, Root;p__Acidobacteria;c_
## Node labels:
##  SUPERKINGDOM:Bacteria, PHYLUM:Acidobacteria, CLASS:Acidobacteria, ORDER:Acidobacteriales, FAMILY:Acidobac
##
## Unrooted; includes branch lengths.
```

Here, `changeTree` is used to replace the phylogenetic tree with the taxonomic tree and the link data is updated accordingly.

```
tse_tax <- changeTree(x = tse_phy, rowTree = tax_tree,
                      rowNodeLab = rowData(tse_phy)$CONSENSUS_LINEAGE)
```

3.3 Compare samples across centers

Here, we extract the sample information from the `colData`, and calculate the sequencing depths and the number of non-zero OTUs for samples using the data in `assays` table.

```
count <- assays(tse_phy)[[1]]

df_OTU <- colData(tse_phy) %>%
  data.frame() %>%
  mutate(nOTU = colSums(count > 0),
         Total_count = colSums(count))
```

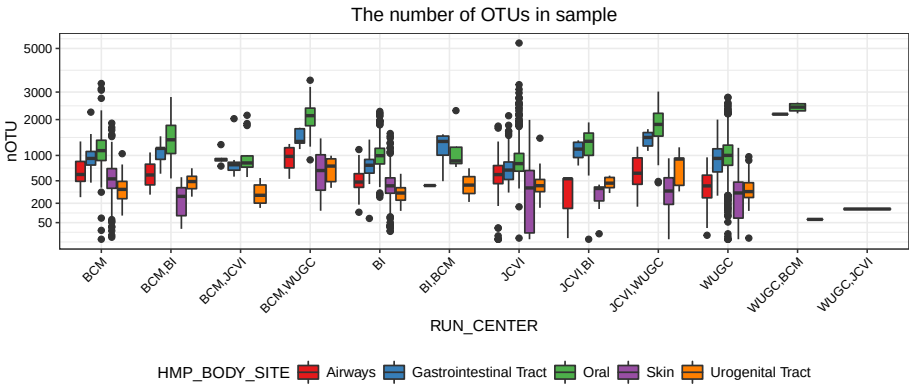
To make pretty figures from *ggplot2*, we customized a theme to be applied to several plots in this section.

```
# Customized the plot theme
prettify <- theme_bw(base_size = 10) + theme(
    panel.spacing = unit(0, "lines"),
    axis.text = element_text(color = "black"),
    axis.text.x = element_text(angle = 45, hjust = 1),
    legend.key.size= unit(6, "mm"),
    legend.spacing.x = unit(1, "mm"),
    plot.title = element_text(hjust = 0.5),
    legend.text = element_text(size = 9),
    legend.position="bottom",
    strip.background = element_rect(colour = "black", fill = "gray90"),
    strip.text.x = element_text(color = "black", size = 10),
    strip.text.y = element_text(color = "black", size = 10))
```

The number of OTUs are quite different in samples from different body sites.

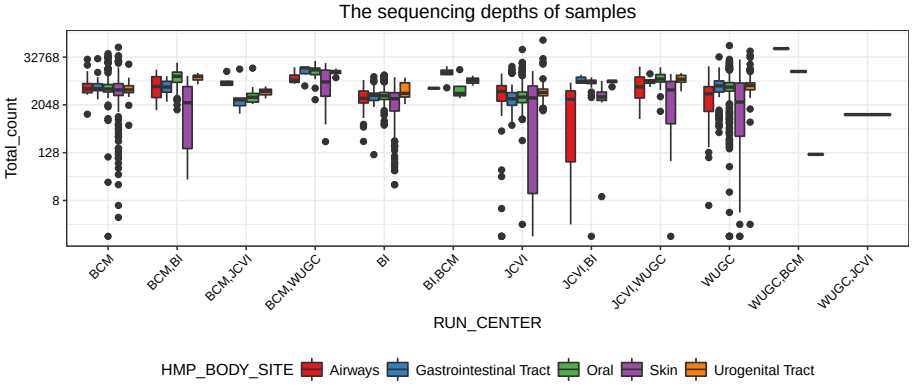
```
ggplot(df_OTU) +
  geom_boxplot(aes(x = RUN_CENTER, y = nOTU,
```

