

Analysis of community composition data using phyloseq

Équipe Formation 16S

14th April 2016



- 1 Goals of the tutorial
- 2 `phyloseq`
- 3 Biodiversity indices
- 4 Exploring the structure
- 5 Diversity Partitioning
- 6 To Go Further

phyloseq

Learn about and become familiar with phyloseq R package for the analysis of **microbial census** data.

Exploratory Data Analysis

- **α -diversity**: how diverse is my community?
- **β -diversity**: how different are two communities?
- Use a distance matrix to study **structures**:
 - **Hierarchical clustering**: how do the communities cluster?
 - **Permutational ANOVA**: are the communities structured by some *known* environmental factor (pH, height, etc)?
- **Visual assessment** of the data
 - **bar plots**: what is the composition of each community?
 - **Multidimensional Scaling**: how are communities related?
 - **Heatmaps**: are there interactions between species and (groups of) communities?

1 Goals of the tutorial

2 phyloseq

- About phyloseq
- phyloseq data structure
- Other accessors
- Manipulating a phyloseq object: Filtering
- Manipulating a phyloseq object: Smoothing
- Manipulating a phyloseq object: Abundance counts
- Importing a phyloseq object

3 Biodiversity indices

4 Exploring the structure

- ① R package (McMurdie and Holmes, 2013) to analyze community composition data in a [phylogenetic](#) framework
- ② Community ecology functions from `vegan`, `ade4`, `picante`
- ③ Tree manipulation from `ape`
- ④ Graphics from `ggplot2`
- ⑤ Differential analysis from `DESeq2`

Installing phyloseq

From bioconductor

```
## try http if https is not available  
source("https://bioconductor.org/biocLite.R")  
biocLite("phyloseq")
```

From developer's website

```
install.packages("devtools") ## If not installed previously  
library("devtools")  
install_github("phyloseq", "joey711")
```

phyloseq comes with two vignettes

```
vignette("phyloseq-basics")  
vignette("phyloseq-analysis")
```

The first one gives insights about data structure and data manipulation (Section 2), the second one about data analysis (Section 3 to 5).

1 Goals of the tutorial

2 phyloseq

- About phyloseq
- **phyloseq data structure**
- Other accessors
- Manipulating a phyloseq object: Filtering
- Manipulating a phyloseq object: Smoothing
- Manipulating a phyloseq object: Abundance counts
- Importing a phyloseq object

3 Biodiversity indices

4 Exploring the structure

Let's get started

We first load the phyloseq package

```
library(phyloseq)
```

some additional custom function

```
source("Custom_Functions.R")
```

And load some data, `food`, kindly provided by Stéphane Chaillou et al. (2015), into the workspace

```
load("data/chaillou/chaillouISMEJ2015.RData")
```

What information do we expect to find in `food`?

```
food

## phyloseq-class experiment-level object
## otu_table() OTU Table: [ 508 taxa and 64 samples ]
## sample_data() Sample Data: [ 64 samples by 3 sample variables ]
## tax_table() Taxonomy Table: [ 508 taxa by 7 taxonomic ranks ]
## phy_tree() Phylogenetic Tree: [ 508 tips and 507 internal nodes ]
```

Let's get started

We first load the phyloseq package

```
library(phyloseq)
```

some additional custom function

```
source("Custom_Functions.R")
```

And load some data, `food`, kindly provided by Stéphane Chaillou et al. (2015), into the workspace

```
load("data/chaillou/chaillouISMEJ2015.RData")
```

What information do we expect to find in `food`?

```
food

## phyloseq-class experiment-level object
## otu_table()   OTU Table:           [ 508 taxa and 64 samples ]
## sample_data() Sample Data:        [ 64 samples by 3 sample variables ]
## tax_table()   Taxonomy Table:      [ 508 taxa by 7 taxonomic ranks ]
## phy_tree()    Phylogenetic Tree:   [ 508 tips and 507 internal nodes ]
```

Let's get started (II)

Our phyloseq object `food` is made up of four parts:

- OTU Table
- Sample Data
- Taxonomy Table
- Phylogenetic Tree

Let's have a quick look at each using the hinted at functions `otu_table`, `sample_data`, `tax_table` and `phy_tree`.

otu_table

matrix-like object

```
head(otu_table(food))
```

```
## OTU Table:           [6 taxa and 64 samples]
##                   taxa are rows
##      DLTO.LOT08 DLTO.LOT05 DLTO.LOT03 DLTO.LOT07 DLTO.LOT06
## otu_00520      0          0          0          0          0
## otu_00555      4          0          8          3          15
## otu_00568      0          0          0          0          0
## otu_00566      0          3          0          0          0
## otu_00569     12          3         26         16         38
## otu_00545      0          0          0          0          0
##      DLTO.LOT01 DLTO.LOT04 DLTO.LOT10 MVTO.LOT05 MVTO.LOT01
## otu_00520      0          0          0          0          0
## otu_00555      4          2          0          12         17
## otu_00568      0          0          0          0          0
## otu_00566      0          0          0          0          0
## otu_00569      0          4          0          10         15
## otu_00545      0          0          0          0          0
##      MVTO.LOT06 MVTO.LOT07 MVTO.LOT03 MVTO.LOT09 MVTO.LOT08
## otu_00520      0          0          0          0          0
## otu_00555     25         31         11         40         12
## otu_00568      0          0          0          0          0
## otu_00566      0          0          0          0          0
```

