Lesson 12

Vicki Hertzberg

4/2/2017

Beyond the OTU Table

`phyloseq` is an R package for efficient interactive and reproducible analyses of OTU-clustered high-throughput phylogenetic sequencing data. We can take the sequence variant table produced by `dada2` and use it, along with taxonomic classification and data about the samples, in these analyses. `Phyloseq` imports, stores, analyzes, and displays these data.

In our last lesson we showed how to hand off data to `phyloseq` from `dada2` , using the `phyloseq` command to create a `phyloseq` object from an OTU table, a taxonomic table, and a sample dataset.

In this lesson we will some of the other functionality of `phyloseq` :

- explore some of the example datasets that are included;
- creating `phyloseq` objects from the output of other programs;
- graphics in `phyloseq'.

Load Packages and Example Data

To get started we will need to load both `phyloseq` and `ggplot2` .

```
# Load packages

library(phyloseq)
packageVersion("phyloseq")
```

```
## [1] '1.19.1'
```

```
library(ggplot2)
packageVersion("ggplot2")
```

```
## [1] '2.2.1'
```

Load example data using the `data()` command.

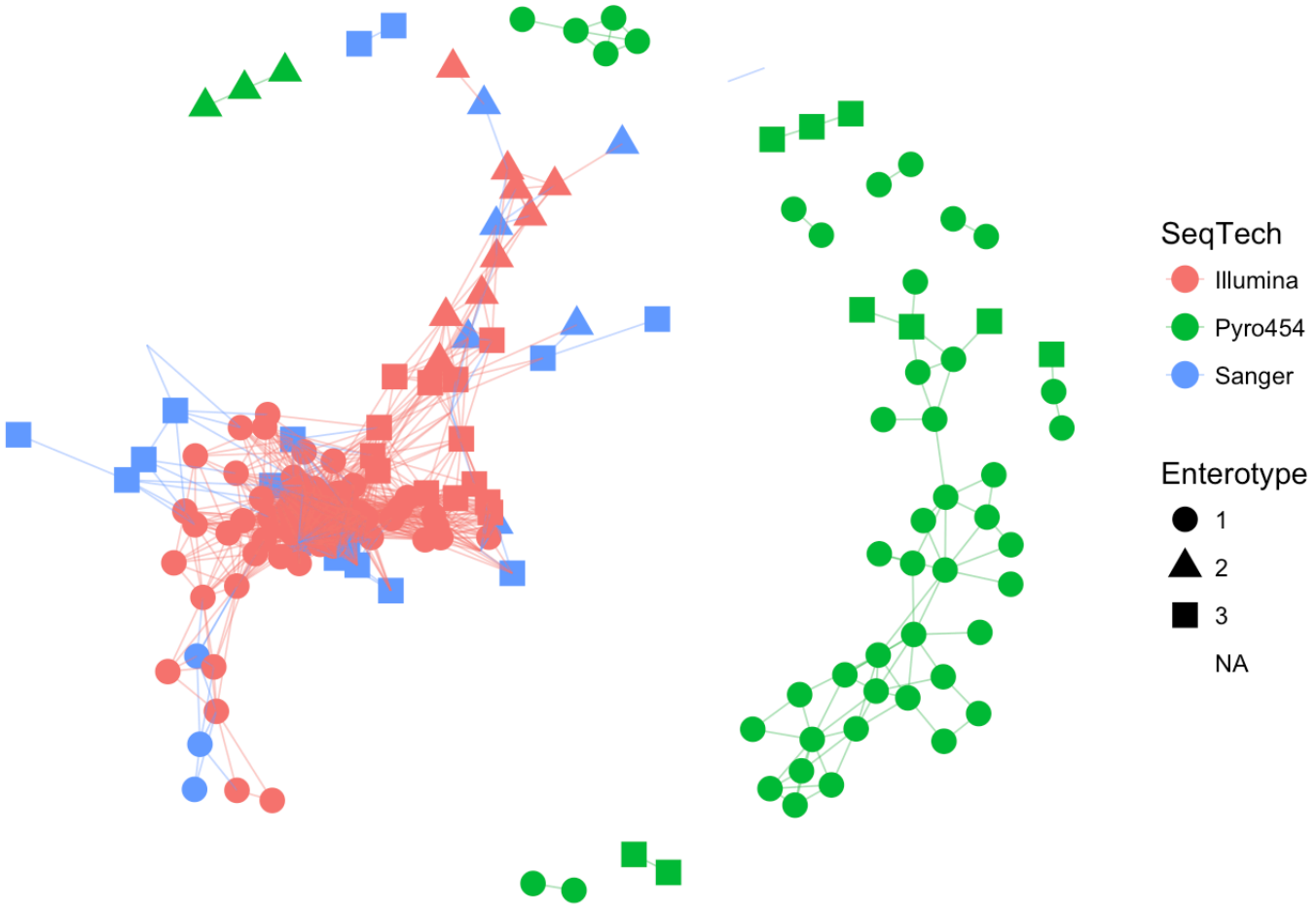
```
# Load example data

data(GlobalPatterns)
data(esophagus)
data(enterotype)
data(soilrep)
```

You can try these examples using the `example()` command.

```
# Use the example() command on the enterotype database

example(enterotype, ask=FALSE)
```

```
##
## entrty> data(enterotype)
##
## entrty> ig <- make_network(enterotype, "samples", max.dist=0.3)
##
## entrty> plot_network(ig, enterotype, color="SeqTech", shape="Enterotype", line_wei
ght=0.3, label=NULL)
```

```
## Warning: attributes are not identical across measure variables; they will
## be dropped
```



You can also import data from other packages such as MG-RAST or QIIME. The newer versions of QIIME produce files formatted according to the *biom* standard (see http://biom-format.org (http://biom-format.org)).

The phyloseq package has small examples of biom files with different levels and organization of data. The following chunk illustrates how to import each of the 4 main types of biom files. Note that in practice you won't know what type your file is, so best to include all 4. In addition, the `import_biom` funciton allows you to import an associated phylogenetic tree file and reference sequence file (e.g., fasta file).

To do this you must first define the file paths. For this example, this should be within the phyloseq package. Thus we use the `system.file` command to get the paths.

```
# Load the files
rich_dense_biom  <- system.file("extdata", "rich_dense_otu_table.biom",  package="phy
loseq")
rich_sparse_biom <- system.file("extdata", "rich_sparse_otu_table.biom", package="phy
loseq")
min_dense_biom   <- system.file("extdata", "min_dense_otu_table.biom",   package="phy
loseq")
min_sparse_biom  <- system.file("extdata", "min_sparse_otu_table.biom",  package="phy
loseq")
treefilename <- system.file("extdata", "biom-tree.phy",  package="phyloseq")
refseqfilename <- system.file("extdata", "biom-refseq.fasta",  package="phyloseq")
```

Next we use these paths as argments to the `import_biom` function. The tree and reference sequence files are both suitable for any of the biom file types, so we only need one path for each.

```
# Import via the paths

import_biom(rich_dense_biom, treefilename, refseqfilename, parseFunction = parse_taxo
nomy_greengenes)
```

```
## phyloseq-class experiment-level object
## otu_table()   OTU Table:         [ 5 taxa and 6 samples ]
## sample_data() Sample Data:       [ 6 samples by 4 sample variables ]
## tax_table()   Taxonomy Table:    [ 5 taxa by 7 taxonomic ranks ]
## phy_tree()    Phylogenetic Tree: [ 5 tips and 4 internal nodes ]
## refseq()      DNAStringSet:      [ 5 reference sequences ]
```

```
import_biom(rich_sparse_biom, treefilename, refseqfilename, parseFunction = parse_tax
onomy_greengenes)
```

```
## phyloseq-class experiment-level object
## otu_table()   OTU Table:         [ 5 taxa and 6 samples ]
## sample_data() Sample Data:       [ 6 samples by 4 sample variables ]
## tax_table()   Taxonomy Table:    [ 5 taxa by 7 taxonomic ranks ]
## phy_tree()    Phylogenetic Tree: [ 5 tips and 4 internal nodes ]
## refseq()      DNAStringSet:      [ 5 reference sequences ]
```

```
import_biom(min_dense_biom, treefilename, refseqfilename, parseFunction = parse_taxon
omy_greengenes)
```

```
## phyloseq-class experiment-level object
## otu_table()   OTU Table:         [ 5 taxa and 6 samples ]
## phy_tree()    Phylogenetic Tree: [ 5 tips and 4 internal nodes ]
## refseq()      DNAStringSet:      [ 5 reference sequences ]
```

```
import_biom(min_sparse_biom, treefilename, refseqfilename, parseFunction = parse_taxo
nomy_greengenes)
```