```
                    block = HMP_BODY_SITE, fill = HMP_BODY_SITE))+
  scale_y_sqrt(breaks = c(50, 200, 500, 1000, 2000, 3000, 5000)) +
  labs(title = "The number of OTUs in sample") +
  scale_fill_brewer(palette = "Set1") +
  prettify
```



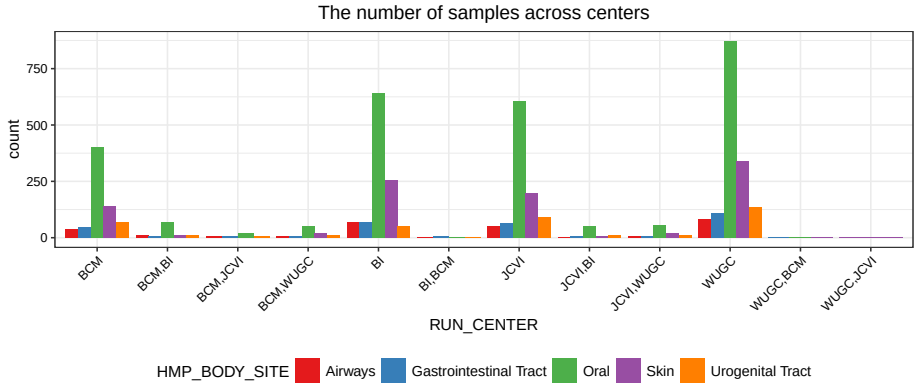
The sequencing depth of samples across different coordination centers are quite similar. Within the coordination center, samples collected from `Skin` are more spread out in the sequencing depth than those from other body sites.

```
ggplot(df_OTU) +
  geom_boxplot(aes(x = RUN_CENTER, y = Total_count,
                   block = HMP_BODY_SITE, fill = HMP_BODY_SITE)) +
  scale_y_continuous(trans = log2_trans()) +
  labs(title = "The sequencing depths of samples") +
  scale_fill_brewer(palette = "Set1") +
  prettify
```



More samples are taken from the `oral` site than other body sites

```
ggplot(df_OTU) +
  geom_bar(aes(RUN_CENTER, fill = HMP_BODY_SITE),
           position = position_dodge()) +
  labs(title = "The number of samples across centers") +
  scale_fill_brewer(palette = "Set1") +
  prettify
```



3.4 The relative abundance of phyla in samples

The relative abundance of OTUs within sample are calculated and stored in the `assays` with the name `rel_abund`.

```
# add relative abundance in the second assays
abd <- assays(tse_tax)[[1]]
assays(tse_tax)[["rel_abund"]] <- apply(abd, 2, FUN = function(x){
  x/sum(x)
})
```

We aggregate the relative abundance to the phylum level using `aggValue`. It calculate the relative abundance of a phylum as the sum of the relative abundance of OTUs belonging to it.

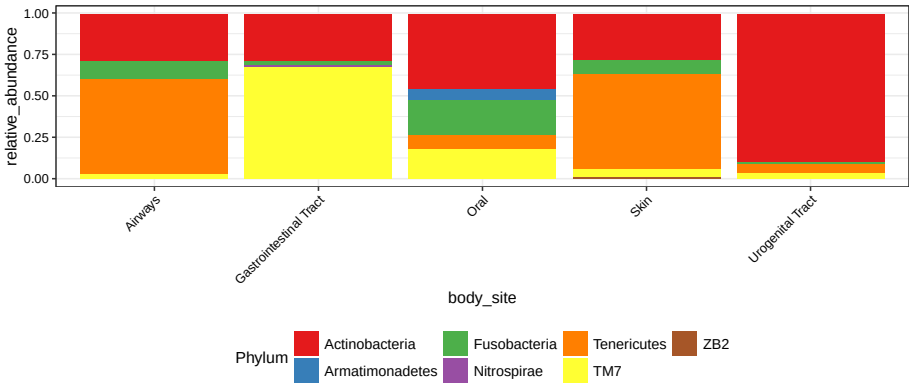
```
# aggregation to the phylum level
lab <- c(rowTree(tse_tax)$node.label, rowTree(tse_tax)$tip.label)
lab_phylum <- lab[startsWith(lab, "PHYLUM")]
agg_phylum <- aggValue(x = tse_tax,
                  rowLevel = lab_phylum,
                  assay = "rel_abund",
                  message = FALSE)
```

To compare the relative abundance of phyla across body sites, we average them over samples from the same body site, and show those with relative abundance above 0.01 in the scale $[0, 1]$.

```
aC <- assays(agg_phylum)$rel_abund
rD <- rowData(agg_phylum)
cD <- colData(agg_phylum)
rL <- rowLinks(agg_phylum)
df_abd <- t(aC) %>%
  data.frame() %>%
  mutate(body_site = cD$HMP_BODY_SITE,
         sample_id = colnames(aC)) %>%
  gather(key = "node", value = "rel_abund",
         -c(body_site, sample_id)) %>%
  group_by(body_site, node) %>%
  summarize(relative_abundance = mean(rel_abund)) %>%
```

```
    mutate(Phylum = rD$PHYLUM)

