Parsing, manipulating, and visualizing taxonomic data in R

Zachary S. L. Foster

September 20, 2017

Parsing, manipulating, and visualizing taxonomic data in R

Motivation:

- High-throughput sequencing of environmental DNA has led to large taxonomic data sets
- The hierarchical nature of taxonomic classifications make manipulating them and associated data difficult

The `taxa` package:

- Flexible reading of most sources of taxonomic information
- `dplyr`-inspired filtering and subsetting of taxa and associated information
- A standard set of classes for taxonomic data

The `metacoder` package:

- Simple reading of specific file formats commonly used in metabarcoding
- Flexible, information-rich plotting of taxonomic data
- Functions for analyzing community taxonomic diversity

Parsing taxonomic data

Most sources of taxonomic data in most formats can be parsed.

Input data format

Input type

	Simple	Embedded	Raw string

	`> print(data)`
	`[1] "input_1" "input_2"`
	`[3] "input_3"`
		`> print(data)`
		`  x   input   y`
		`1 a input_1 100`
		`2 b input_2 200`
		`3 c input_3 300`
			`> print(data)`
			`[1] ">id:a-tax:input_1"`
			`[2] ">id:b-tax:input_2"`
			`[3] ">id:c-tax:input_3"`

Classification
`Primates;Hominidae;Homo;sapiens`
	`> print(data)`
	`[1] "Primates;Hominidae;Hom...`
	`[2] "Primates;Haplorhini;Cr...`
	
	`> parse_tax_data(data,`
	`  class_sep = ";")`
		`> print(data)`
		`  x                 class   y`
		`1 a Primates;Hominidae;... 100`
		`2 b Primates;Haplorhini... 200`
		
		`> parse_tax_data(data,`
		`  class_cols = "class",`
		`  class_sep = ";")`
			`> print(data)`
			`[1] ">id:a-tax:Primates;Hom..."`
			`[2] ">id:b-tax:Primates;Hap..."`
			
			`> extract_tax_data(data,`
			`  regex = ">id:(.+)-tax:(.+)",`
			`  key = c("info", "class"),`
			`  class_sep = ";")`

Taxon ID
`9606`
	`> print(data)`
	`[1] "9606" "100937" ...`
	
	`> lookup_tax_data(data,`
	`  type = "taxon_id")`
		`> print(data)`
		`  x     id   y`
		`1 a 9606   100`
		`2 b 100937 200`
		
		`> lookup_tax_data(data,`
		`  type = "taxon_id",`
		`  column = "id")`
			`> print(data)`
			`[1] ">id:a-tax:9606"`
			`[2] ">id:b-tax:100937"`
			
			`> extract_tax_data(data,`
			`  regex = ">id:(.+)-tax:(.+)",`
			`  key = c("info", "taxon_id"),`
			`  database = "ncbi")`

Taxon name
`Homo sapiens`
	`> print(data)`
	`[1] "Homo sapiens"`
	`[2] "Primates" ...`
	
	`> lookup_tax_data(data,`
	`  type = "taxon_name")`
		`> print(data)`
		`  x         name   y`
		`1 a Homo sapiens 100`
		`2 b Primates     200`
		
		`> lookup_tax_data(data,`
		`  type = "taxon_name",`
		`  column = "name")`
			`> print(data)`
			`[1] ">id:a-tax:Homo sapiens"`
			`[2] ">id:b-tax:Primates"`
			
			`> extract_tax_data(data,`
			`  regex = ">id:(.+)-tax:(.+)",`
			`  key = c("info","taxon_name"),`
			`  database = "ncbi")`

Sequence ID
`AC073210`
	`> print(data)`
	`[1] "AC073210" "KC312885" ...`
	
	`> lookup_tax_data(data,`
	`  type = "seq_id")`
		`> print(data)`
		`  x  ncbi_id   y`
		`1 a AC073210 100`
		`2 b KC312885 200`
		
		`> lookup_tax_data(data,`
		`  type = "seq_id",`
		`  column = "ncbi_id")`
			`> print(data)`
			`[1] ">id:a-tax:AC073210"`
			`[2] ">id:b-tax:KC312885"`
			
			`> extract_tax_data(data,`
			`  regex = ">id:(.+)-tax:(.+)",`
			`  key = c("info","seq_id"),`
			`  database = "ncbi")`

Parsing taxonomic data: Embedded classifications

Embedded classifications often appear in abundance matrices and FASTA headers and are the best source of taxonomic information.

Any extra observation info that should be preserved in the parsed data can be added to the `key` option as `"info"`.

The code below parses the Mothur 16s RDP training set.

```
library(taxa)

seqs <- ape::read.FASTA("trainset14_032015.rdp.fasta")

cat(names(seqs)[1])

## AB294171_S001198039  Root;Bacteria;Firmicutes;Bacilli;Lactobacillales;Carnobacteriaceae

data <- extract_tax_data(names(seqs)[1:1000],
                         regex = "(.*)\\t(.*)",
                         key = c(rdp_id = "info", my_class = "class"),
                         class_sep = ";")
```

Parsing taxonomic data: Embedded classifications

```
print(data)
```

```
## <Taxmap>
##   955 taxa: aab. Root, aac. Bacteria ... bks. Phycicoccus, bkt. Herbiconiux
##   955 edges: NA->aab, aab->aac, aab->aad ... bhs->bkr, bgw->bks, bgv->bkt
##   1 data sets:
##     tax_data:
##       # A tibble: 1,000 x 4
##         taxon_id              rdp_id
##            <chr>               <chr>
##       1      aqb AB294171_S001198039
##       2      bht AB243007_S000622964
##       3      aqd AJ717394_S000623308
##       # ... with 997 more rows, and 2 more variables: my_class <chr>, input <chr>
##   0 functions:
```

Parsing taxonomic data: Genbank accession numbers

GenBank accession numbers can be used to look up the NCBI taxonomy for sequences.

However, this can be quite slow since this information must be queried from NCBI's servers, which are heavily used.

```
ids <- c("JQ086376.1", "AM946981.2", "JQ182735.1", "CP001396.1", "J02459.1",
         "AC150248.3", "X64334.1", "CP001509.3", "CP006698.1", "AC198536.1")
contaminants <- lookup_tax_data(ids, type = "seq_id")
print(contaminants)
```

```
## <Taxmap>
##   32 taxa: 10239. Viruses ... 1385755. synthetic Escherichia coli C321.deltaA
##   32 edges: NA->10239, NA->131567 ... 83333->511145, 511145->1385755
##   2 data sets:
##     tax_data:
##       # A tibble: 32 x 4
##         taxon_id                    ncbi_name    ncbi_rank ncbi_id
##            <chr>                        <chr>        <chr>   <chr>
##       1    10239                      Viruses superkingdom   10239
##       2    35237 dsDNA viruses, no RNA stage      no rank   35237
##       3    28883                Caudovirales        order   28883
##       # ... with 29 more rows
##     query_data:  JQ086376.1, AM946981.2 ... CP001509.3, CP006698.1, AC198536.1
##   0 functions:
```

Parsing taxonomic data: Taxon names

Taxon names can also be used to look up complete classifications.