tax_table

matrix-like object

```
head(tax_table(food))
```

```
## Taxonomy Table:      [6 taxa by 7 taxonomic ranks]:  
##      Kingdom      Phylum      Class  
## otu_00520 "Bacteria" "Proteobacteria" "Gammaproteobacteria"  
## otu_00555 "Bacteria" "Proteobacteria" "Gammaproteobacteria"  
## otu_00568 "Bacteria" "Proteobacteria" "Gammaproteobacteria"  
## otu_00566 "Bacteria" "Proteobacteria" "Gammaproteobacteria"  
## otu_00569 "Bacteria" "Proteobacteria" "Gammaproteobacteria"  
## otu_00545 "Bacteria" "Proteobacteria" "Gammaproteobacteria"  
##      Order      Family  
## otu_00520 "Enterobacteriales" "Enterobacteriaceae"  
## otu_00555 "Enterobacteriales" "Enterobacteriaceae"  
## otu_00568 "Enterobacteriales" "Enterobacteriaceae"  
## otu_00566 "Enterobacteriales" "Enterobacteriaceae"  
## otu_00569 "Enterobacteriales" "Enterobacteriaceae"  
## otu_00545 "Enterobacteriales" "Enterobacteriaceae"  
##      Genus      Species  
## otu_00520 "Raoultella" "Ornithinolytica"  
## otu_00555 "Hafnia-Obesumbacterium" "Alveii"  
## otu_00568 "Serratia" "Fonticola"  
## otu_00566 "Serratia" "Liquefaciens"  
## otu_00569 "Serratia" "Dysenteriae"  
## otu_00545 "Serratia" "Dysenteriae"
```

data.frame-like object

```
head(sample_data(food))
```

```
## Sample Data:          [6 samples by 3 sample variables]:  
##           EnvType FoodType Description  
## DLT0.LOT08 DesLardons   Meat      LOT8  
## DLT0.LOT05 DesLardons   Meat      LOT5  
## DLT0.LOT03 DesLardons   Meat      LOT3  
## DLT0.LOT07 DesLardons   Meat      LOT7  
## DLT0.LOT06 DesLardons   Meat      LOT6  
## DLT0.LOT01 DesLardons   Meat      LOT1
```

phylo-class (tree) object

```
phy_tree(food)

##
## Phylogenetic tree with 508 tips and 507 internal nodes.
##
## Tip labels:
##  otu_00520, otu_00555, otu_00568, otu_00566, otu_00569, otu_00545, ...
##
## Rooted; includes branch lengths.
```

A phyloseq object is made of up to 5 **components** (or **slots**):

- 1 **otu_table**: an otu abundance table;
- 2 **sample_data**: a table of sample metadata, like sequencing technology, location of sampling, etc;
- 3 **tax_table**: a table of taxonomic descriptors for each otu, typically the taxonomic assignation at different levels (phylum, order, class, etc.);
- 4 **phy_tree**: a phylogenetic tree of the otus;
- 5 **refseq**: a set of reference sequences (one per otu), not present in food.

A phyloseq object is made of up to 5 **components** (or **slots**):

- 1 **otu_table**: an otu abundance table;
- 2 **sample_data**: a table of sample metadata, like sequencing technology, location of sampling, etc;
- 3 **tax_table**: a table of taxonomic descriptors for each otu, typically the taxonomic assignation at different levels (phylum, order, class, etc.);
- 4 **phy_tree**: a phylogenetic tree of the otus;
- 5 **refseq**: a set of reference sequences (one per otu), not present in food.

A phyloseq object is made of up to 5 **components** (or **slots**):

- 1 **otu_table**: an otu abundance table;
- 2 **sample_data**: a table of sample metadata, like sequencing technology, location of sampling, etc;
- 3 **tax_table**: a table of taxonomic descriptors for each otu, typically the taxonomic assignation at different levels (phylum, order, class, etc.);
- 4 **phy_tree**: a phylogenetic tree of the otus;
- 5 **refseq**: a set of reference sequences (one per otu), not present in food.