# The phylum level
ggplot(df_abd) +
  geom_bar(data = . %>% filter(relative_abundance >= 0.01),
           aes(x= body_site, y = relative_abundance, fill = Phylum),
           stat = "identity") +
  scale_fill_brewer(palette = "Set1") +
  prettify
```



3.5 Dimensionality reduction

We visualize samples in reduced dimensions to see whether those from the same body site are more similar. Three dimensionality reduction techniques, including principal component analysis (PCA), t-distributed Stochastic Neighbor Embedding (t-SNE), uniform manifold approximation and projection (UMAP) are used. As `TreeSummarizedExperiment` is inherited from `SingleCellExperiment`, we could directly use functions from the package *scater*. We first apply techniques using data at the OTU level, then we further try t-SNE using data at different taxonomic levels, e.g., the genus and the phylum levels, to see whether the resolution affects the separation of samples.

3.5.1 PCA

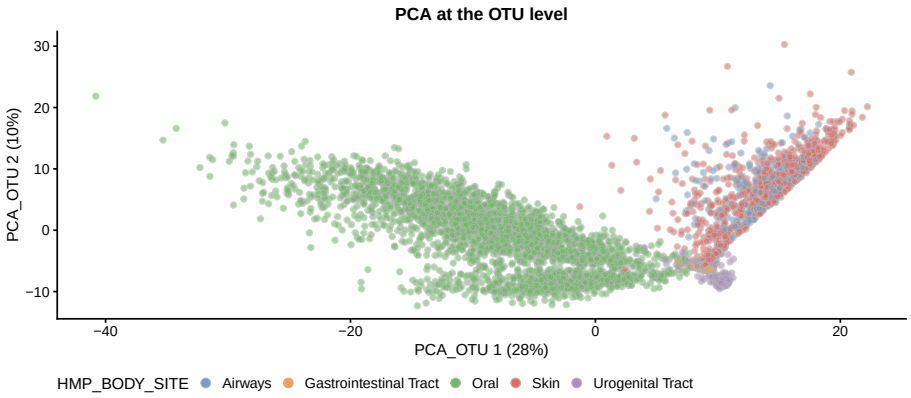
The PCA is performed on the log-transformed counts that are stored in the `assays` table with the name `logcounts`. We see that the `Oral` samples are distinct from those of other body sites. Samples from `Skin`, `Urogenital Tract`, `Airways` and `Gastrointestinal Tract` are not separated very well in the top two principle components of PCA.

```
# log-transformed data
assays(tse_tax)$logcounts <- log(assays(tse_tax)[[1]] + 1)

# run PCA at the OTU level
tse_tax <- runPCA(tse_tax, name="PCA_OTU", exprs_values = "logcounts")

# plot samples in the reduced dimensions
```

```
plotReducedDim(tse_tax, dimred = "PCA_OTU",
               colour_by = "HMP_BODY_SITE")+
  labs(title = "PCA at the OTU level") +
  guides(fill = guide_legend(override.aes = list(size=2.5, alpha = 1))) +
  theme(plot.title = element_text(hjust = 0.5),
        legend.position = "bottom")
```

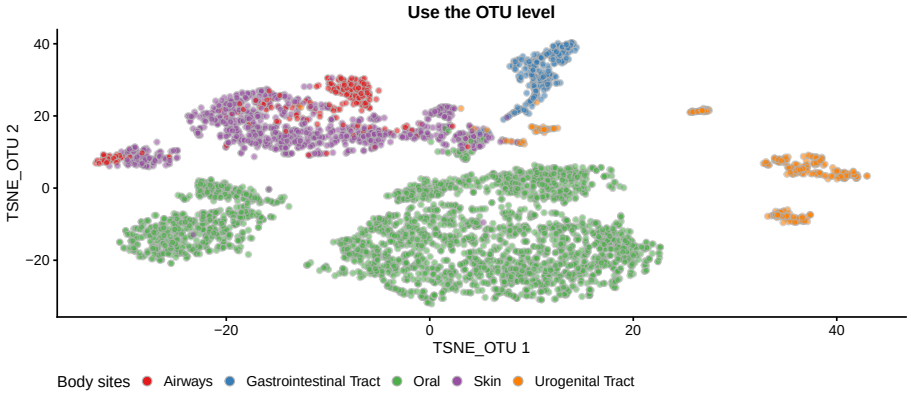


3.5.2 t-SNE

The separation is well improved with the use of `t-SNE`. Samples from `Oral`, `Gastrointestinal Tract`, and `Urogenital Tract` are in distinct clusters. Skin samples and airways samples still overlap.

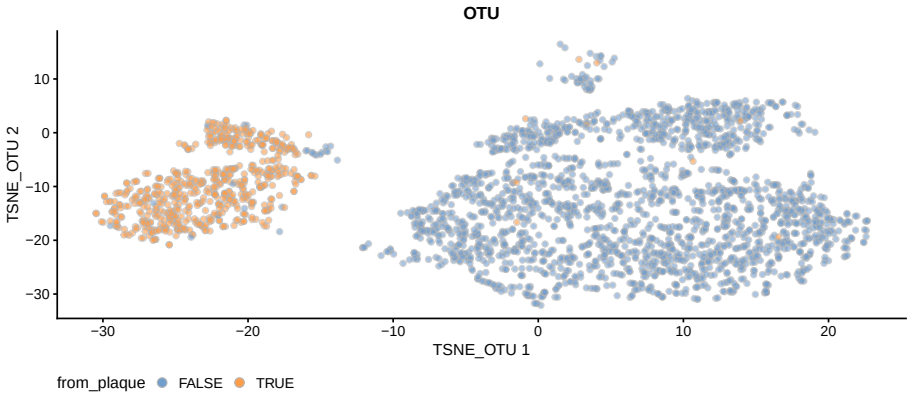
```
# run PCA at the OTU level
tse_tax <- runTSNE(tse_tax, name="TSNE_OTU")