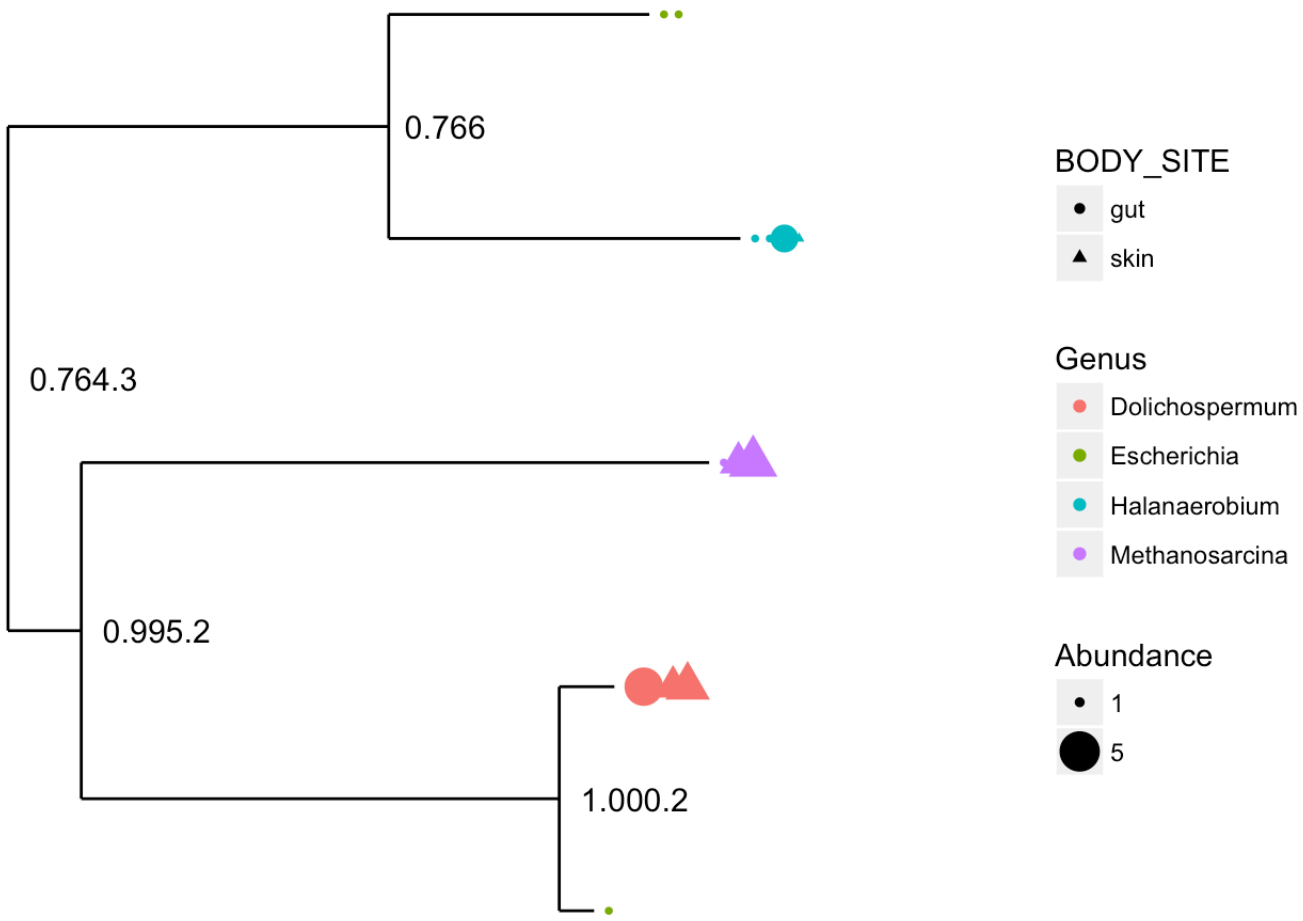
```
## phyloseq-class experiment-level object
## otu_table()   OTU Table:         [ 5 taxa and 6 samples ]
## phy_tree()    Phylogenetic Tree: [ 5 tips and 4 internal nodes ]
## refseq()      DNAStringSet:      [ 5 reference sequences ]
```

In practice you will store the result of your import as some variable name, like `myData` , then use this object in downstream analyses. For examples,

```
# Example of storing results of import and manipulating it

myData <- import_biom(rich_dense_biom, treefilename, refseqfilename, parseFunction =
parse_taxonomy_greengenes)
myData
```

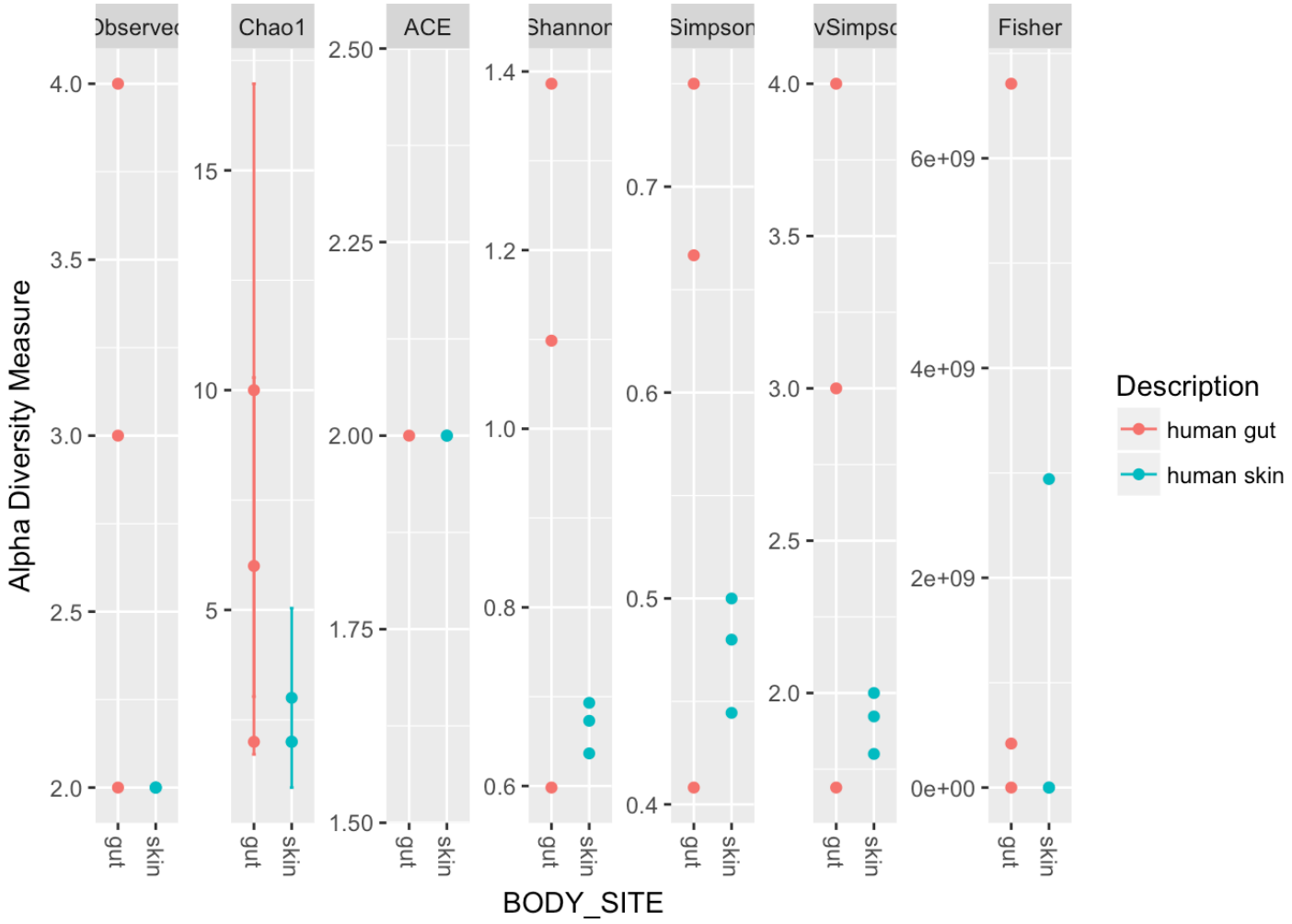
```
## phyloseq-class experiment-level object
## otu_table()   OTU Table:         [ 5 taxa and 6 samples ]
## sample_data() Sample Data:       [ 6 samples by 4 sample variables ]
## tax_table()   Taxonomy Table:    [ 5 taxa by 7 taxonomic ranks ]
## phy_tree()    Phylogenetic Tree: [ 5 tips and 4 internal nodes ]
## refseq()      DNAStringSet:      [ 5 reference sequences ]
```

```
plot_tree(myData, color = "Genus", shape = "BODY_SITE", size = "abundance")
```