Which taxon names are valid depends on the database used.

```
taxon_names <- c("Acrobolbaceae", "Adelanthaceae", "Allisoniaceae", "Amblystegiaceae",
                 "Andreaeaceae", "Andreaeobryaceae", "Aneuraceae", "Antheliaceae")
```

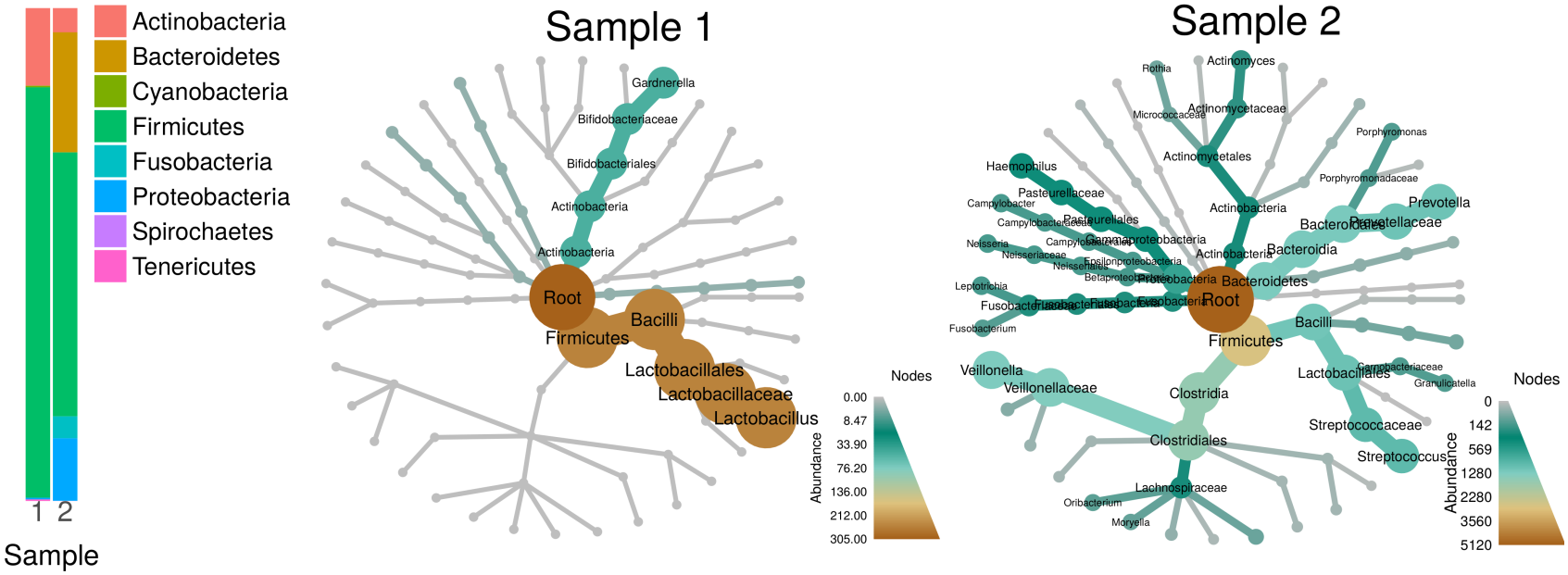
Note that a different database is being used this time: The Integrated Taxonomic Information System.

```
bryophytes <- lookup_tax_data(taxon_names, type = "taxon_name", database = "itis")
print(bryophytes)
```

```
## <Taxmap>
##   31 taxa: 202422. Plantae ... 15416. Allisoniaceae, 14406. Antheliaceae
##   31 edges: NA->202422, 202422->954898 ... 846198->15416, 846194->14406
##   2 data sets:
##     tax_data:
##       # A tibble: 31 x 4
##         taxon_id     itis_name    itis_rank itis_id
##            <chr>         <chr>        <chr>   <chr>
##       1   202422       Plantae      kingdom  202422
##       2   954898 Viridiplantae   subkingdom  954898
##       3   846494  Streptophyta infrakingdom  846494
##       # ... with 28 more rows
##     query_data:  Acrobolbaceae, Adelanthaceae ... Aneuraceae, Antheliaceae
##   0 functions:
```

Plotting taxonomic data with metacoder

`metacoder` uses the classes defined by `taxa` to plot taxonomic information.

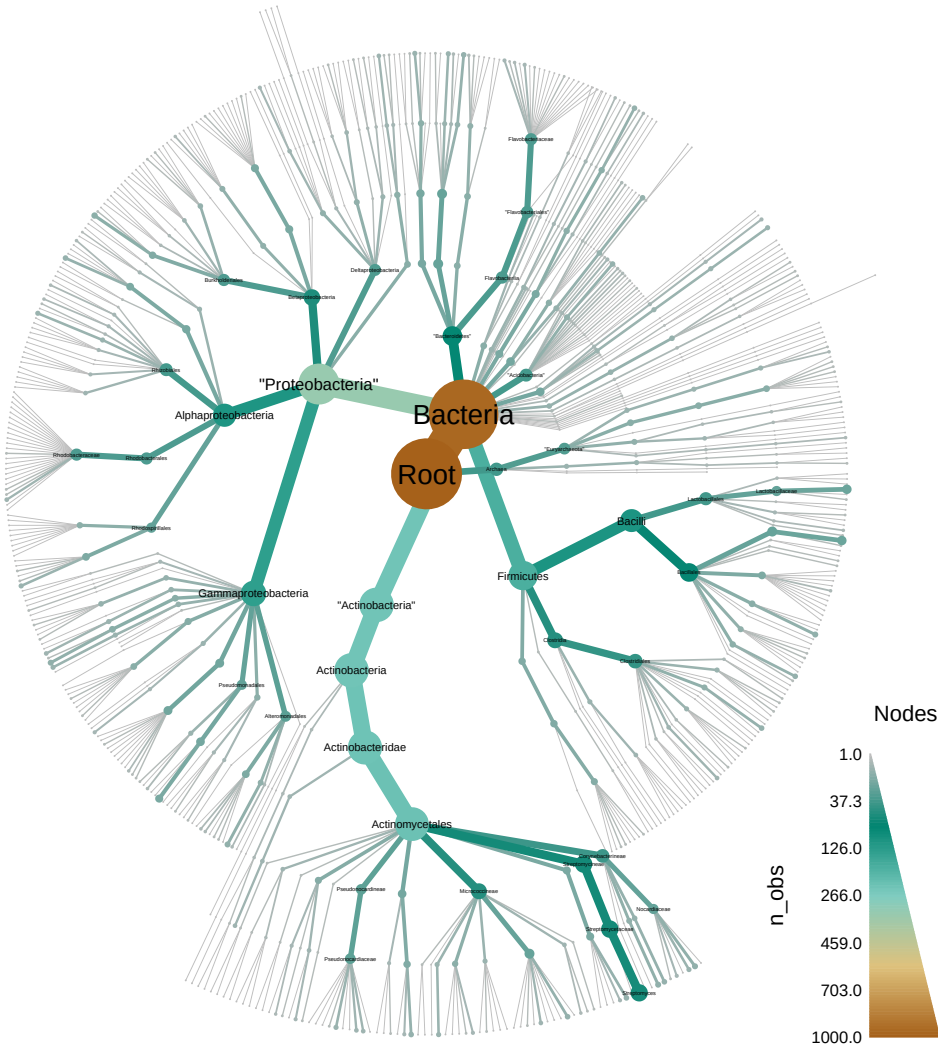


Plotting taxonomic data with metacoder

Parsed data can be plotted using the `heat_tree` function.