A phyloseq object is made of up to 5 **components** (or **slots**):

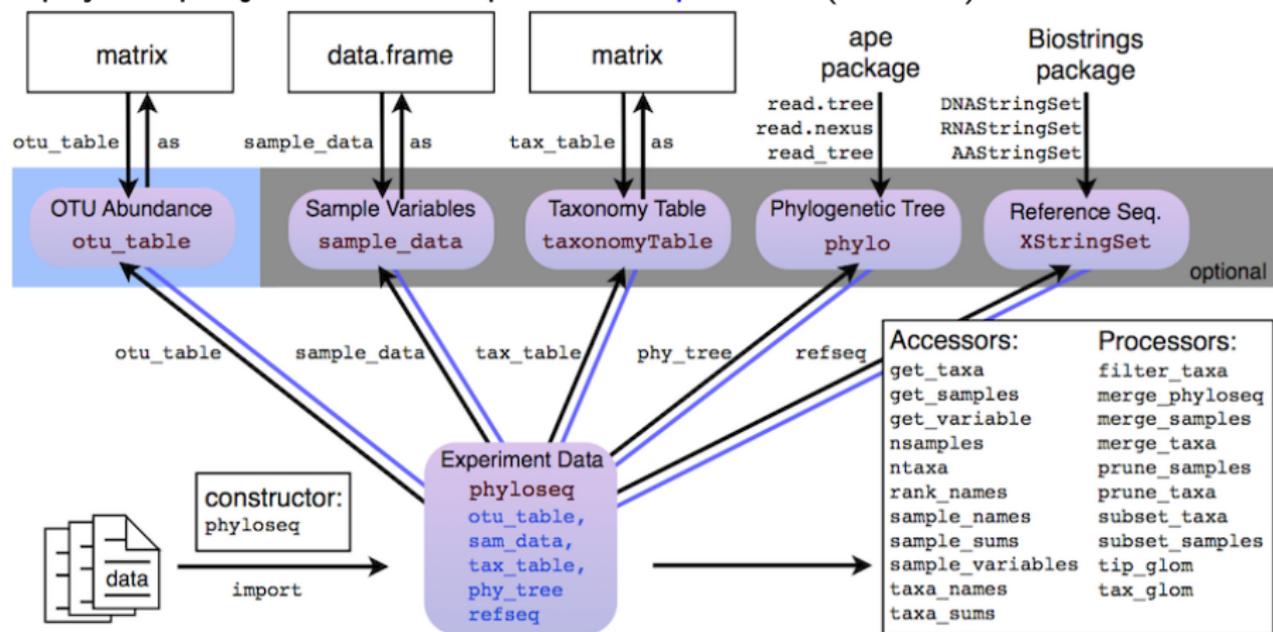
- 1 **otu_table**: an otu abundance table;
- 2 **sample_data**: a table of sample metadata, like sequencing technology, location of sampling, etc;
- 3 **tax_table**: a table of taxonomic descriptors for each otu, typically the taxonomic assignation at different levels (phylum, order, class, etc.);
- 4 **phy_tree**: a phylogenetic tree of the otus;
- 5 **refseq**: a set of reference sequences (one per otu), not present in food.

A phyloseq object is made of up to 5 **components** (or **slots**):

- 1 **otu_table**: an otu abundance table;
- 2 **sample_data**: a table of sample metadata, like sequencing technology, location of sampling, etc;
- 3 **tax_table**: a table of taxonomic descriptors for each otu, typically the taxonomic assignation at different levels (phylum, order, class, etc.);
- 4 **phy_tree**: a phylogenetic tree of the otus;
- 5 **refseq**: a set of reference sequences (one per otu), not present in food.

Data structure (II)

A phyloseq object is made up of 5 **components** (or **slots**):



1 Goals of the tutorial

2 phyloseq

- About phyloseq
- phyloseq data structure
- **Other accessors**
- Manipulating a phyloseq object: Filtering
- Manipulating a phyloseq object: Smoothing
- Manipulating a phyloseq object: Abundance counts
- Importing a phyloseq object

3 Biodiversity indices

4 Exploring the structure

phyloseq also offers the following [accessors](#):

- `ntaxa / nsamples`
- `sample_names / taxa_names`
- `sample_sums / taxa_sums`
- `rank_names`
- `sample_variables`
- `get_taxa`
- `get_samples`
- `get_variable`

to extract parts of a phyloseq object.

Try them on your own (on `food`) and guess what they do.

phyloseq also offers the following [accessors](#):

- `ntaxa / nsamples`
- `sample_names / taxa_names`
- `sample_sums / taxa_sums`
- `rank_names`
- `sample_variables`
- `get_taxa`
- `get_samples`
- `get_variable`

to extract parts of a phyloseq object.

Try them on your own (on `food`) and guess what they do.