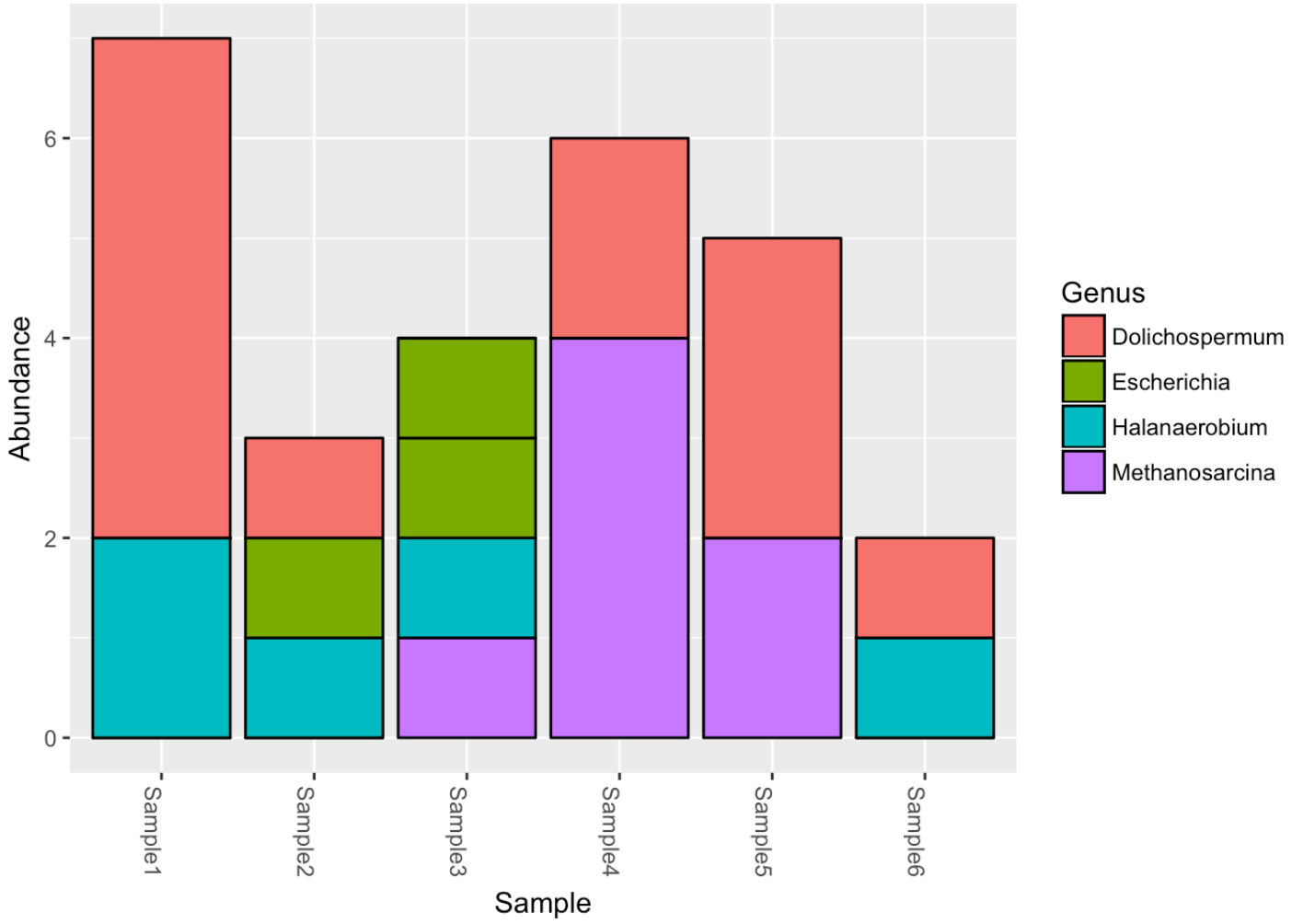



```
plot_richness(myData, x="BODY_SITE", color="Description")
```

```
## Warning: Removed 33 rows containing missing values (geom_errorbar).
```




```
plot_bar(myData, fill="Genus")
```




```
refseq(myData)
```

```
##   A DNAStringSet instance of length 5
##     width seq                                               names
## [1]   334 AACGTAGGTCACAAGCGTTGT...TTCCGTGCCGGAGTTAACAC GG_OTU_1
## [2]   465 TACGTAGGGAGCAAGCGTTAT...CCTTACCAGGGCTTGACATA GG_OTU_2
## [3]   249 TACGTAGGGGGCAAGCGTTAT...GGCTCGAAAGCGTGGGGAGC GG_OTU_3
## [4]   453 TACGTATGGTGCAAGCGTTAT...AAGCAACGCGAAGAACCTTA GG_OTU_4
## [5]   178 AACGTAGGGTGCAAGCGTTGT...GGAATGCGTAGATATCGGGA GG_OTU_5
```

The good people at `phyloseq` have also downloaded data from the Human Microbiome Project and made it available as an R dataset. So let's look at it

```
# Get the HMP data
# You will need to change your path

load("~/Documents/NRSG_741/Lesson12/HMPv35.RData")

HMPv35
```

```
## phyloseq-class experiment-level object
## otu_table()   OTU Table:         [ 45336 taxa and 4743 samples ]
## sample_data() Sample Data:       [ 4743 samples by 9 sample variables ]
## tax_table()   Taxonomy Table:    [ 45336 taxa by 6 taxonomic ranks ]
## phy_tree()    Phylogenetic Tree: [ 45336 tips and 45099 internal nodes ]
## refseq()      DNAStringSet:      [ 45336 reference sequences ]
```

Merge Data

`phyloseq` supports two types of merging.

1. You can merge OTUs or samples in a phyloseq object, based on a taxonomic or sample variable using the `merge_samples()` or `merge_taxa()` commands.
2. YOu can merge two or more data objects from the same experiment so that their data become part of the same phyloseq object using the `merge_phyloseq()` commend.

Merging Samples

Merging samples with `merge_samples()` can be a useful means of reducing noise or excess features, say by removing the individual effects between replicates or between samples for a particular explanatory variable. With `merge_samples()` the abundance values for the merged samples are summed.

As an example, let's first remove unobserved OTUs (those that sum 0 across all samples) and add an explanatory variable with which to organize later in plots, using the GlobalPatterns dataset.

```
# Remove empty samples

GP <- GlobalPatterns
GP <- prune_taxa(taxa_sums(GlobalPatterns) > 0, GlobalPatterns)
humantypes <- c("Feces", "Mock", "Skin", "Tongue")
sample_data(GP)$human <- get_variable(GP, "SampleType") %in% humantypes
```

Now on to merging samples:

```
# Merge Samples

mergedGP <- merge_samples(GP, "SampleType")
SD <- merge_samples(sample_data(GP), "SampleType")
print(SD[, "SampleType"])
```

```
##                     SampleType
## Feces                        1
## Freshwater                   2
## Freshwater (creek)           3
## Mock                         4
## Ocean                        5
## Sediment (estuary)           6
## Skin                         7
## Soil                         8
## Tongue                       9
```

```
print(mergedGP)
```

```
## phyloseq-class experiment-level object
## otu_table()   OTU Table:         [ 18988 taxa and 9 samples ]
## sample_data() Sample Data:       [ 9 samples by 8 sample variables ]
## tax_table()   Taxonomy Table:    [ 18988 taxa by 7 taxonomic ranks ]
## phy_tree()    Phylogenetic Tree: [ 18988 tips and 18987 internal nodes ]
```

```
sample_names(GP)
```

```
##  [1] "CL3"      "CC1"      "SV1"      "M31Fcsw"  "M11Fcsw"  "M31Plmr"
##  [7] "M11Plmr"  "F21Plmr"  "M31Tong"  "M11Tong"  "LMEpi24M" "SLEpi20M"
## [13] "AQC1cm"   "AQC4cm"   "AQC7cm"   "NP2"      "NP3"      "NP5"
## [19] "TRRsed1"  "TRRsed2"  "TRRsed3"  "TS28"     "TS29"     "Even1"
## [25] "Even2"    "Even3"
```

```
sample_names(mergedGP)
```

```
## [1] "Feces"             "Freshwater"        "Freshwater (creek)"
## [4] "Mock"              "Ocean"             "Sediment (estuary)"
## [7] "Skin"              "Soil"              "Tongue"
```

```
identical(SD, sample_data(mergedGP))
```

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

The OTU abundances of merged samples are summed. Let's investigate this ourselves looking at just the top 10 most abundant OTUs.

```
# Look at top 10 most abundant OTUs

OTUnames10 <- names(sort(taxa_sums(GP), TRUE)[1:10])
GP10 <- prune_taxa(OTUnames10, GP)
mGP10 <- prune_taxa(OTUnames10,mergedGP)
ocean_samples <- sample_names(subset(sample_data(GP), SampleType == "Ocean"))
print(ocean_samples)
```

```
## [1] "NP2" "NP3" "NP5"
```

```
otu_table(GP10)[, ocean_samples]
```

```
## OTU Table:          [10 taxa and 3 samples]
##                      taxa are rows
##         NP2   NP3   NP5
## 329744   91   126   120
## 317182 3148 12370 63084
## 549656 5045 10713  1784
## 279599  113   114   126
## 360229   16    83   786
## 94166    49   128   709
## 550960   11    86    65
## 158660   13    39    28
## 331820   24   101   105
## 189047    4    33    29
```

```
rowSums(otu_table(GP10)[, ocean_samples])
```

```
## 329744 317182 549656 279599 360229  94166 550960 158660 331820 189047
##    337  78602  17542    353    885    886    162     80    230     66
```