Any statistic can be mapped to the size and color of nodes and edges.

```
library(metacoder)
heat_tree(data, node_size = n_obs, node_label = taxon_names, node_color = n_obs)
```



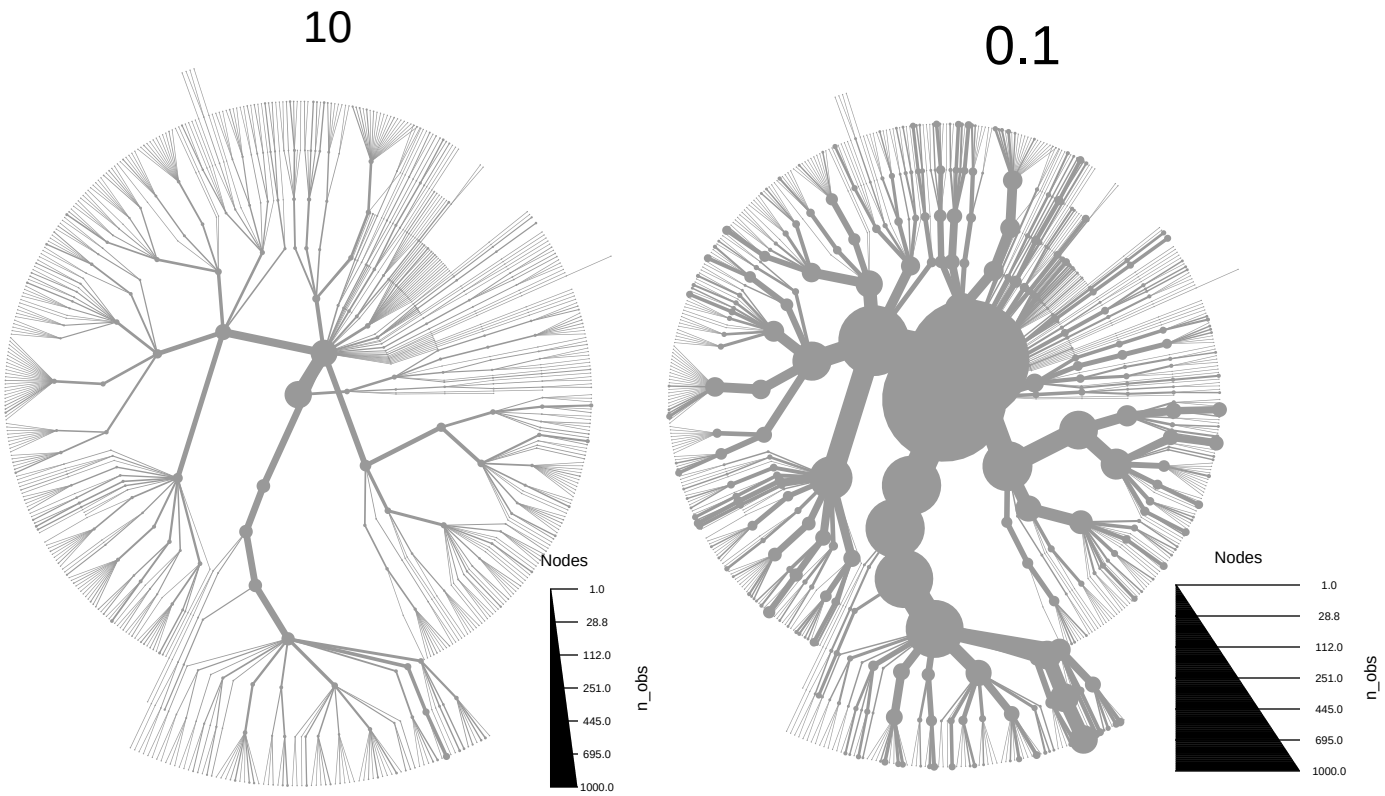
Plotting taxonomic data: Overlap optimization

The size range of nodes is optimized for each graph by default to avoid overlaps while maximizing size range.

The balance between the two goals can be modified with the `overlap_avoidance` option.

```
heat_tree(data, node_size = n_obs, overlap_avoidance = 10, title = "10")

heat_tree(data, node_size = n_obs, overlap_avoidance = 0.1, title = "0.1")
```