Dimensions

```
ntaxa(food)
```

```
## [1] 508
```

```
nsamples(food)
```

```
## [1] 64
```

- `ntaxa` returns the number of taxa;
- `nsamples` returns the number of samples;

```
ntaxa(food)
## [1] 508

nsamples(food)
## [1] 64
```

- `ntaxa` returns the number of taxa;
- `nsamples` returns the number of samples;

sample_names, taxa_names

```
head(sample_names(food))
```

```
## [1] "DLT0.LOT08" "DLT0.LOT05" "DLT0.LOT03" "DLT0.LOT07" "DLT0.LOT06"  
## [6] "DLT0.LOT01"
```

```
head(taxa_names(food))
```

```
## [1] "otu_00520" "otu_00555" "otu_00568" "otu_00566" "otu_00569" "otu_005
```

Names of the samples and taxa included in the phyloseq object.

sample_names, taxa_names

```
head(sample_names(food))
```

```
## [1] "DLT0.LOT08" "DLT0.LOT05" "DLT0.LOT03" "DLT0.LOT07" "DLT0.LOT06"  
## [6] "DLT0.LOT01"
```

```
head(taxa_names(food))
```

```
## [1] "otu_00520" "otu_00555" "otu_00568" "otu_00566" "otu_00569" "otu_005
```

Names of the **samples** and taxa included in the phyloseq object.

sample_names, taxa_names

```
head(sample_names(food))
```

```
## [1] "DLT0.LOT08" "DLT0.LOT05" "DLT0.LOT03" "DLT0.LOT07" "DLT0.LOT06"  
## [6] "DLT0.LOT01"
```

```
head(taxa_names(food))
```

```
## [1] "otu_00520" "otu_00555" "otu_00568" "otu_00566" "otu_00569" "otu_00567"
```

Names of the samples and **taxa** included in the phyloseq object.

sample_sums, taxa_sums

```
head(sample_sums(food))
```

```
## DLT0.LOT08 DLT0.LOT05 DLT0.LOT03 DLT0.LOT07 DLT0.LOT06 DLT0.LOT01  
##      11812      11787      11804      11806      11832      11857
```

```
head(taxa_sums(food))
```

```
## otu_00520 otu_00555 otu_00568 otu_00566 otu_00569 otu_00545  
##      55      395      22      13      1998      210
```

Total count of each sample (*i.e.* sample library sizes) or of each taxa (*i.e.* overall abundances across all samples)

sample_sums, taxa_sums

```
head(sample_sums(food))
```

```
## DLT0.LOT08 DLT0.LOT05 DLT0.LOT03 DLT0.LOT07 DLT0.LOT06 DLT0.LOT01  
##      11812      11787      11804      11806      11832      11857
```

```
head(taxa_sums(food))
```

```
## otu_00520 otu_00555 otu_00568 otu_00566 otu_00569 otu_00545  
##      55      395      22      13      1998      210
```

Total count of each **sample** (*i.e.* sample library sizes) or of each taxa (*i.e.* overall abundances across all samples)

sample_sums, taxa_sums

```
head(sample_sums(food))
```

```
## DLT0.LOT08 DLT0.LOT05 DLT0.LOT03 DLT0.LOT07 DLT0.LOT06 DLT0.LOT01  
##      11812      11787      11804      11806      11832      11857
```

```
head(taxa_sums(food))
```

```
## otu_00520 otu_00555 otu_00568 otu_00566 otu_00569 otu_00545  
##      55      395      22      13      1998      210
```

Total count of each sample (*i.e.* sample library sizes) or of each **taxa** (*i.e.* overall abundances across all samples)

rank_names

```
rank_names(food)

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

Names of the taxonomic levels available in the `tax_table` slot.

rank_names

```
rank_names(food)

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

Names of the **taxonomic levels** available in the `tax_table` slot.

sample_variables

```
head(sample_variables(food))  
## [1] "EnvType"      "FoodType"     "Description"
```

Names of the contextual data recorded on the samples.

sample_variables

```
head(sample_variables(food))  
## [1] "EnvType"      "FoodType"     "Description"
```

Names of the **contextual data** recorded on the samples.

A little exercise

Find the

- library size of samples MVT0.LOT01, MVT0.LOT07, MVT0.LOT09
- overall abundance of otus otu_00520, otu_00569, otu_00527

Hint: What's the class of `sample_sums(food)` and `taxa_sums(food)`?
How do you index them?

```
## sample library sizes
sample_sums(food)[c("MVT0.LOT01", "MVT0.LOT07", "MVT0.LOT09")]
```

```
## MVT0.LOT01 MVT0.LOT07 MVT0.LOT09
##      11743      11765      11739
```

```
## Otu overall abundances
taxa_sums(food)[c("otu_00520", "otu_00569", "otu_00527")]
```

```
## otu_00520 otu_00569 otu_00527
##      55      1998      58
```

A little exercise

Find the

- library size of samples MVT0.LOT01, MVT0.LOT07, MVT0.LOT09
- overall abundance of otus otu_00520, otu_00569, otu_00527