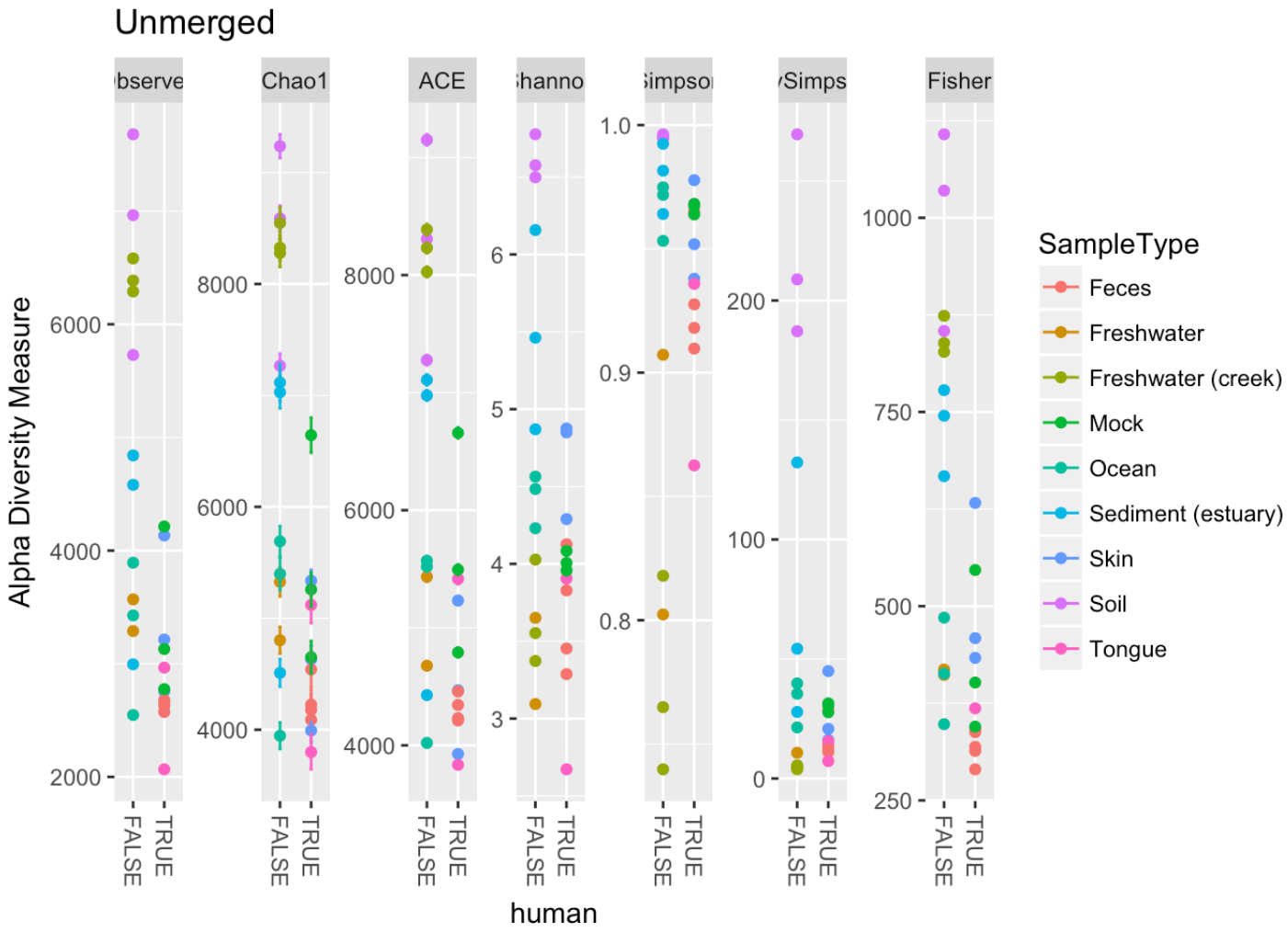
```
otu_table(mGP10)["Ocean", ]
```

```
## OTU Table:          [10 taxa and 1 samples]
##                      taxa are columns
##       329744 317182 549656 279599 360229 94166 550960 158660 331820 189047
## Ocean    337  78602  17542    353    885   886    162     80    230     66
```

Now let's look at the merge graphically between two richnes estimate summary plots.

```
# Richness estimate plots

plot_richness(GP, "human", "SampleType", title="Unmerged")
```

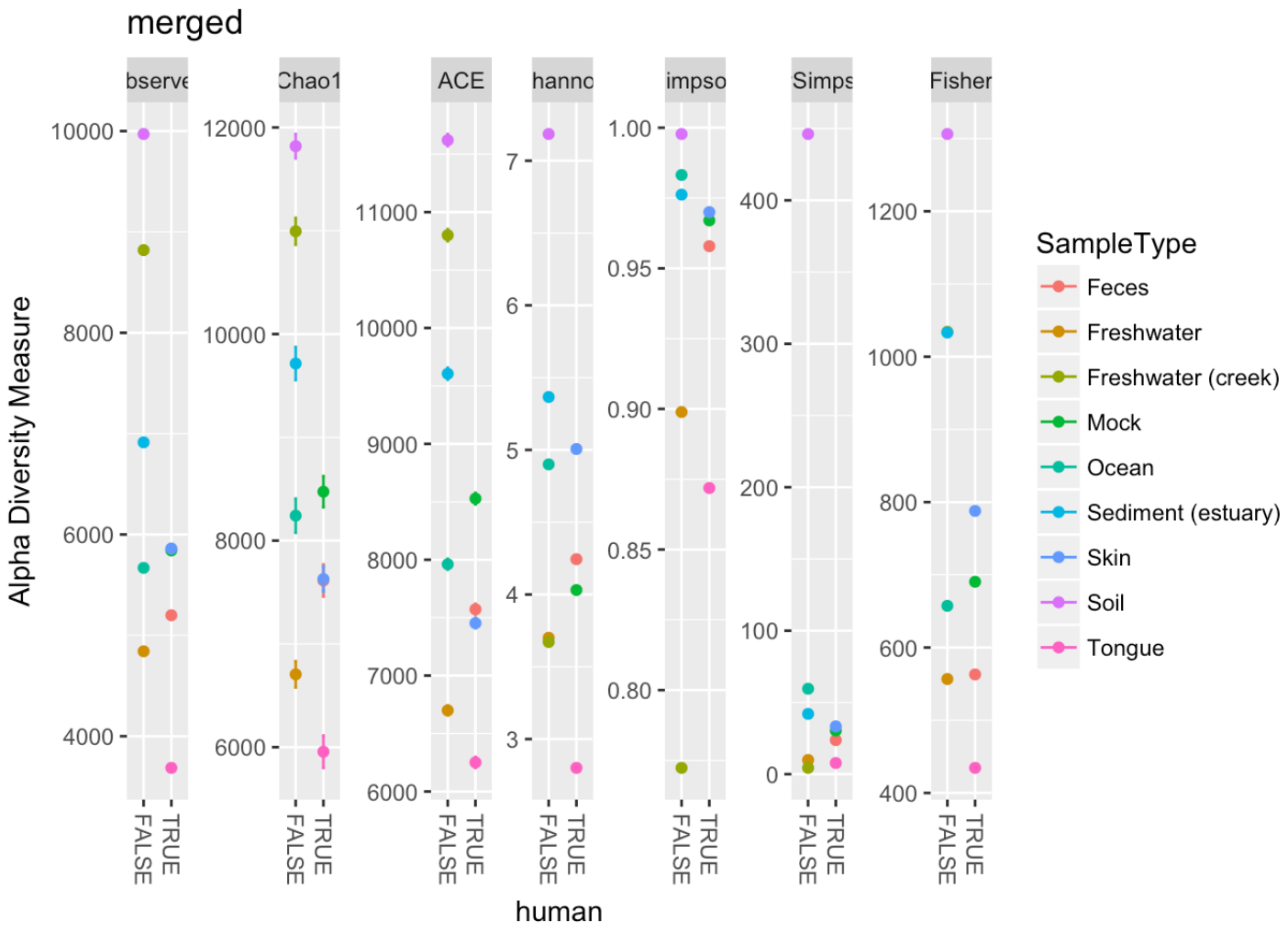
```
## Warning: Removed 130 rows containing missing values (geom_errorbar).
```



The merge can do some weird things to samples variables, so let's re-add these variables to the `sample_data` before we plot.

```
# Re-add sample data and plot richness plots again.

sample_data(mergedGP)$SampleType <- sample_names(mergedGP)
sample_data(mergedGP)$human <- sample_names(mergedGP) %in% humantypes
plot_richness(mergedGP, "human", "SampleType", title="merged")
```

```
## Warning: Removed 45 rows containing missing values (geom_errorbar).
```