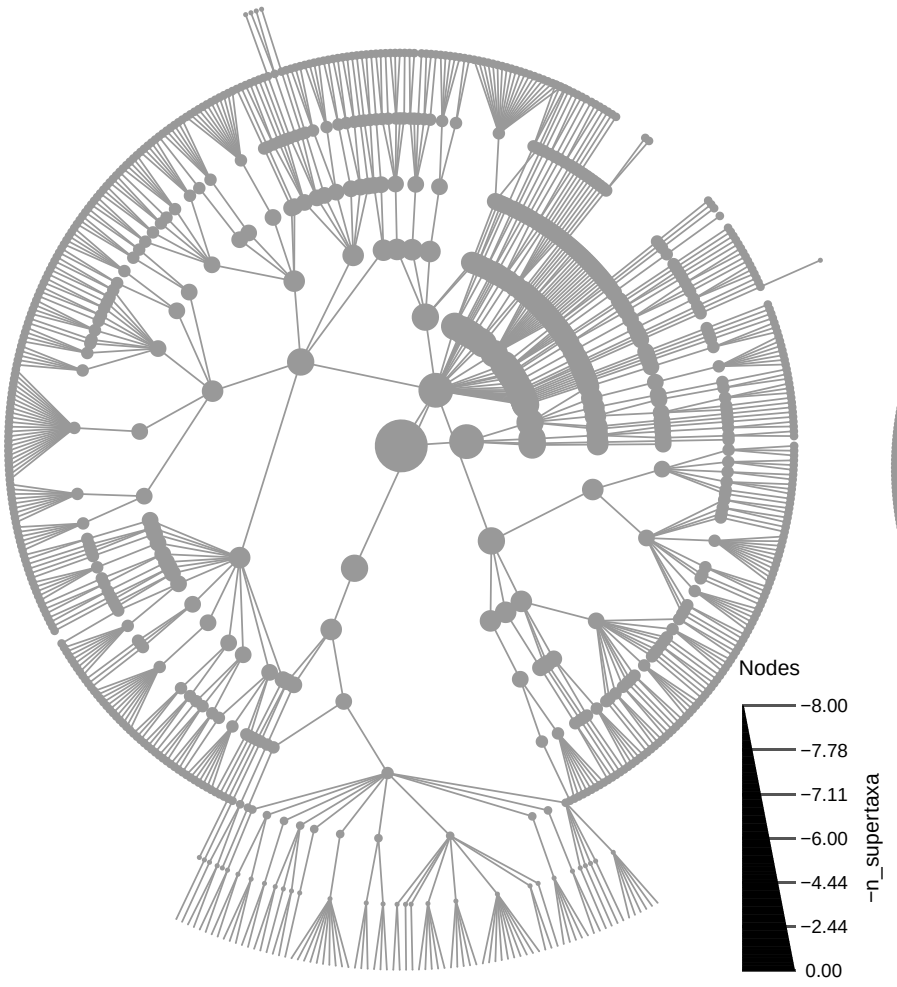


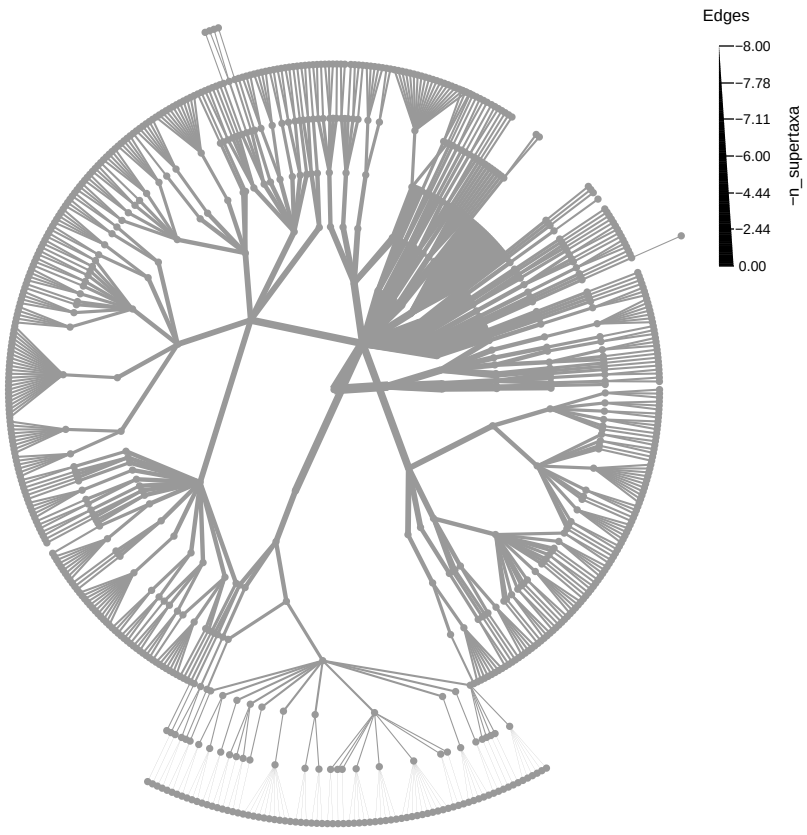
Plotting taxonomic data: Size

The size range of nodes and edges can also be specified manually.

```
heat_tree(data, node_size = - n_supertaxa, node_size_range = c(0.001, 0.03),
          edge_size_range = c(0.001, 0.001))

heat_tree(data, edge_size = - n_supertaxa, edge_size_range = c(0.0001, 0.01),
          node_size_range = c(0.005, 0.005))
```





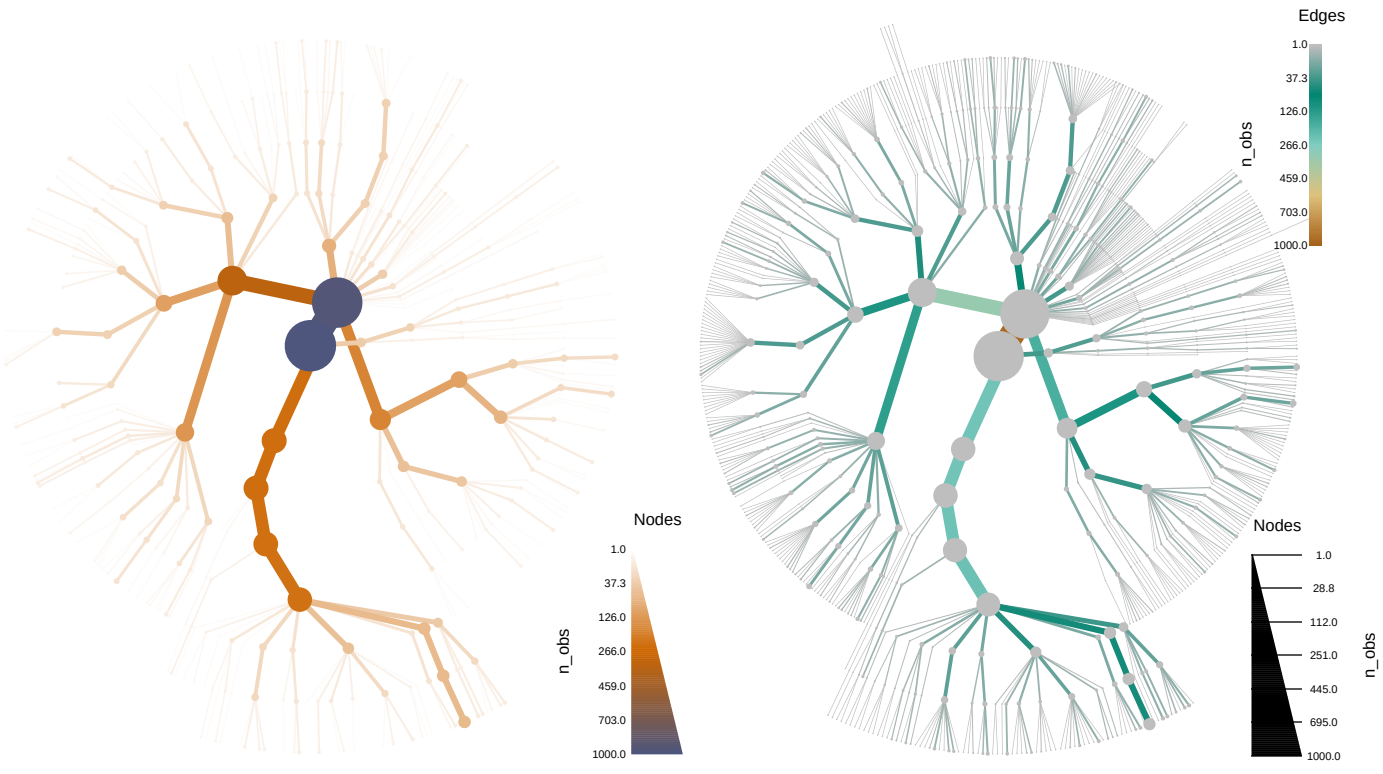
Plotting taxonomic data: Color

Mapping statistics to color works like it does for size.