# plot samples in the reduced dimensions
tsne_otu <- plotReducedDim(tse_tax, dimred = "TSNE_OTU",
                           colour_by = "HMP_BODY_SITE") +
  labs(title = "Use the OTU level") +
  theme(plot.title = element_text(hjust = 0.5)) +
  scale_fill_brewer(palette = "Set1") +
  labs(fill = "Body sites") +
  guides(fill = guide_legend(override.aes = list(size=2.5, alpha = 1))) +
  theme(plot.title = element_text(hjust = 0.5),
        legend.position = "bottom")
tsne_otu
```



Notably, there are two well separated clusters belonging to oral samples. The small cluster includes samples from the `Supragingival Plaque` and `Subgingival Plaque`, and the other cluster includes samples from other oral sub-sites.

```
is_oral <- colData(tse_tax)$HMP_BODY_SITE %in% "Oral"
colData(tse_tax)$from_plaque <- grepl(pattern = "Plaque",
                                      colData(tse_tax)$HMP_BODY_SUBSITE)
# Oral samples
plotReducedDim(tse_tax[, is_oral], dimred = "TSNE_OTU",
               colour_by = "from_plaque") +
  labs(title = "OTU") +
  guides(fill = guide_legend(override.aes = list(size=2.5, alpha = 1))) +
  theme(plot.title = element_text(hjust = 0.5),
        legend.position = "bottom")
```



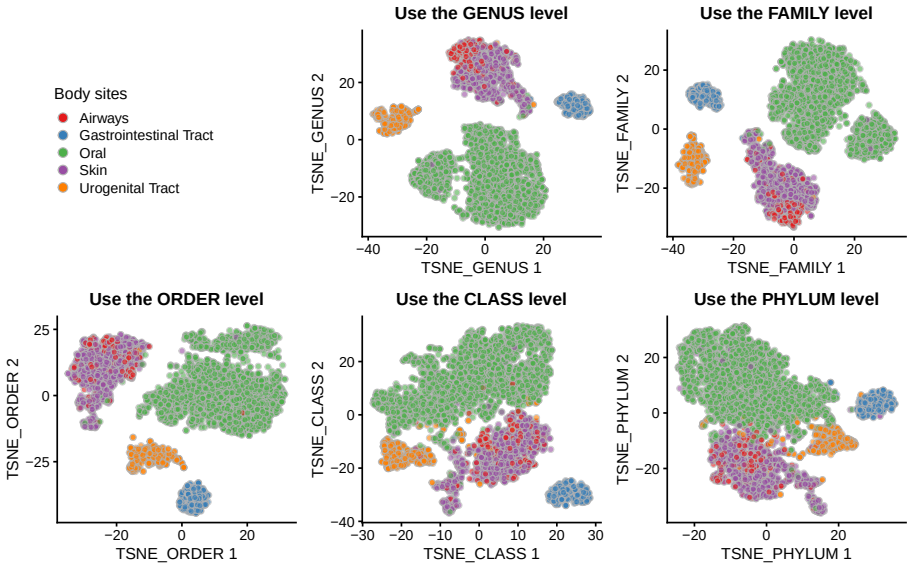
The separation of samples from different body sites become worse when the data on broader resolution is used.

```
# aggregation data to all internal nodes
tse_agg <- aggValue(x = tse_tax,
                    rowLevel = tax_tree$node.label,
                    assay = "16SrRNA",
                    message = FALSE)
# log-transform count
assays(tse_agg)$logcounts <- log(assays(tse_agg)[[1]] + 1)
```

```
tax_rank <- c("GENUS", "FAMILY", "ORDER", "CLASS", "PHYLUM")
names(tax_rank) <- tax_rank
fig_list <- lapply(tax_rank, FUN = function(x) {
  xx <- startsWith(rowLinks(tse_agg)$nodeLab, x)
  xx_tse <- runTSNE(tse_agg, name = paste0("TSNE_", x),
                    exprs_values = "logcounts",
                    subset_row = rownames(tse_agg)[xx])