Hint: What's the class of `sample_sums(food)` and `taxa_sums(food)`?
How do you index them?

```
## sample library sizes
sample_sums(food)[c("MVT0.LOT01", "MVT0.LOT07", "MVT0.LOT09")]
```

```
## MVT0.LOT01 MVT0.LOT07 MVT0.LOT09
##      11743      11765      11739
```

```
## Otu overall abundances
taxa_sums(food)[c("otu_00520", "otu_00569", "otu_00527")]
```

```
## otu_00520 otu_00569 otu_00527
##      55      1998      58
```

get_variable, get_sample, get_taxa

```
head(get_variable(food, varName = "EnvType"))

## [1] DesLardons DesLardons DesLardons DesLardons DesLardons DesLardons
## 8 Levels: BoeufHache VeauHache DesLardons MerguezVolaille ... Crevette

head(get_sample(food, i = "otu_00520"))

## DLTO.LOT08 DLTO.LOT05 DLTO.LOT03 DLTO.LOT07 DLTO.LOT06 DLTO.LOT01
##          0          0          0          0          0          0

head(get_taxa(food, i = "MVT0.LOT07"))

## otu_00520 otu_00555 otu_00568 otu_00566 otu_00569 otu_00545
##          0          31          0          0          35          0
```

- values for variable `varName` in sample data
- abundance values of otu `i` in all samples (row of OTU table).
- abundance values of all otus in sample `i` (column of OTU table)

get_variable, get_sample, get_taxa

```
head(get_variable(food, varName = "EnvType"))

## [1] DesLardons DesLardons DesLardons DesLardons DesLardons DesLardons
## 8 Levels: BoeufHache VeauHache DesLardons MerguezVolaille ... Crevette

head(get_sample(food, i = "otu_00520"))

## DLTO.LOT08 DLTO.LOT05 DLTO.LOT03 DLTO.LOT07 DLTO.LOT06 DLTO.LOT01
##          0          0          0          0          0          0

head(get_taxa(food, i = "MVT0.LOT07"))

## otu_00520 otu_00555 otu_00568 otu_00566 otu_00569 otu_00545
##          0          31          0          0          35          0
```

- values for variable `varName` in sample data
- abundance values of otu `i` in all samples (row of OTU table).
- abundance values of all otus in sample `i` (column of OTU table)

get_variable, get_sample, get_taxa

```
head(get_variable(food, varName = "EnvType"))

## [1] DesLardons DesLardons DesLardons DesLardons DesLardons DesLardons
## 8 Levels: BoeufHache VeauHache DesLardons MerguezVolaille ... Crevette

head(get_sample(food, i = "otu_00520"))

## DLTO.LOT08 DLTO.LOT05 DLTO.LOT03 DLTO.LOT07 DLTO.LOT06 DLTO.LOT01
##          0          0          0          0          0          0

head(get_taxa(food, i = "MVT0.LOT07"))

## otu_00520 otu_00555 otu_00568 otu_00566 otu_00569 otu_00545
##          0          31          0          0          35          0
```

- values for variable `varName` in sample data
- **abundance values** of `otu i` in all samples (row of OTU table).
- abundance values of all otus in sample `i` (column of OTU table)

get_variable, get_sample, get_taxa

```
head(get_variable(food, varName = "EnvType"))

## [1] DesLardons DesLardons DesLardons DesLardons DesLardons DesLardons
## 8 Levels: BoeufHache VeauHache DesLardons MerguezVolaille ... Crevette

head(get_sample(food, i = "otu_00520"))

## DLTO.LOT08 DLTO.LOT05 DLTO.LOT03 DLTO.LOT07 DLTO.LOT06 DLTO.LOT01
##          0          0          0          0          0          0

head(get_taxa(food, i = "MVT0.LOT07"))

## otu_00520 otu_00555 otu_00568 otu_00566 otu_00569 otu_00545
##          0          31          0          0          35          0
```

- values for variable `varName` in sample data
- abundance values of otu `i` in all samples (row of OTU table).
- **abundance values** of all otus in **sample `i`** (column of OTU table)

Modifying some values

To modify parts of a phyloseq object, we **must** use (high-levels) accessors such as `otu_table`.

To transform `EnvType` to a factor with meaningful level ordering (meat products first and seafood second), we must use `sample_data`:

```
correct.order <- c("BoeufHache", "VeauHache", "DesLardons",
                  "MerguezVolaille", "SaumonFume", "FiletSaumon",
                  "FiletCabillaud", "Crevette")
sample_data(food)$EnvType <- factor(sample_data(food)$EnvType,
                                   levels = correct.order)
levels(get_variable(food, "EnvType"))

## [1] "BoeufHache"      "VeauHache"      "DesLardons"    "MerguezVolaille"
## [5] "SaumonFume"     "FiletSaumon"   "FiletCabillaud" "Crevette"
```

Likewise, to modify the count of OTU `otu_00520` in sample `DLT0.LOT08`, or its species affiliation we would do

```
otu_table(food)["otu_00520", "DLT0.LOT08"] <- 0
tax_table(food)["otu_00520", "Species"] <- "Ornithinolytica"
```

Modifying some values

To modify parts of a phyloseq object, we **must** use (high-levels) accessors such as `otu_table`.