A custom color range can be specified with `node_color_range` and `edge_color_range`.

```
heat_tree(data, node_size = n_obs, node_color = n_obs,
          node_color_range = c("#FFFFFF", "darkorange3", "#4e567d"))

heat_tree(data, node_size = n_obs, node_color = "grey",
          edge_color = n_obs)
```



Plotting taxonomic data: Labels

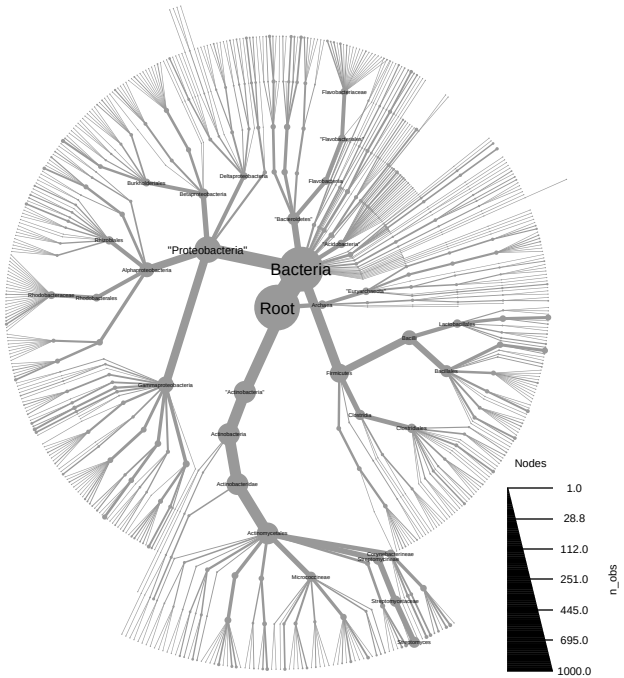
Labels can be added to nodes and edges.

Unlike most graphing in R, label sizes relative to other graph elements do not change with output size or aspect ratio.

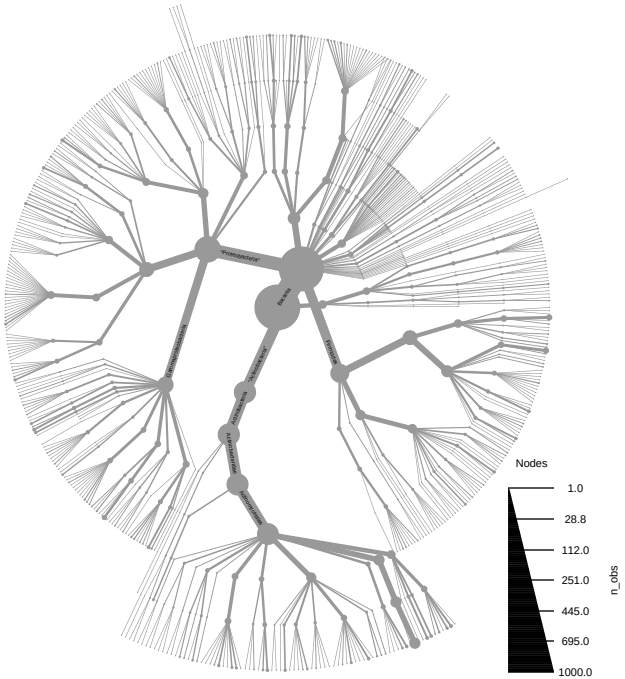
```
heat_tree(data, node_size = n_obs, node_label = taxon_names,
          node_label_max = 200, title = "Node labels")

heat_tree(data, node_size = n_obs, edge_label = taxon_names,
          edge_label_max = 200, title = "Edge labels")
```

Node labels



Edge labels



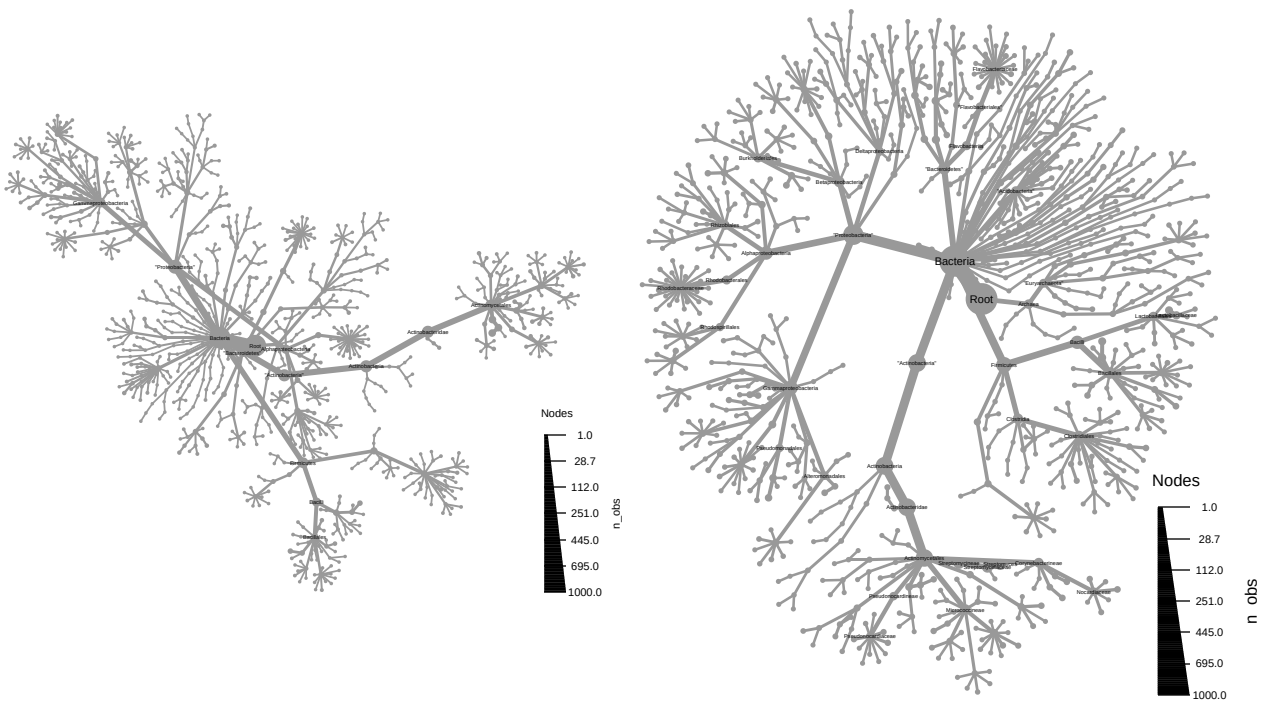
Plotting taxonomic data: Layouts

Different layouts are available that suit different data structures.