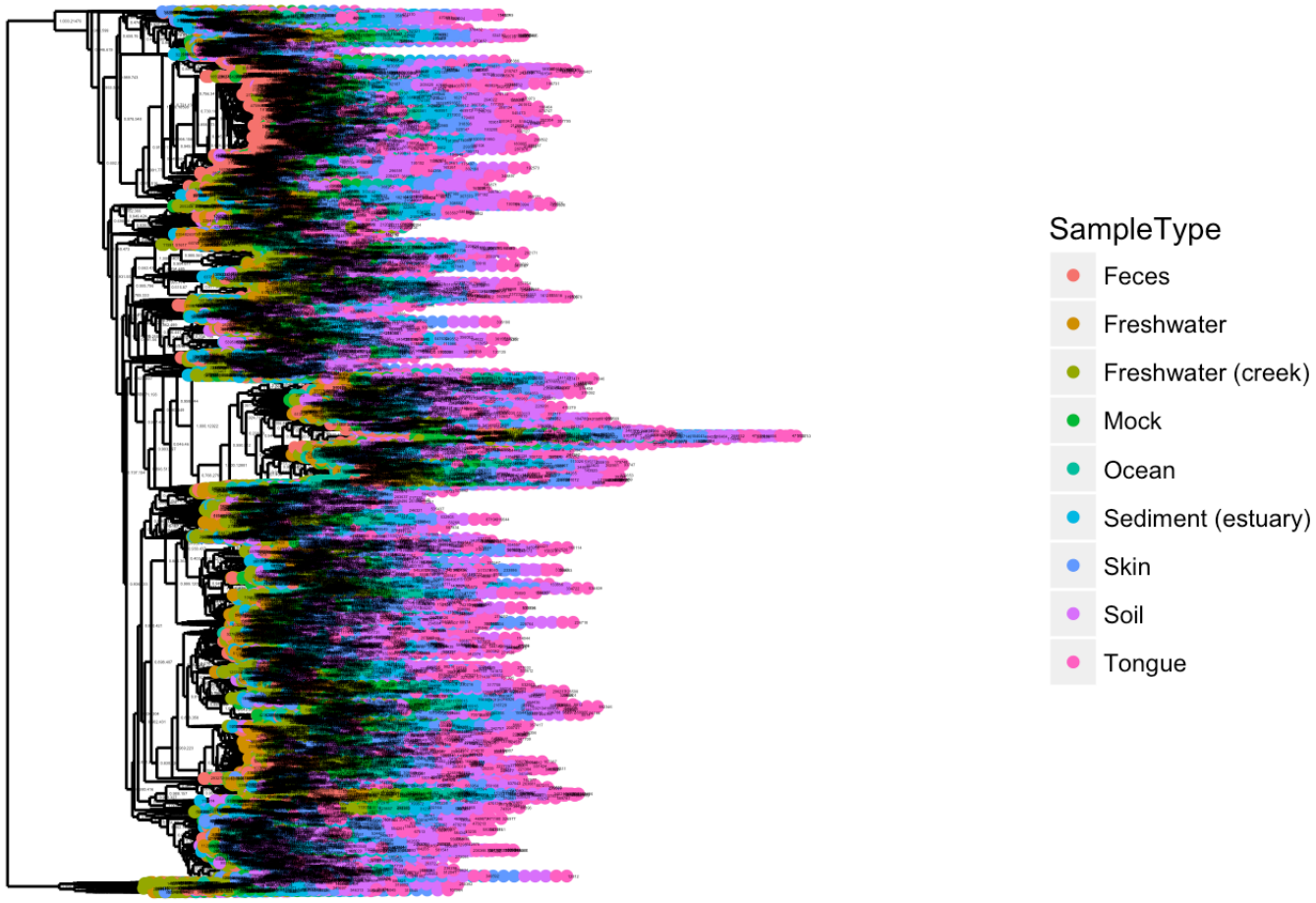


When we combine the abundances of non-replicate samples from the same environment, the estimates of absolute richness increase for each environment. But, more to the point, the new plot is easier to read and interpret, one of the reasons one might use the `merge_samples` function.

Merging Taxa

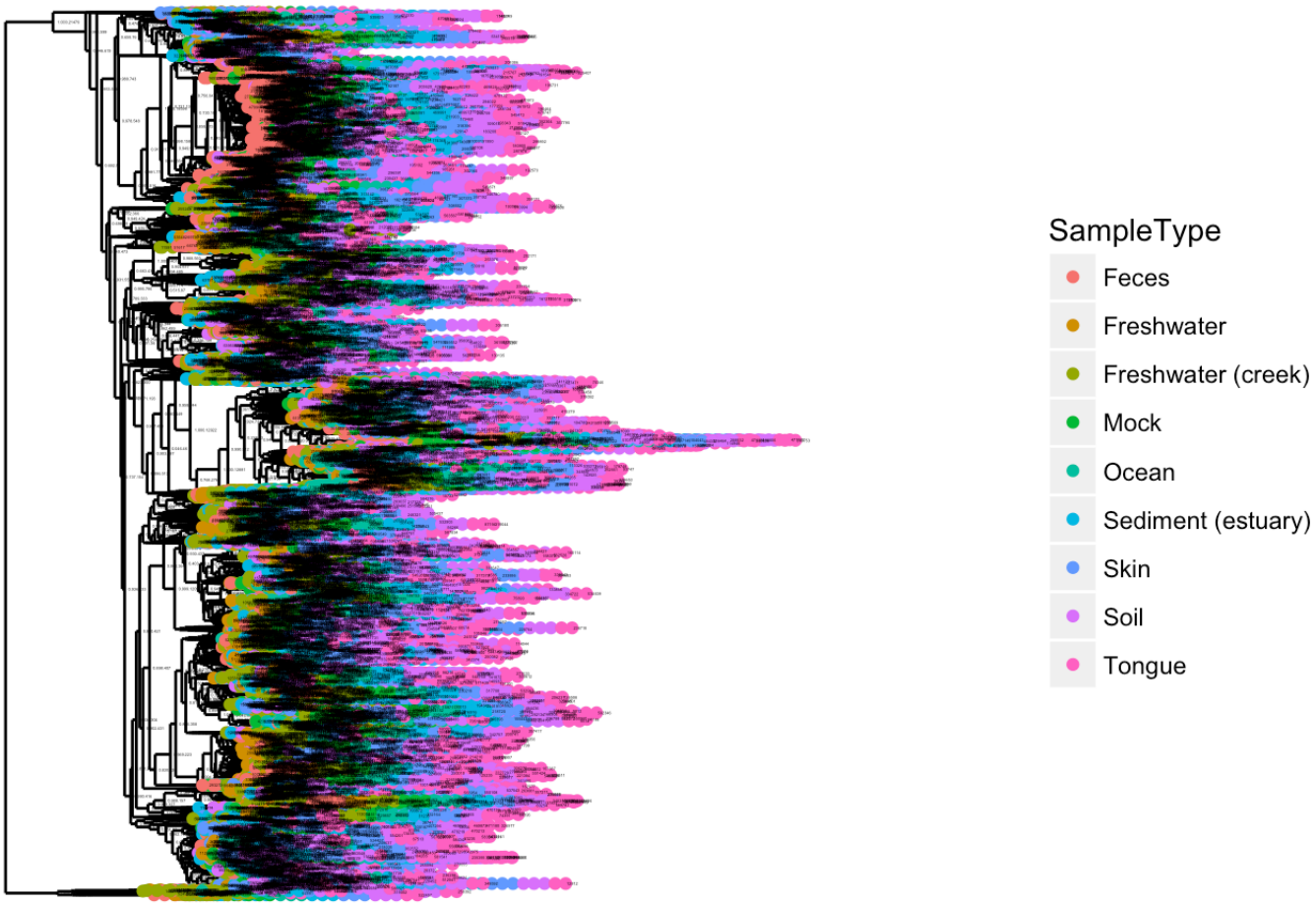
One of the issues with microbial census data is that a fine-scaled definition for OTU may blur a pattern that might otherwise be evident if we considered a higher taxonomic rank. The best way to deal with this is to agglomerate phylogenetic tree tips or taxa by using the agglomeration functions `tip_glom` or `tax_glom`. Let's use the object we imported earlier from the biom format files, myData. First the original tree:

```
# Plot phylogenetic tree of GlobalPatterns object
plot_tree(GP, color="SampleType", sizebase=2, label.tips="taxa_name")
```



What a mess. Let's try using `merge_taxa` to merge the first 100 OTUs into one new OTU. The option 2 in the call means to combine the counts of these 100 OTUs into the index for the new OTU.

```
# Combine the tree tips and plot again

x1 <- merge_taxa(GP, taxa_names(GP)[1:100], 2)
plot_tree(x1, color="SampleType", sizebase=2, label.tips="taxa_name")
```



You can barely see the difference, but it is there.

Merging Objects

You can also merge phyloseq objects. We will demonstrate with a somewhat trivial example by extracting the components of the GlobalPatterns object and then building them back to the original form using the command `merge_phyloseq`.

```
# Split apart GlobalPatterns

data(GlobalPatterns)
tree <- phy_tree(GlobalPatterns)
tax <- tax_table(GlobalPatterns)
otu <- otu_table(GlobalPatterns)
sam <- sample_data(GlobalPatterns)

# Create new phyloseq object

otutax <- phyloseq(otu, tax)
otutax
```

```
## phyloseq-class experiment-level object
## otu_table()   OTU Table:         [ 19216 taxa and 26 samples ]
## tax_table()   Taxonomy Table:    [ 19216 taxa by 7 taxonomic ranks ]
```

You can se that our new `otutax` object has just the OTU table and the taxonomy table. Let's use `merge_phyloseq` to build up the original `GlobalPatterns` object, and compare to make sure they are identical.

```
# Build back up

GP2 = merge_phyloseq(otutax, sam, tree)

# Test for identity

identical(GP2, GlobalPatterns)
```

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

Accessing and (Pre)Processing Data

Accessors

We have already seen a bit above about how to access some of the data within a `phyloseq` object. We will comprehensively cover it now.