  # plot samples in the reduced dimensions
  plotReducedDim(xx_tse, dimred = paste0("TSNE_", x),
                 colour_by = "HMP_BODY_SITE") +
    labs(title = paste0("Use the ", x, " level")) +
    theme(plot.title = element_text(hjust = 0.5))+
    scale_fill_brewer(palette = "Set1") +
    theme(legend.position = "none") +
  guides(fill = guide_legend(override.aes = list(size=2.5)))
})
```

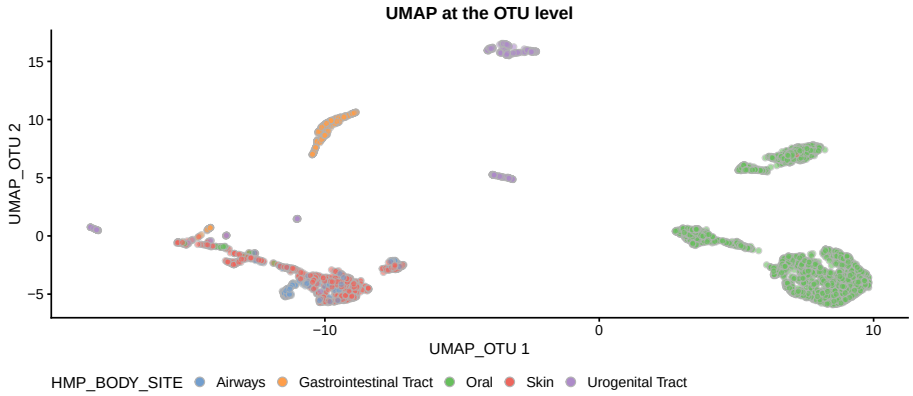
```
legend <- get_legend(
  # create some space to the left of the legend
  tsne_otu +
    theme(legend.box.margin = margin(0, 0, 0, 35),
          legend.position = "right")
  )
plot_grid(plotlist = fig_list,
          legend, nrow = 2)
```



3.5.3 UMAP

```
# run UMAP at the OTU level
tse_tax <- runUMAP(tse_tax, name="UMAP_OTU")
```

```
# plot samples in the reduced dimensions
plotReducedDim(tse_tax, dimred = "UMAP_OTU",
               colour_by = "HMP_BODY_SITE") +
  labs(title = "UMAP at the OTU level") +
  guides(fill = guide_legend(override.aes = list(size=2.5, alpha = 1))) +
  theme(plot.title = element_text(hjust = 0.5),
        legend.position = "bottom")
```



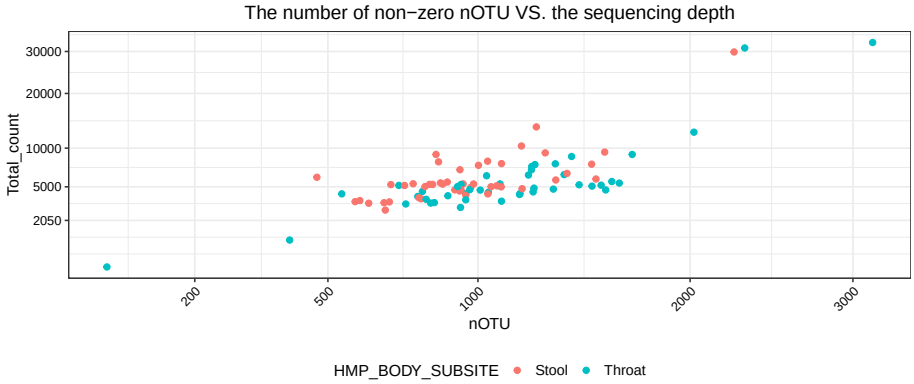
3.6 Fliter samples and OTUs

When preprocessing data, we might want to remove some samples or OTUs for some reasons. For example, if we are only interested in `Stool` and `Throat` samples provided by the center `BCM`.

```
count <- assays(tse_tax)[[1]]
df_BCM <- colData(tse_tax) %>%
  data.frame() %>%
  mutate(nOTU = colSums(count > 0),
         Total_count = colSums(count)) %>%
  mutate(index_sample = row_number()) %>%
  dplyr::filter(RUN_CENTER %in% "BCM") %>%
  dplyr::filter(HMP_BODY_SUBSITE %in% c("Stool", "Throat"))
```

We further remove samples that have relatively lower number of OTUs (< 200) and sequencing depths (< 2050).

```
# The number of OTUs  V.S. The sequencing depth
ggplot(df_BCM) +
  geom_point(aes(x = nOTU, y = Total_count,
                 color = HMP_BODY_SUBSITE))+
  scale_x_sqrt(breaks = c(200, 500, 1000, 2000, 3000)) +
  scale_y_sqrt(breaks = c(2050, 5000, 10000, 20000, 30000)) +
  labs(title = "The number of non-zero nOTU VS. the sequencing depth") +
  prettify
```




```
# samples that are kept
sel <- df_BCM %>%
  dplyr::filter(nOTU > 200 & Total_count > 2050) %>%
  select(index_sample) %>%
  unlist()

tse_BCM <- tse_tax[, sel]
```

OTUs that appear in less than 10 of samples are removed, and their corresponding leaves on the tree are dropped.