For stochastic, simulated layouts like “davidson-harel”, it can sometimes be useful to initialize with another layout.

```
set.seed(2)
heat_tree(data, node_size = n_obs, node_label = taxon_names,
          layout = "davidson-harel")

heat_tree(data, node_size = n_obs, node_label = taxon_names,
          layout = "davidson-harel", initial_layout = "reingold")
```



Subsetting taxonomic data

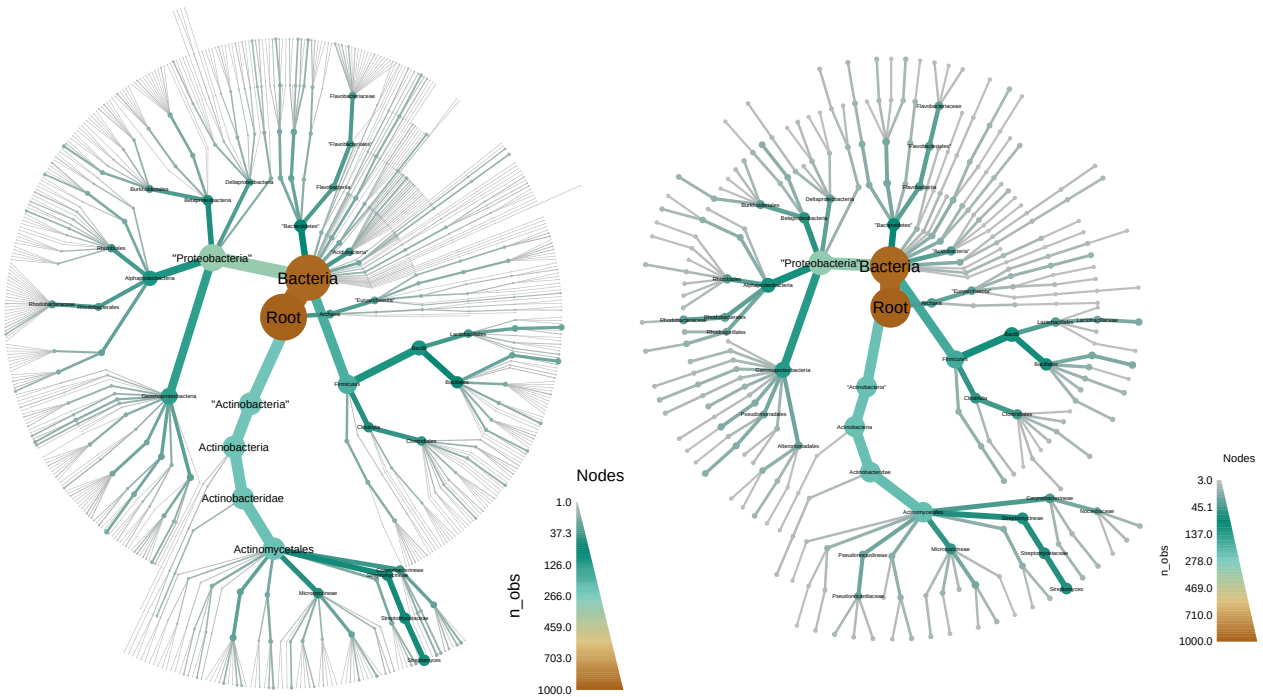
Modifying data is done using functions analogous to `dplyr`'s functions for data frames.

Filtering taxa can be done using `filter_taxa` with one or more filtering conditions.

The code below filters out all taxa with less than 3 observations.

```
data %>%
  heat_tree(node_size = n_obs, node_label = taxon_names, node_color = n_obs)

filter_taxa(data, n_obs >= 3) %>%
  heat_tree(node_size = n_obs, node_label = taxon_names, node_color = n_obs)
```



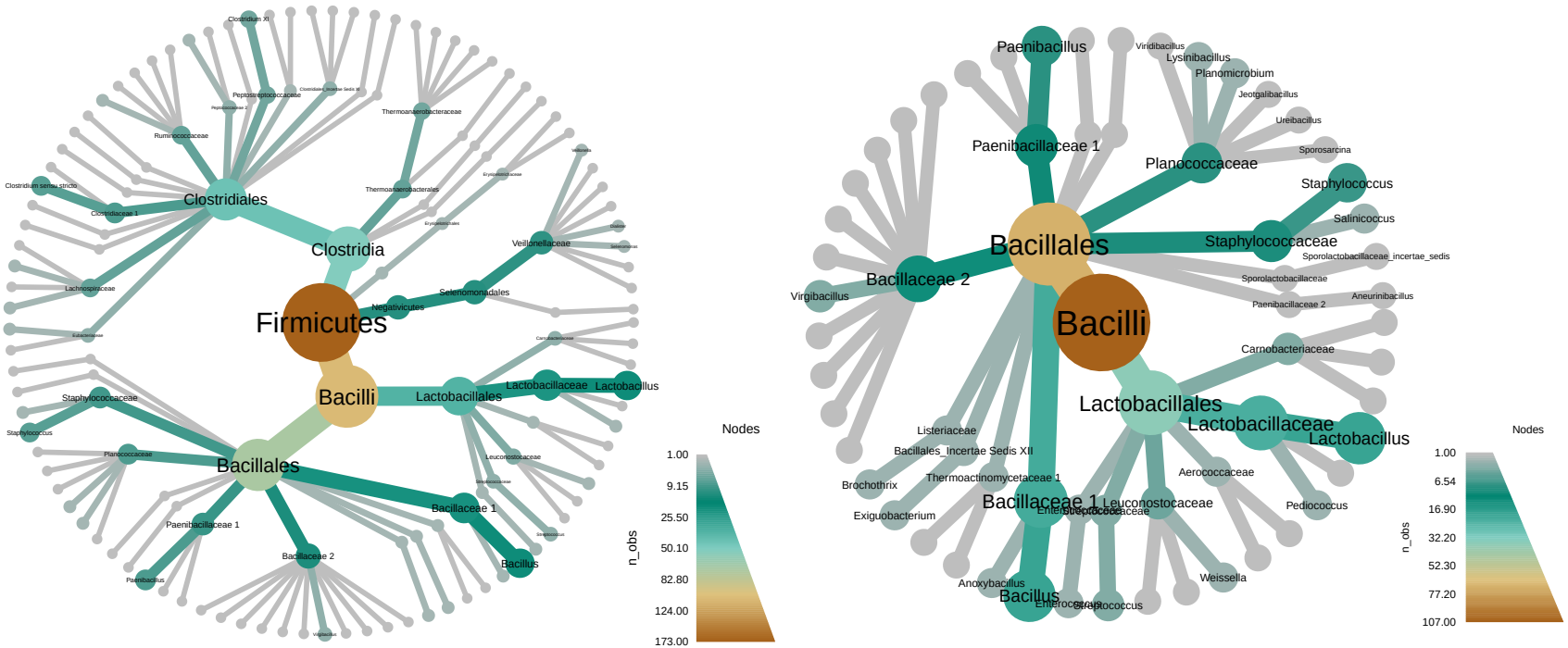
Subsetting taxonomic data: Including subtaxa

The subtaxa of taxa passing the filter can be included as well using the `subtaxa` option.