Let's use the `GlobalPatterns` dataset.

```
# Load data object and see what is there

data("GlobalPatterns")
GlobalPatterns
```

```
## phyloseq-class experiment-level object
## otu_table()   OTU Table:         [ 19216 taxa and 26 samples ]
## sample_data() Sample Data:       [ 26 samples by 7 sample variables ]
## tax_table()   Taxonomy Table:    [ 19216 taxa by 7 taxonomic ranks ]
## phy_tree()    Phylogenetic Tree: [ 19216 tips and 19215 internal nodes ]
```

```
ntaxa(GlobalPatterns)
```

```
## [1] 19216
```

```
nsamples(GlobalPatterns)
```

```
## [1] 26
```

```
sample_names(GlobalPatterns)[1:5]
```

```
## [1] "CL3"     "CC1"     "SV1"     "M31Fcsw" "M11Fcsw"
```

```
rank_names(GlobalPatterns)
```

```
## [1] "Kingdom" "Phylum"  "Class"   "Order"   "Family"  "Genus"   "Species"
```

```
sample_variables(GlobalPatterns)
```

```
## [1] "X.SampleID"               "Primer"
## [3] "Final_Barcode"            "Barcode_truncated_plus_T"
## [5] "Barcode_full_length"      "SampleType"
## [7] "Description"
```

```
otu_table(GlobalPatterns)[1:5, 1:5]
```

```
## OTU Table:          [5 taxa and 5 samples]
##                      taxa are rows
##        CL3 CC1 SV1 M31Fcsw M11Fcsw
## 549322   0   0   0       0       0
## 522457   0   0   0       0       0
## 951      0   0   0       0       0
## 244423   0   0   0       0       0
## 586076   0   0   0       0       0
```

```
tax_table(GlobalPatterns)[1:5, 1:4]
```

```
## Taxonomy Table:     [5 taxa by 4 taxonomic ranks]:
##        Kingdom   Phylum          Class          Order
## 549322 "Archaea" "Crenarchaeota" "Thermoprotei" NA
## 522457 "Archaea" "Crenarchaeota" "Thermoprotei" NA
## 951    "Archaea" "Crenarchaeota" "Thermoprotei" "Sulfolobales"
## 244423 "Archaea" "Crenarchaeota" "Sd-NA"        NA
## 586076 "Archaea" "Crenarchaeota" "Sd-NA"        NA
```

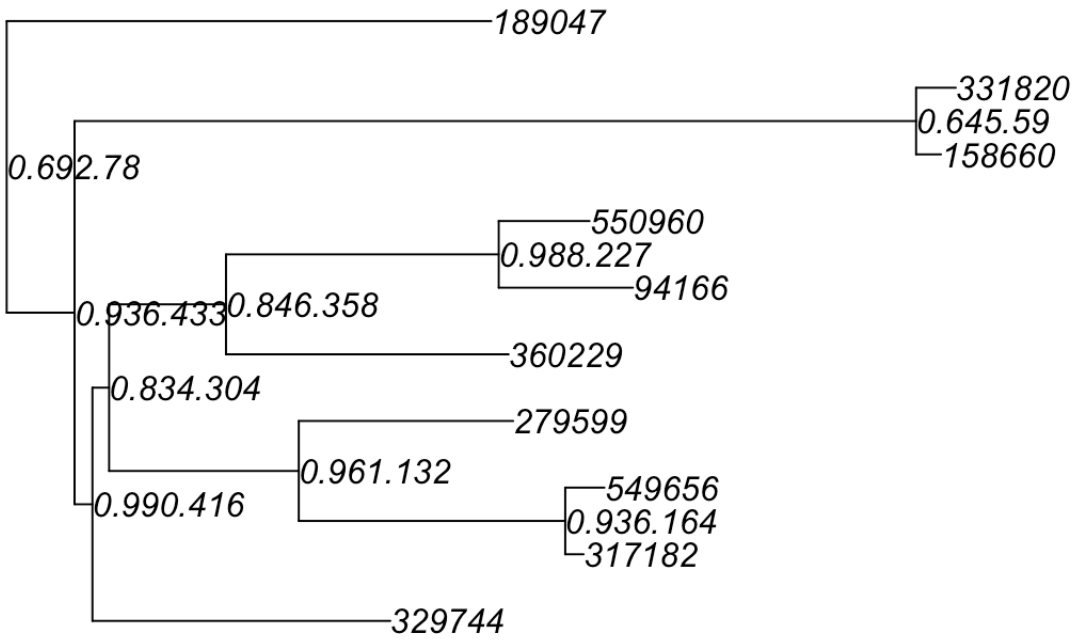
```
phy_tree(GlobalPatterns)
```

```
##
## Phylogenetic tree with 19216 tips and 19215 internal nodes.
##
## Tip labels:
##  549322, 522457, 951, 244423, 586076, 246140, ...
## Node labels:
##  , 0.858.4, 1.000.154, 0.764.3, 0.995.2, 1.000.2, ...
##
## Rooted; includes branch lengths.
```

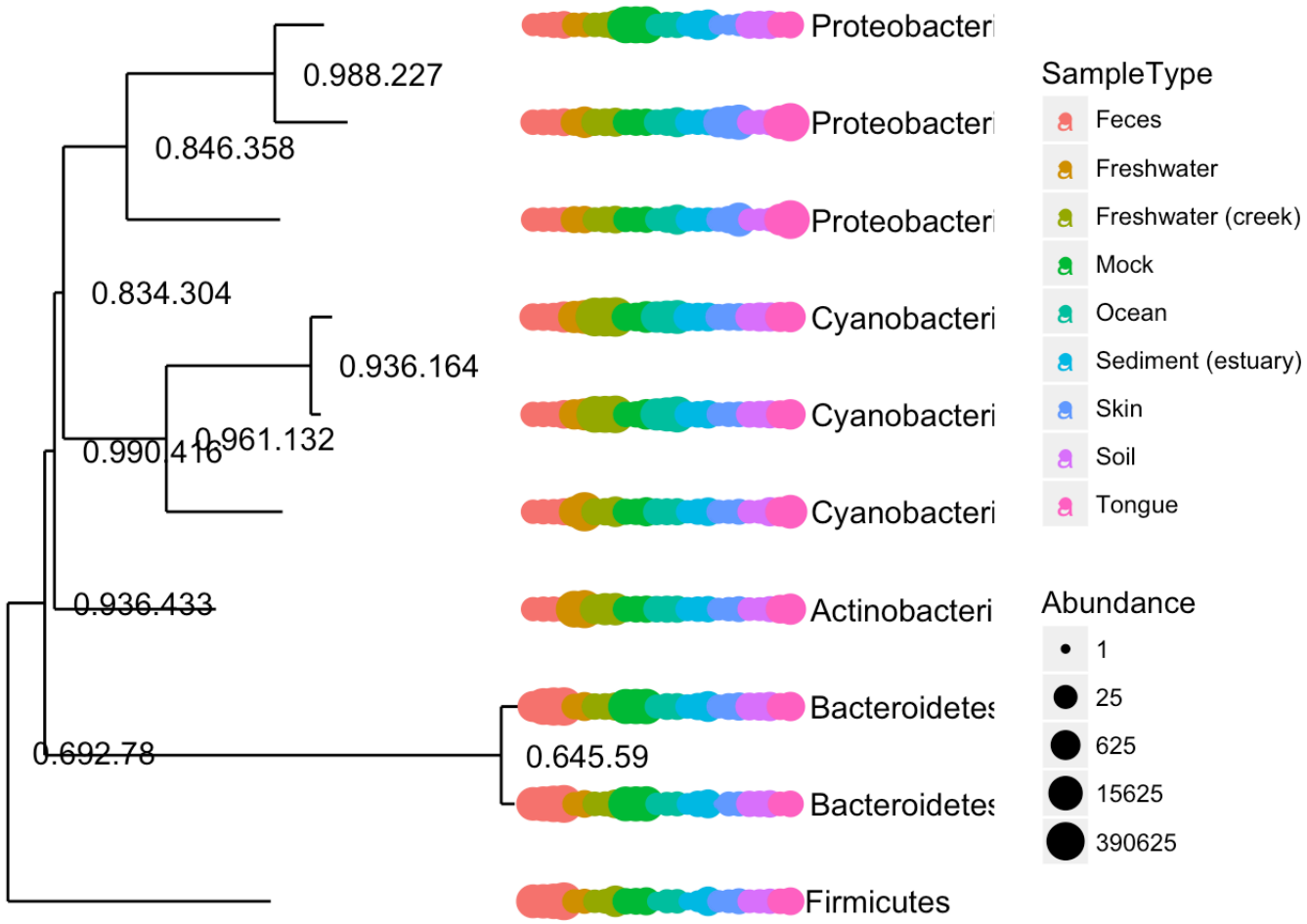
```
taxa_names(GlobalPatterns)[1:10]
```

```
##  [1] "549322" "522457" "951"    "244423" "586076" "246140" "143239"
##  [8] "244960" "255340" "144887"
```

```
myTaxa <- names(sort(taxa_sums(GlobalPatterns), decreasing = TRUE)[1:10])
ex1 <- prune_taxa(myTaxa, GlobalPatterns)
plot(phy_tree(ex1), show.node.label = TRUE)
```




```
plot_tree(ex1, color = "SampleType", label.tips = "Phylum", ladderize = "left", justi
fy = "left", size = "Abundance")
```



Preprocessing

`phyloseq` also includes functions for filtering, subsetting, and merging abundance data. Let's filter!

Filtering

Filtering is designed in a modular fashion. The functions `prune_taxa` and `prune_samples` will directly remove unwanted indices. The functions `filterfun_sample` and `genefilter_sample` will allow you to build arbitrarily complex sample-wise filtering criteria. The function `filter_taxa` allows for taxa-wise filtering.