```
# remove OTUs
count <- assays(tse_BCM)[[1]]
isRare <- rowSums(count>0) < 0.1*ncol(tse_BCM)
tse_BCM <- tse_BCM[!isRare, ]

# drop leaves of the removed OTUs from the tree
old_tree <- rowTree(tse_BCM)
rmTip <- setdiff(old_tree$tip.label, rowLinks(tse_BCM)$nodeLab)
new_tree <- drop.tip(phy = old_tree, tip = rmTip,
                     trim.internal = TRUE,
                     collapse.singles = FALSE)

# update the TreeSumamrizedExperiment
tse_BCM <- changeTree(x = tse_BCM, rowTree = new_tree,
                      rowNodeLab = rowData(tse_BCM)$CONSENSUS_LINEAGE)
tse_BCM
## class: TreeSummarizedExperiment
## dim: 3243 88
## metadata(1): experimentData
## assays(3): 16SrRNA rel_abund logcounts
## rownames(3243): OTU_97.10033 OTU_97.1005 ... OTU_97.997 OTU_97.9994
## rowData names(7): CONSENSUS_LINEAGE SUPERKINGDOM ... FAMILY GENUS
## colnames(88): 700013549 700016542 ... 700111985 700111996
## colData names(8): RSID VISITNO ... SRS_SAMPLE_ID from_plaque
## reducedDimNames(3): PCA_OTU TSNE_OTU UMAP_OTU
## altExpNames(0):
## rowLinks: a LinkDataFrame (3243 rows)
## rowTree: a phylo (96 leaves)
```

```
## colLinks: NULL
## colTree: NULL
```

3.7 Heatmap at different taxonomic levels

The relative abundance of taxa could be visualized at different taxonomic levels. At the phylum level, stool samples have higher *Bacteroidetes* and lower *Firmicutes* than throat samples. When visualizing data on a higher resolution (e.g., the genus level), heterogeneities are seen within the same phylum. For example, the *Prevotella* has lower relative abundance in stool samples but *Parabacteroides* is the other way around even they belong to the same phylum *Bacterodetes*.

We first calculate the relative abundance of OTUs within the sample, and then aggregate to all internal nodes that represent taxa at different levels.

```
# store the relative abundance in a table of assays
assays(tse_BCM)[["rel_abund"]] <- apply(assays(tse_BCM)[[1]], 2,
                                       FUN = function(x){
                                         x/sum(x)
                                       })

# aggregation: all internal nodes
agg_BCM <- aggValue(x = tse_BCM,
                    rowLevel = rowTree(tse_BCM)$node.label,
                    FUN = sum)
```

Nodes (`lab_phylum`) representing taxa at the phylum level are obtained, and their relative abundances in different samples (`mat_phylum`) are extracted.

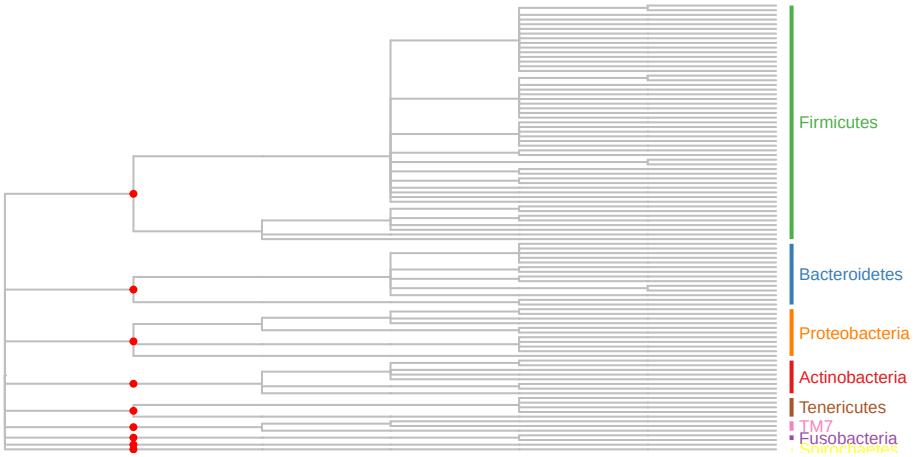
```
# Extract the data at the phylum level
lab <- c(rowTree(agg_BCM)$node.label, rowTree(agg_BCM)$tip.label)
lab_phylum <- lab[startsWith(lab, "PHYLUM")]
mat_phylum <- agg_BCM %>%
  subsetByNode(rowNode = lab_phylum) %>% assays %>%
  nth(2) %>%
  data.frame(check.names = FALSE)
```

Then, we plot the tree and label nodes representing phyla using *ggtree*

```
# Plot the tree figure
treeFig <- ggtree(rowTree(agg_BCM), branch.length = "none",
                  color = "grey") +
  geom_point2(aes(subset = label %in% lab_phylum),
              color = "red", size  = 1.3) +
    expand_limits(x = 7.5)