The code below subsets the data for "Firmicutes" and "Bacilli" and their subtaxa.

```
filter_taxa(data, taxon_names == "Firmicutes", subtaxa = TRUE) %>%
  heat_tree(node_size = n_obs, node_label = taxon_names, node_color = n_obs)

filter_taxa(data, taxon_names == "Bacilli", subtaxa = TRUE) %>%
  heat_tree(node_size = n_obs, node_label = taxon_names, node_color = n_obs)
```



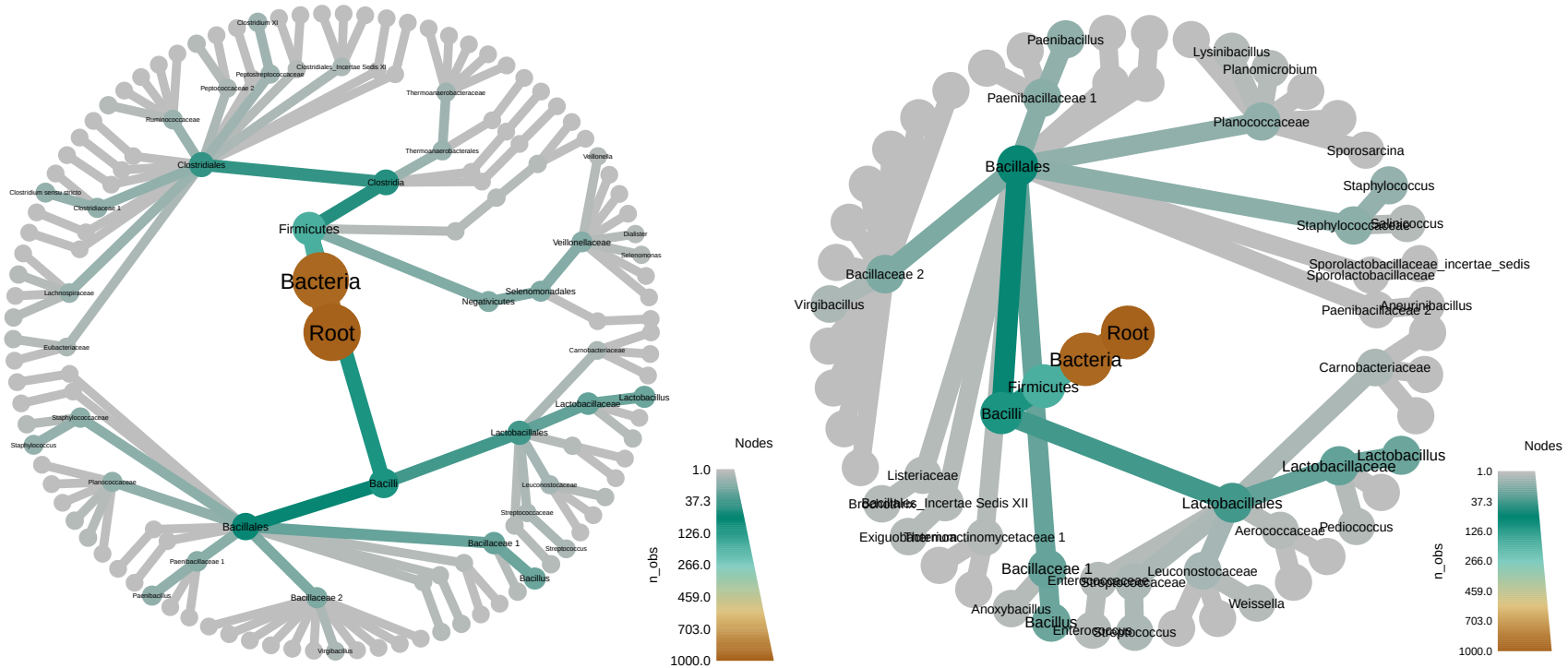
Subsetting taxonomic data: Including supertaxa

The `supertaxa` option works similar to the `subtaxa` option.

The code below does the same subset as last slide, but includes the supertaxa as well.

```
filter_taxa(data, taxon_names == "Firmicutes", subtaxa = TRUE, supertaxa = TRUE) %>%
  heat_tree(node_size = n_obs, node_label = taxon_names, node_color = n_obs)
```

```
filter_taxa(data, taxon_names == "Bacilli", subtaxa = TRUE, supertaxa = TRUE) %>%
  heat_tree(node_size = n_obs, node_label = taxon_names, node_color = n_obs)
```



Subsetting taxonomic data: Removing internal taxa

By default, when internal taxa are removed their subtaxa are reassigned to any supertaxon that passed the filter.

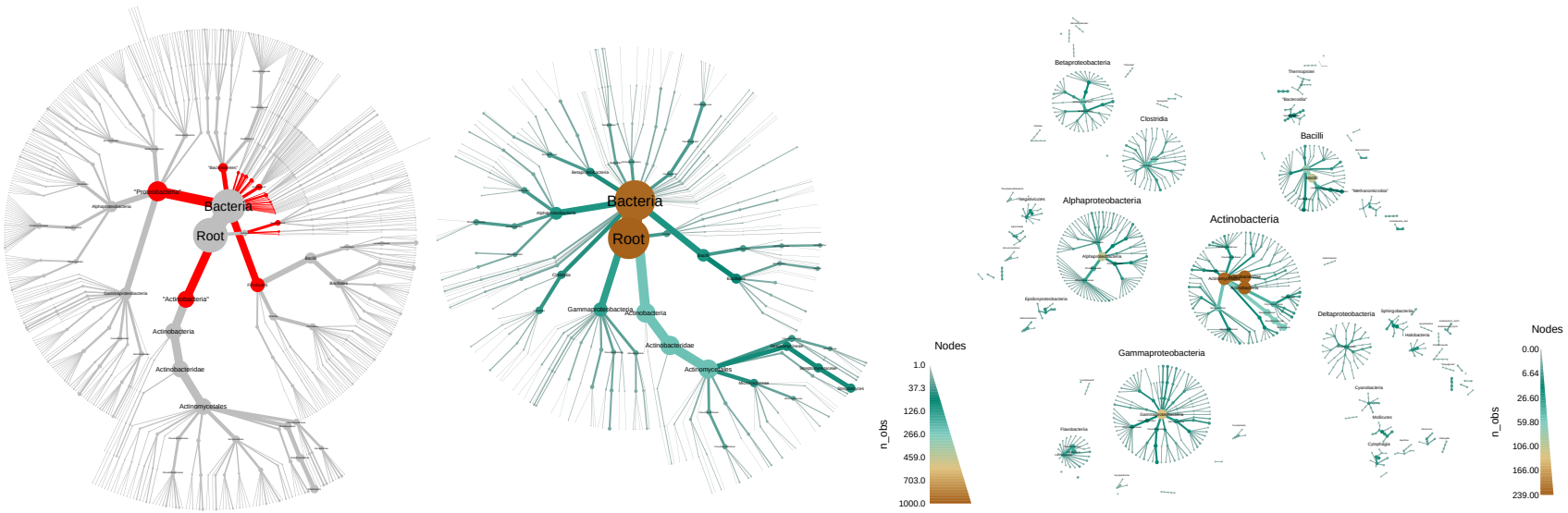
Not doing this is possible with the `reassign_taxa` option, but usually makes a mess.