In the following example we will use the GlobalPatterns object again, first transforming it to relative abundance in a new `GPr` object, then filtering that object to remove all OTUs that have a mean abundance $< 10^-5$, creating another new object.

```
# Create the object with the relative abundance data

GPr <- transform_sample_counts(GlobalPatterns, function(x) x / sum(x))
GPfr <- filter_taxa(GPr, function(x) mean(x) > 1e-5, TRUE)

GlobalPatterns
```

```
## phyloseq-class experiment-level object
## otu_table()   OTU Table:         [ 19216 taxa and 26 samples ]
## sample_data() Sample Data:       [ 26 samples by 7 sample variables ]
## tax_table()   Taxonomy Table:    [ 19216 taxa by 7 taxonomic ranks ]
## phy_tree()    Phylogenetic Tree: [ 19216 tips and 19215 internal nodes ]
```

```
GPr
```

```
## phyloseq-class experiment-level object
## otu_table()   OTU Table:         [ 19216 taxa and 26 samples ]
## sample_data() Sample Data:       [ 26 samples by 7 sample variables ]
## tax_table()   Taxonomy Table:    [ 19216 taxa by 7 taxonomic ranks ]
## phy_tree()    Phylogenetic Tree: [ 19216 tips and 19215 internal nodes ]
```

```
GPfr
```

```
## phyloseq-class experiment-level object
## otu_table()   OTU Table:         [ 4624 taxa and 26 samples ]
## sample_data() Sample Data:       [ 26 samples by 7 sample variables ]
## tax_table()   Taxonomy Table:    [ 4624 taxa by 7 taxonomic ranks ]
## phy_tree()    Phylogenetic Tree: [ 4624 tips and 4623 internal nodes ]
```

Subsetting

We see that the final object, `GPfr` is highly subsetted containing just a little less than a quarter of the original OTUs (4624 / 19216).

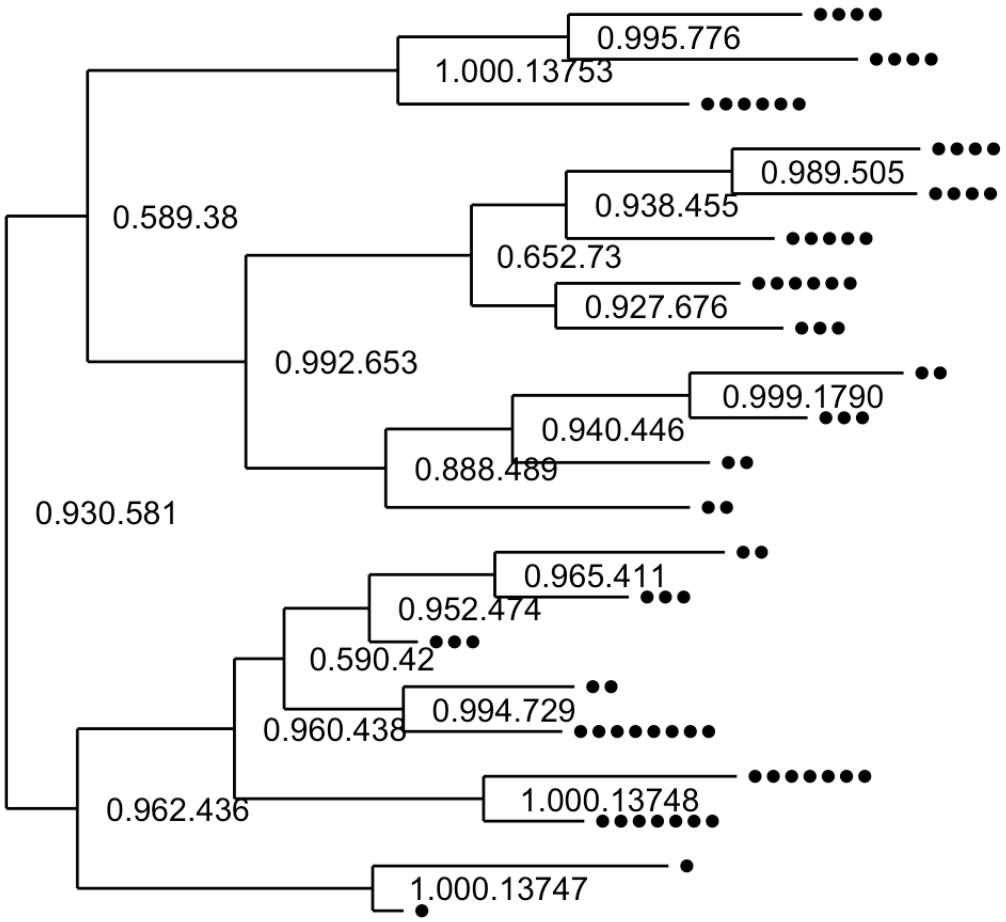
The functions `prune_sample` and `prune_taxa` are to be used in cases where the complete subset of OTUS or samples is directly available. Alternatively the `subsets_taxa` and `subset_samples` are best for subsetting based on other data contained in the taxonomy and sample datasets respectively.

As an example we will obtain the subset of the GlobalPatterns dataset that is part of the Chlamydiae phylum, then remove samples with less than 20 reads.

```
# Subset for Chlamydiae

GP.ch1 <- subset_taxa(GlobalPatterns, Phylum == "Chlamydiae")

# Eliminate samples whose total reads are less than 20

GP.ch1 <- prune_samples(sample_sums(GP.ch1) >= 20, GP.ch1)

GP.ch1
```

```
## phyloseq-class experiment-level object
## otu_table()   OTU Table:         [ 21 taxa and 9 samples ]
## sample_data() Sample Data:       [ 9 samples by 7 sample variables ]
## tax_table()   Taxonomy Table:    [ 21 taxa by 7 taxonomic ranks ]
## phy_tree()    Phylogenetic Tree: [ 21 tips and 20 internal nodes ]
```

```
plot_tree(GP.ch1)
```



Merging

Merging methods include `merge_taxa` and `merge_samples` for merging specific OTUs or samples, respectively.

The following chunk shows the merge of the first 5 OTUs in teh Chlamydiae-only dataset.

```
# Merge first 5 OTUSs in Chlamydiae-only dataset

GP.ch1.merged <- merge_taxa(GP.ch1, taxa_names(GP.ch1)[1:5])
```

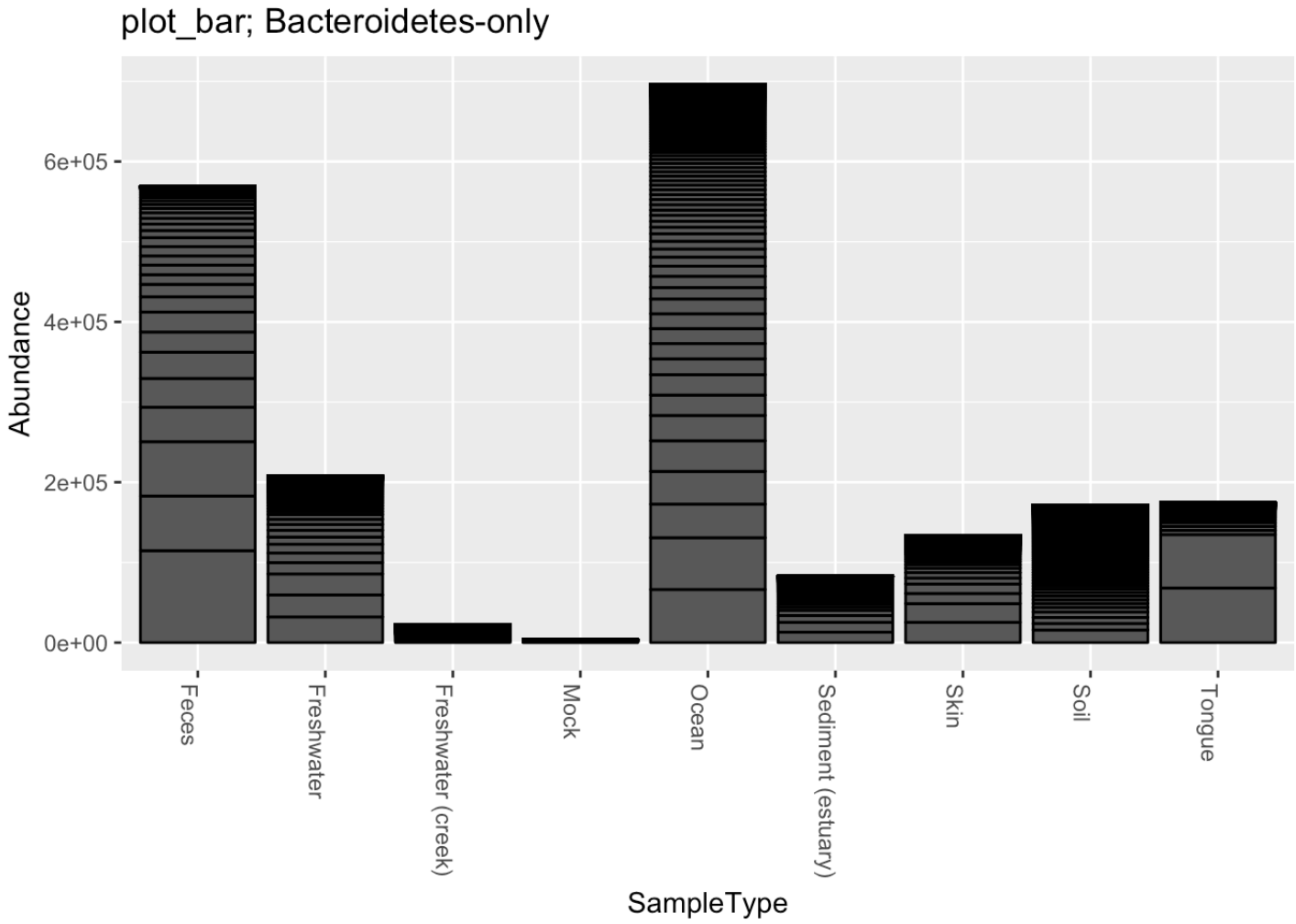
Let's do one final set of preprocessing steps

```
# Final preprocess for GlobalPatterns: filter out taxa not see more than 3 times in a
t least
# 20% of samples

GP <- filter_taxa(GlobalPatterns, function(x) sum(x > 3) > (0.2*length(x)), TRUE)

# Define human v non-human and add to sample data:

sample_data(GP)$human <- factor(get_variable(GP, "SampleType") %in% c("Feces", "Mock"
, "Skin", "Tongue"))

# Standardize abundances to median sequencing depth

total <- median(sample_sums(GP))
standf <- function(x, t=total) round(t * (x / sum(x)))
gps <- transform_sample_counts(GP, standf)

# Filter out taxa with CV > 3.0

gpsf <- filter_taxa(gps, function(x) sd(x) / mean(x) > 3.0, TRUE)

# Subset to Bacteroidetes

gpsfb <- subset_taxa(gpsf, Phylum == "Bacteroidetes")
```