To transform `EnvType` to a factor with meaningful level ordering (meat products first and seafood second), we must use `sample_data`:

```
correct.order <- c("BoeufHache", "VeauHache", "DesLardons",
                  "MerguezVolaille", "SaumonFume", "FiletSaumon",
                  "FiletCabillaud", "Crevette")
sample_data(food)$EnvType <- factor(sample_data(food)$EnvType,
                                   levels = correct.order)
levels(get_variable(food, "EnvType"))

## [1] "BoeufHache"      "VeauHache"      "DesLardons"    "MerguezVolaille"
## [5] "SaumonFume"     "FiletSaumon"   "FiletCabillaud" "Crevette"
```

Likewise, to modify the count of OTU `otu_00520` in sample `DLT0.LOT08`, or its species affiliation we would do

```
otu_table(food)["otu_00520", "DLT0.LOT08"] <- 0
tax_table(food)["otu_00520", "Species"] <- "Ornithinolytica"
```

Modifying some values

To modify parts of a phyloseq object, we **must** use (high-levels) accessors such as `otu_table`.

To transform `EnvType` to a factor with meaningful level ordering (meat products first and seafood second), we must use `sample_data`:

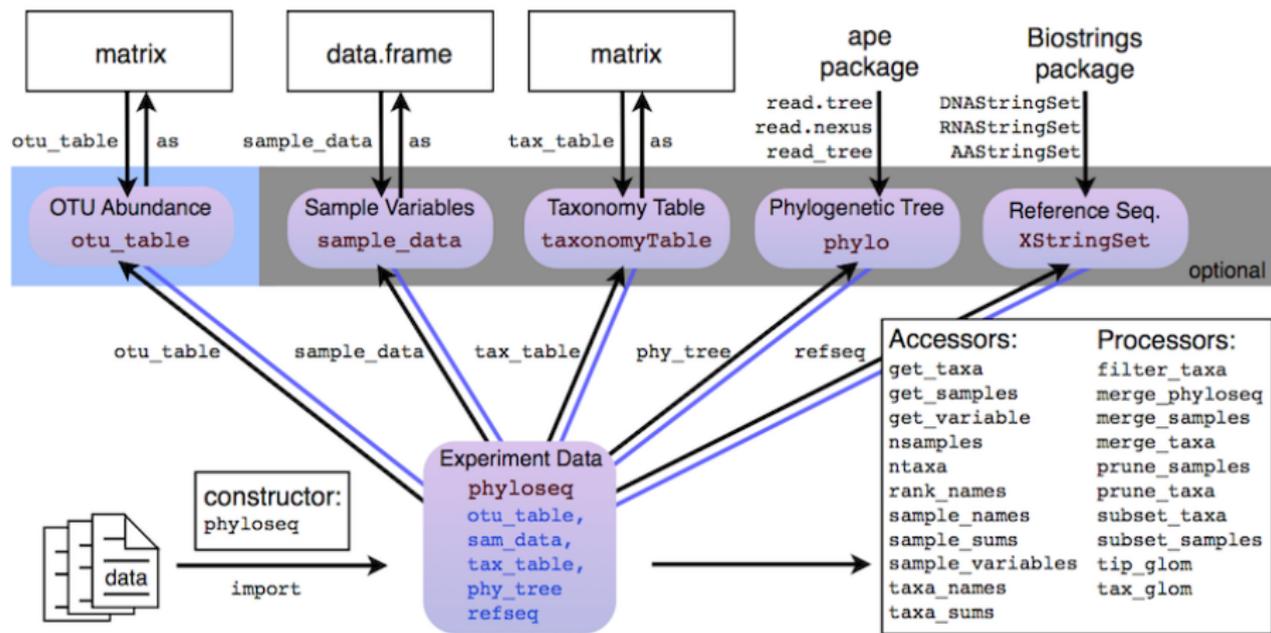
```
correct.order <- c("BoeufHache", "VeauHache", "DesLardons",
                  "MerguezVolaille", "SaumonFume", "FiletSaumon",
                  "FiletCabillaud", "Crevette")
sample_data(food)$EnvType <- factor(sample_data(food)$EnvType,
                                   levels = correct.order)
levels(get_variable(food, "EnvType"))

## [1] "BoeufHache"      "VeauHache"      "DesLardons"     "MerguezVolaille"
## [5] "SaumonFume"     "FiletSaumon"   "FiletCabillaud" "Crevette"
```

Likewise, to modify the count of OTU `otu_00520` in sample `DLT0.LOT08`, or its species affiliation we would do

```
otu_table(food)["otu_00520", "DLT0.LOT08"] <- 0
tax_table(food)["otu_00520", "Species"] <- "Ornithinolytica"
```

Data structure Recap



1 Goals of the tutorial

2 phyloseq

- About phyloseq
- phyloseq data structure
- Other accessors
- **Manipulating a phyloseq object: Filtering**
- Manipulating a phyloseq object: Smoothing
- Manipulating a phyloseq object: Abundance counts
- Importing a phyloseq object

3 Biodiversity indices

4 Exploring the structure

Filtering via prune, subset and filter (I)