clade_phylum <- gsub(pattern = ".*:", "", lab_phylum)
node_phylum <- transNode(tree = rowTree(agg_BCM), node = lab_phylum)
colrs <- brewer_pal(palette = "Set1")(8)
for (i in seq_along(node_phylum)) {
  treeFig <- treeFig +
```

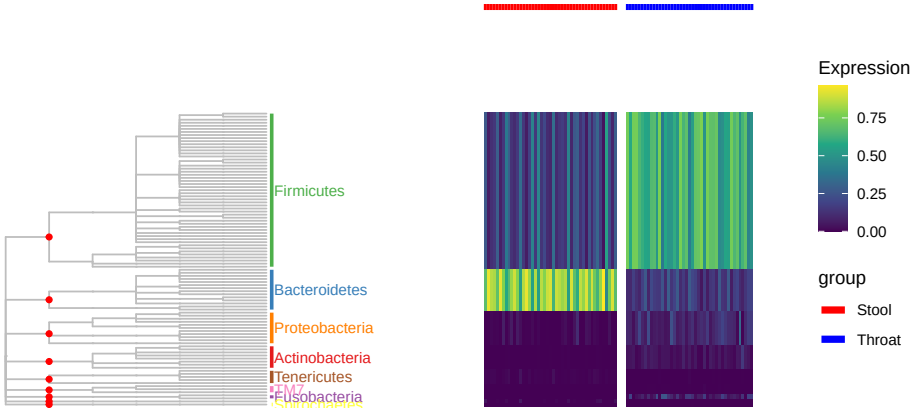
```
        geom_cladelabel(node  = node_phylum[i], label = clade_phylum[i],
                        fontsize = 3, color = colrs[i],
                        barsize = 1)
    }
    treeFig
```



The relative abundance of phyla are visualized as a heatmap using *TreeHeatmap*. Samples from throat and stool are split as shown in Figure **??**.

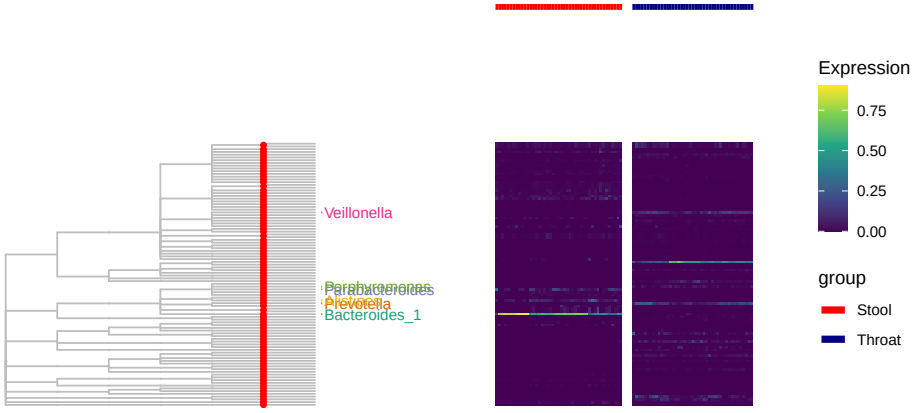
```
    # label the body subsites of samples
    anno_c <- setNames(colData(agg_BCM)$HMP_BODY_SUBSITE,
                       rownames(colData(agg_BCM)))

    ## align heatmap to the tree
    fig1 <- TreeHeatmap(tree = rowTree(agg_BCM),
                hm_data = mat_phylum,
                tree_fig = treeFig,
                tree_hm_gap = 5,
                column_split = anno_c,
                column_anno = anno_c,
                column_anno_color = c(Stool = "red", Throat = "blue"),
                column_anno_size = 2,
                column_anno_gap = 35)
    fig1
```



The heatmap at the geneus level of the tree could be generated in the same way. To avoid the repetition, we wrap the codes as a function `plotAssay` shown after the Figure **??**. Users could use it as a template or example to customize visualization function for the class `TreeSummarizedExperiment`.

```
sel_genus <- c("GENUS:Bacteroides_1", "GENUS:Prevotella",
               "GENUS:Parabacteroides", "GENUS:Veillonella",
               "GENUS:Porphyromonas", "GENUS:Alistipes")
fig2 <- plotAssay(x = agg_BCM,
                  taxo_level = "GENUS",
                  assay = 2,
                  split_by = "HMP_BODY_SUBSITE",
                  clade_label = sel_genus,
                  clade_fontsize = 3,
                  clade_color = brewer_pal(palette = "Dark2")(6),
                  gap_tree_heatmp = 3.5,
                  rel_width = 0.8,
                  anno_gap = 50,
                  anno_color = c("red", "navy"))
fig2
```




```
plotAssay <- function(x,
                      taxo_level = "PHYLUM", assay = 1,
                      tree_fig = NULL, color_by = NULL,
                      split_by = NULL,
```

```
                        anno_gap = 10, anno_color = NULL,
                        clade_label = NULL,
                        clade_fontsize = 4,
                        clade_color = NULL,
                        gap_tree_heatmp = 1, rel_width = 1) {

  # Tree
  rTree <- rowTree(x)
  lab <- c(rTree$tip.label, rTree$node.label)

  # level
  isLev <- startsWith(lab, taxo_level)
  if (!sum(isLev)) {
    stop("Can't find nodes corresponding to the ", taxo_level, " level.")
  }
  lev <- lab[isLev]

  # assays
  sx <- subsetByNode(x = x, rowNode = lev)
  if (any(!dim(sx))) {
    stop("No data is available for plot")
  }
  cx <- assays(sx)[[assay]]
  df_x <- data.frame(cx, check.names = FALSE)