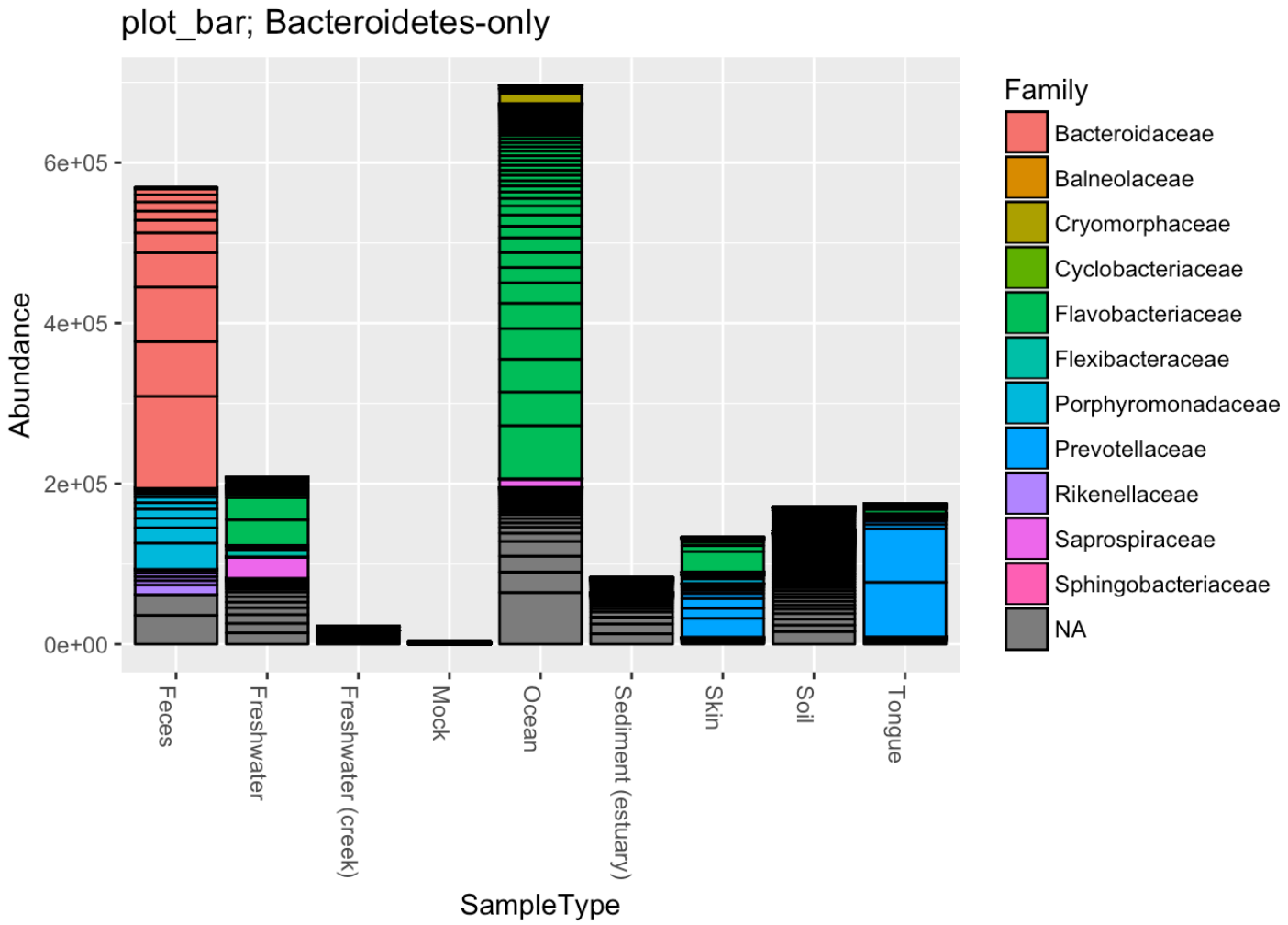
Graphical Display

Let's graph this slice of data

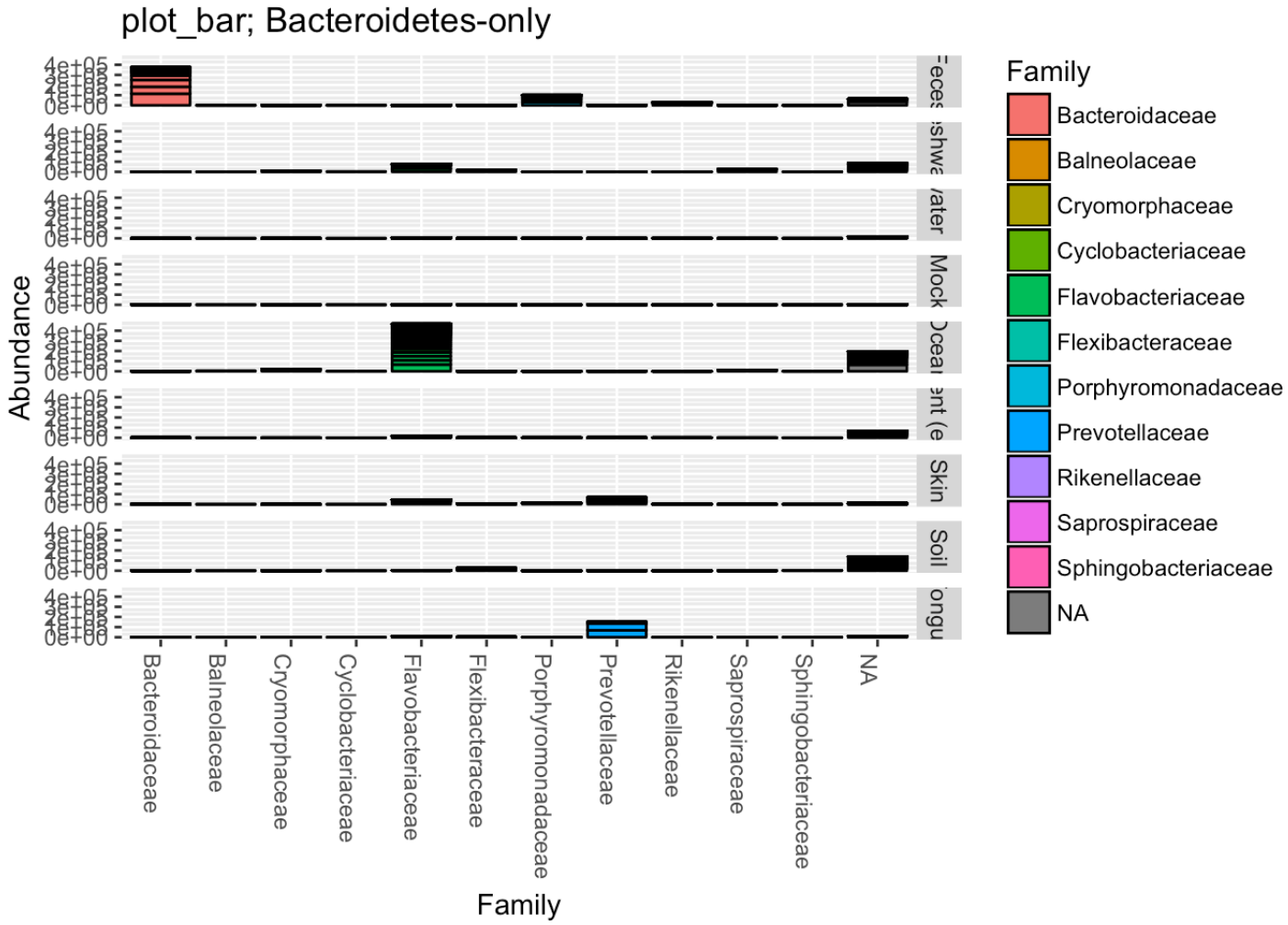
```
# Graph the slice of data

# First a simple bar graph

title = "plot_bar; Bacteroidetes-only"
plot_bar(gpsfb, "SampleType", "Abundance", title = title)
```




```
# Now colored by family

plot_bar(gpsfb, "SampleType", "Abundance", "Family", title = title)
```




```
# Now faceted with type of samples

plot_bar(gpsfb, "Family", "Abundance", "Family", title = title, facet_grid = "SampleT
ype~.")
```


plot_bar; Bacteroidetes-only
Abundance
Family
Bacteroidaceae
Balneolaceae
Cryomorphaceae
Cyclobacteriaceae
Flavobacteriaceae
Flexibacteraceae
Porphyromonadaceae
Prevotellaceae
Rikenellaceae
Saprospiraceae
Sphingobacteriaceae
NA