Prune

- `prune_taxa` (`prune_samples`) prunes unwanted `taxa` (samples) from a phyloseq object based on a vector of taxa to keep
- The taxa are passed as a vector `taxa` of character (otu1, otu4) or of logical (TRUE, FALSE, FALSE, TRUE)
- `prune_taxa(taxa, physeq)` would keep only otus otu1, otu4

Subset

- `subset_taxa` (`subset_samples`) subsets unwanted taxa (samples) from a phyloseq object based on conditions that must be met
- The conditions (any number) can apply to any descriptor (e.g. taxonomy) of the otus included in the phyloseq object `physeq`
- `subset_taxa(physeq, Phylum == "Firmicutes")` would keep only Firmicutes.

Filtering via prune, subset and filter (I)

Prune

- `prune_taxa` (`prune_samples`) prunes unwanted taxa (`samples`) from a phyloseq object based on a vector of taxa to keep
- The taxa are passed as a vector `taxa` of character (otu1, otu4) or of logical (TRUE, FALSE, FALSE, TRUE)
- `prune_taxa(taxa, physeq)` would keep only otus otu1, otu4

Subset

- `subset_taxa` (`subset_samples`) subsets unwanted taxa (`samples`) from a phyloseq object based on conditions that must be met
- The conditions (any number) can apply to any descriptor (e.g. taxonomy) of the otus included in the phyloseq object `physeq`
- `subset_taxa(physeq, Phylum == "Firmicutes")` would keep only Firmicutes.

Filtering via prune, subset and filter (I)

Prune

- `prune_taxa` (`prune_samples`) prunes unwanted taxa (samples) from a phyloseq object based on a **vector of taxa to keep**
- The taxa are passed as a vector `taxa` of character (otu1, otu4) or of logical (TRUE, FALSE, FALSE, TRUE)
- `prune_taxa(taxa, physeq)` would keep only otus otu1, otu4

Subset

- `subset_taxa` (`subset_samples`) subsets unwanted taxa (samples) from a phyloseq object based on conditions that must be met
- The conditions (any number) can apply to any descriptor (e.g. taxonomy) of the otus included in the phyloseq object `physeq`
- `subset_taxa(physeq, Phylum == "Firmicutes")` would keep only Firmicutes.

Filtering via prune, subset and filter (I)

Prune

- `prune_taxa` (`prune_samples`) prunes unwanted taxa (samples) from a phyloseq object based on a vector of taxa to keep
- The taxa are passed as a vector `taxa` of character (otu1, otu4) or of logical (TRUE, FALSE, FALSE, TRUE)
- `prune_taxa(taxa, physeq)` would keep only otus otu1, otu4

Subset

- `subset_taxa` (`subset_samples`) subsets unwanted taxa (samples) from a phyloseq object based on conditions that must be met
- The conditions (any number) can apply to any descriptor (e.g. taxonomy) of the otus included in the phyloseq object `physeq`
- `subset_taxa(physeq, Phylum == "Firmicutes")` would keep only Firmicutes.

Filtering via prune, subset and filter (I)

Prune

- `prune_taxa` (`prune_samples`) prunes unwanted taxa (samples) from a phyloseq object based on a vector of taxa to keep
- The taxa are passed as a vector `taxa` of character (otu1, otu4) or of logical (TRUE, FALSE, FALSE, TRUE)
- `prune_taxa(taxa, physeq)` would keep only otus otu1, otu4

Subset

- `subset_taxa` (`subset_samples`) subsets unwanted taxa (samples) from a phyloseq object based on conditions that must be met
- The conditions (any number) can apply to any `descriptor` (e.g. taxonomy) of the otus included in the phyloseq object `physeq`
- `subset_taxa(physeq, Phylum == "Firmicutes")` would keep only Firmicutes.

Filtering via prune, subset and filter (I)

Prune

- `prune_taxa` (`prune_samples`) prunes unwanted taxa (samples) from a phyloseq object based on a vector of taxa to keep
- The taxa are passed as a vector `taxa` of character (otu1, otu4) or of logical (TRUE, FALSE, FALSE, TRUE)
- `prune_taxa(taxa, physeq)` would keep only otus otu1, otu4

Subset

- `subset_taxa` (`subset_samples`) subsets unwanted taxa (`samples`) from a phyloseq object based on conditions that must be met
- The conditions (any number) can apply to any `descriptor` (e.g. taxonomy) of the otus included in the phyloseq object `physeq`
- `subset_taxa(physeq, Phylum == "Firmicutes")` would keep only Firmicutes.