  # Tree figure
  if (is.null(tree_fig)) {
    tree_fig <- ggtree(rTree, branch.length = "none", color = "grey") +
      geom_point2(aes(subset = (label %in% lev)), color = "red",
                  size = 1.2)
  }
  if (!is.null(clade_label)) {
     clade_node <- transNode(tree = rTree, node = clade_label)
     if (is.null(clade_color)) {
       clade_color <- rep("black", length(clade_label))
     }
     if (length(clade_color) != length(clade_color)) {
       warning("clade_color has different lengths to clade_node.")
     }
     clade_label <- gsub(pattern = ".*:", "", clade_label)
     for (i in seq_along(clade_node)) {
       tree_fig <- tree_fig +
         geom_cladelabel(node  = clade_node[i], label = clade_label[i],
                         fontsize = clade_fontsize, color = clade_color[i])
     }
  }
  # split by
  colD <- colData(sx)
  if (!is.null(split_by)) {
    split_v <- colD[[split_by]]
    names(split_v) <- colnames(x)
```

```
  } else {
    split_v <- NULL
  }

  # column annotation
  if (!is.null(anno_color)) {
    un <- unique(colD[[split_by]])
    if (length(anno_color) != length(un)) {
      stop("anno_color has length: ", length(anno_color), "; ",
           length(un) , " colors are expected")
    }
    names(anno_color) <- un
  }
  # column annotation
  f_heat <- TreeHeatmap(hm_data = df_x,
                        tree = rTree,
                        tree_hm_gap = gap_tree_heatmp,
                        tree_fig = tree_fig,
                        column_split = split_v,
                        column_split_gap = 0.2,
                        column_anno = split_v,
                        column_anno_gap = anno_gap,
                        column_anno_size = 4,
                        column_anno_color = anno_color,
                        rel_width = rel_width,
                        cluster_column = TRUE,
                        colnames_angle = 45)
  f_heat

}
```

4 Summary

This section is required if the paper does not include novel data or analyses. It allows authors to briefly summarize the key points from the article.

5 Software availability

The TreeSummarizedExperiment package is available at https://www.bioconductor.org/packages/release/bioc/html/TreeSummarizedExperiment.html

Source code of the development version of the package is available at https://github.com/fionarhuang/TreeSummarizedExperiment

6 Author information

Ruizhu Huang Roles: Conceptualization, Software, Writing – Original Draft Preparation, Writing – Review & Editing

Charlotte Soneson Roles: Conceptualization, Supervision, Writing – Review & Editing

Mark D Robinson Roles: Conceptualization, Supervision, Writing – Review & Editing

7 Competing interests

No competing interests were disclosed.

8 Grant information

Please state who funded the work discussed in this article, whether it is your employer, a grant funder etc. Please do not list funding that you have that is not relevant to this specific piece of research. For each funder, please state the funder's name, the grant number where applicable, and the individual to whom the grant was assigned. If your work was not funded by any grants, please include the line: 'The author(s) declared that no grants were involved in supporting this work.'

9 Acknowledgments

We thank Héctor Corrada Bravo, Levi Waldron, Hervé Pagès, Martin Morgan, Ernst Felix G.M., Federico Marini, Jayaram Kancherla, Domenick Braccia, Guangchuang YU, Vince Carey, Kasper D Hansen, Kévin Rue-Albrecht, Davide Risso, Stephanie Hicks, Daniel van Twisk, Marcel Ramos and other members of the Bioconductor community for their helpful suggestions.

Lun, Aaron, and Davide Risso. 2020. *SingleCellExperiment: S4 Classes for Single Cell Data*.

Morgan, Martin, Valerie Obenchain, Jim Hester, and Hervé Pagès. 2020. *SummarizedExperiment: SummarizedExperiment container*.

Paradis, E, and K Schliep. 2019. "ape 5.0: an environment for modern phylogenetics and evolutionary analyses in {R}." *Bioinformatics* 35: 526–28.

Wang, Li-Gen, Tommy Tsan-Yuk Lam, Shuangbin Xu, Zehan Dai, Lang Zhou, Tingze Feng, Pingfan Guo, et al. 2019. "Treeio: An R Package for Phylogenetic Tree Input and Output with Richly Annotated and Associated Data." *Molecular Biology and Evolution* 37 (2): 599–603. https://doi.org/10.1093/molbev/msz240.

Wickham, Hadley, Winston Chang, Lionel Henry, Thomas Lin Pedersen, Kohske Takahashi, Claus Wilke, Kara Woo, Hiroaki Yutani, and Dewey Dunnington. 2020. *ggplot2: Create Elegant Data Visualisations Using the Grammar of Graphics*. https://cran.r-project.org/package=ggplot2.

Yu, Guangchuang, David K. Smith, Huachen Zhu, Yi Guan, and Tommy Tsan Yuk Lam. 2017. "Ggtree: An R Package for Visualization and Annotation of Phylogenetic Trees with Their Covariates and Other Associated Data." *Methods in Ecology and Evolution* 8 (1): 28–36. https://doi.org/10.1111/2041-210X.12628.