The code below removes the taxa in red with and without reassigning taxa.

```
data %>%
  heat_tree(node_size = n_obs, node_label = taxon_names,
            node_color = ifelse(n_supertaxa == 2, "red", "grey"), make_legend = FALSE)

filter_taxa(data, n_supertaxa != 2) %>%
  heat_tree(node_size = n_obs, node_label = taxon_names, node_color = n_obs)

filter_taxa(data, n_supertaxa != 2, reassign_taxa = FALSE) %>%
  heat_tree(node_size = n_obs, node_label = taxon_names, node_color = n_obs,
            tree_label = taxon_names)
```



Sampling taxa

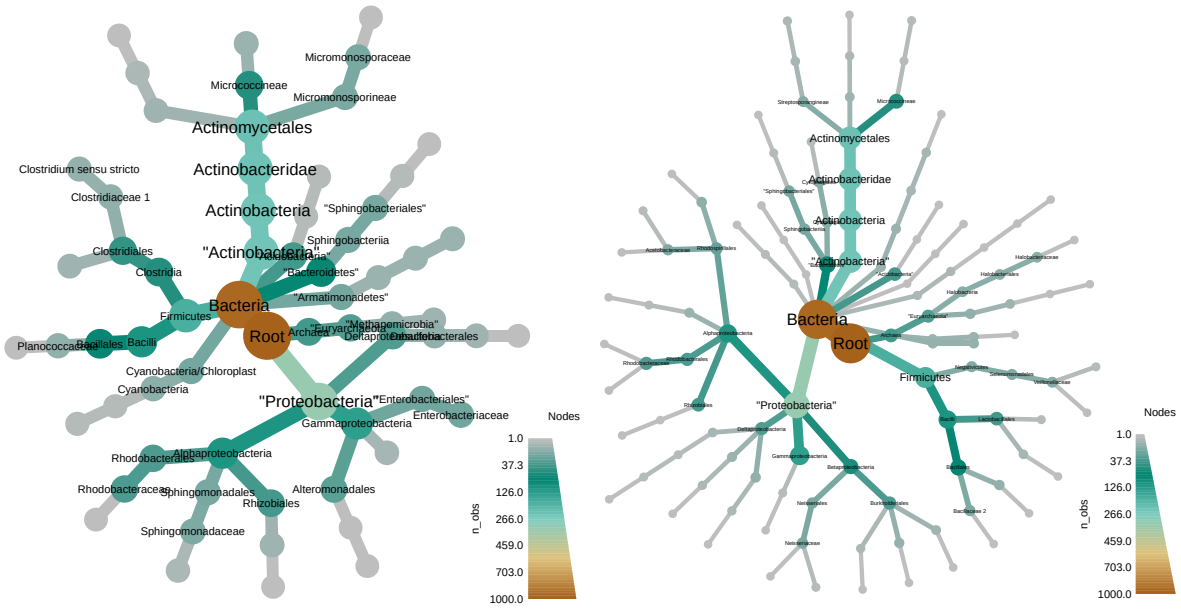
Random sampling of taxa works in a similar way to filtering taxa.

All of the options for `filter_taxa`, such as `subtaxa` and `reassign_taxa`, are available to `sample_n_taxa`.

```
set.seed(1)

sample_n_taxa(data, 20, supertaxa = TRUE) %>%
  heat_tree(node_size = n_obs, node_label = taxon_names, node_color = n_obs)

sample_n_taxa(data, 40, supertaxa = TRUE) %>%
  heat_tree(node_size = n_obs, node_label = taxon_names, node_color = n_obs)
```



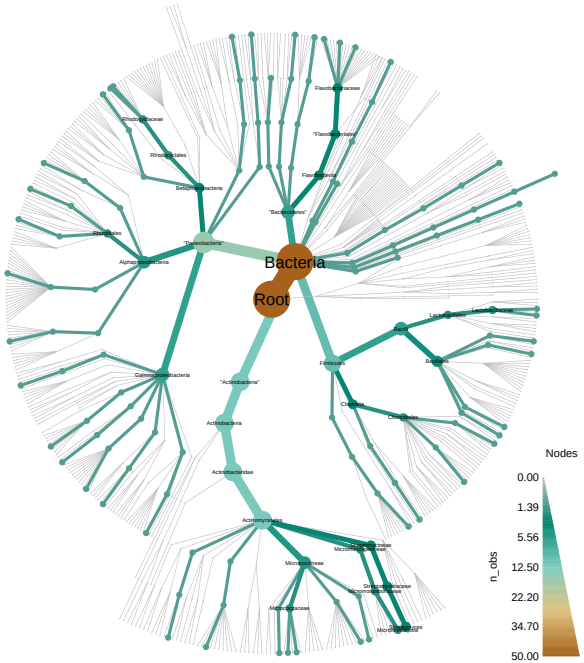
Sampling observations

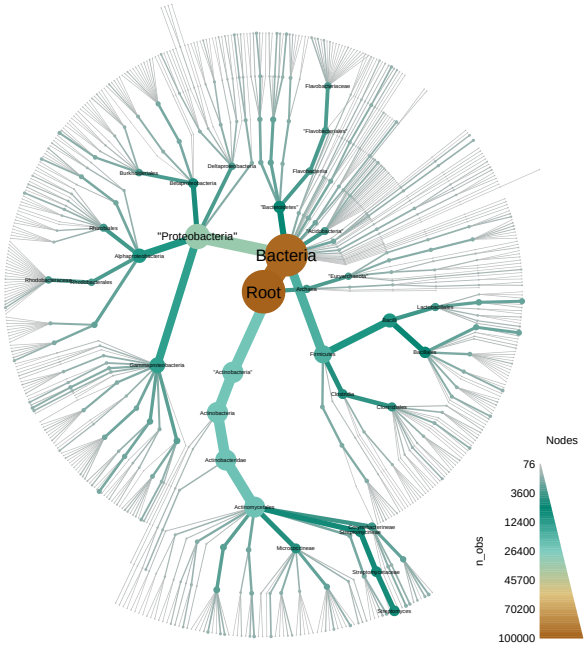
Observations assigned to taxa can also be filtered and sampled.

The code below randomly selects 50 (without replacement) and 100,000 (with replacement) observations.

```
sample_n_obs(data, "tax_data", 50) %>%
  heat_tree(node_size = n_obs, node_label = taxon_names, node_color = n_obs)

sample_n_obs(data, "tax_data", 100000, replace = TRUE) %>%
  heat_tree(node_size = n_obs, node_label = taxon_names, node_color = n_obs)
```





Example Application: The Human Microbiome project

Pairwise comparisons of microbiome composition in different parts of the human body. Only significant differences in abundance are colored.