Filtering via prune, subset and filter (I)

Prune

- `prune_taxa` (`prune_samples`) prunes unwanted taxa (samples) from a phyloseq object based on a vector of taxa to keep
- The taxa are passed as a vector `taxa` of character (otu1, otu4) or of logical (TRUE, FALSE, FALSE, TRUE)
- `prune_taxa(taxa, physeq)` would keep only otus otu1, otu4

Subset

- `subset_taxa` (`subset_samples`) subsets unwanted taxa (samples) from a phyloseq object based on **conditions that must be met**
- The conditions (any number) can apply to any **descriptor** (e.g. taxonomy) of the otus included in the phyloseq object `physeq`
- `subset_taxa(physeq, Phylum == "Firmicutes")` would keep only Firmicutes.

Filtering via prune, subset and filter (I)

Prune

- `prune_taxa` (`prune_samples`) prunes unwanted taxa (samples) from a phyloseq object based on a vector of taxa to keep
- The taxa are passed as a vector `taxa` of character (otu1, otu4) or of logical (TRUE, FALSE, FALSE, TRUE)
- `prune_taxa(taxa, physeq)` would keep only otus otu1, otu4

Subset

- `subset_taxa` (`subset_samples`) subsets unwanted taxa (samples) from a phyloseq object based on conditions that must be met
- The conditions (any number) can apply to any `descriptor` (e.g. taxonomy) of the otus included in the phyloseq object `physeq`
- `subset_taxa(physeq, Phylum == "Firmicutes")` would keep only Firmicutes.

Prune and subset

Prune

```
samplesToKeep <- sample_names(food)[1:10]
prune_samples(samplesToKeep, food)

## phyloseq-class experiment-level object
## otu_table() OTU Table: [ 508 taxa and 10 samples ]
## sample_data() Sample Data: [ 10 samples by 3 sample variables ]
## tax_table() Taxonomy Table: [ 508 taxa by 7 taxonomic ranks ]
## phy_tree() Phylogenetic Tree: [ 508 tips and 507 internal nodes ]
```

Subset

```
subset_samples(food, EnvType %in% c("DesLardons", "MerguezVolaille"))

## phyloseq-class experiment-level object
## otu_table() OTU Table: [ 508 taxa and 16 samples ]
## sample_data() Sample Data: [ 16 samples by 3 sample variables ]
## tax_table() Taxonomy Table: [ 508 taxa by 7 taxonomic ranks ]
## phy_tree() Phylogenetic Tree: [ 508 tips and 507 internal nodes ]
```

Prune and subset

Prune

```
samplesToKeep <- sample_names(food)[1:10]
prune_samples(samplesToKeep, food)

## phyloseq-class experiment-level object
## otu_table() OTU Table: [ 508 taxa and 10 samples ]
## sample_data() Sample Data: [ 10 samples by 3 sample variables ]
## tax_table() Taxonomy Table: [ 508 taxa by 7 taxonomic ranks ]
## phy_tree() Phylogenetic Tree: [ 508 tips and 507 internal nodes ]
```

Subset

```
subset_samples(food, EnvType %in% c("DesLardons", "MerguezVolaille"))

## phyloseq-class experiment-level object
## otu_table() OTU Table: [ 508 taxa and 16 samples ]
## sample_data() Sample Data: [ 16 samples by 3 sample variables ]
## tax_table() Taxonomy Table: [ 508 taxa by 7 taxonomic ranks ]
## phy_tree() Phylogenetic Tree: [ 508 tips and 507 internal nodes ]
```

A bit more about subset (II)

Multiple conditions can be combined with the usual logical operator (& for AND and | for OR)

```
small.food <- subset_taxa(food, Phylum == "Firmicutes" & Class == "Bacilli")
head(tax_table(small.food)[ , c("Phylum", "Class", "Order")])
```

```
## Taxonomy Table:      [6 taxa by 3 taxonomic ranks]:
##      Phylum      Class      Order
## otu_00583 "Firmicutes" "Bacilli" "Lactobacillales"
## otu_00574 "Firmicutes" "Bacilli" "Lactobacillales"
## otu_00581 "Firmicutes" "Bacilli" "Lactobacillales"
## otu_00591 "Firmicutes" "Bacilli" "Lactobacillales"
## otu_00582 "Firmicutes" "Bacilli" "Lactobacillales"
## otu_00586 "Firmicutes" "Bacilli" "Lactobacillales"
```

```
## Unique combinations (Phylum, Class)
unique(tax_table(small.food)[ , c("Phylum", "Class")])
```

```
## Taxonomy Table:      [1 taxa by 2 taxonomic ranks]:
##      Phylum      Class
## otu_00583 "Firmicutes" "Bacilli"
```

1 Goals of the tutorial

2 phyloseq

- About phyloseq
- phyloseq data structure
- Other accessors
- Manipulating a phyloseq object: Filtering
- **Manipulating a phyloseq object: Smoothing**
- Manipulating a phyloseq object: Abundance counts
- Importing a phyloseq object

3 Biodiversity indices

4 Exploring the structure

Smoothing with `tax_glom` (I)

`tax_glom` agglomerates otus at a given **taxonomic level**. Finer taxonomic information is lost.

```
mergedData <- tax_glom(food, "Phylum")
ntaxa(mergedData) ## number of different phyla

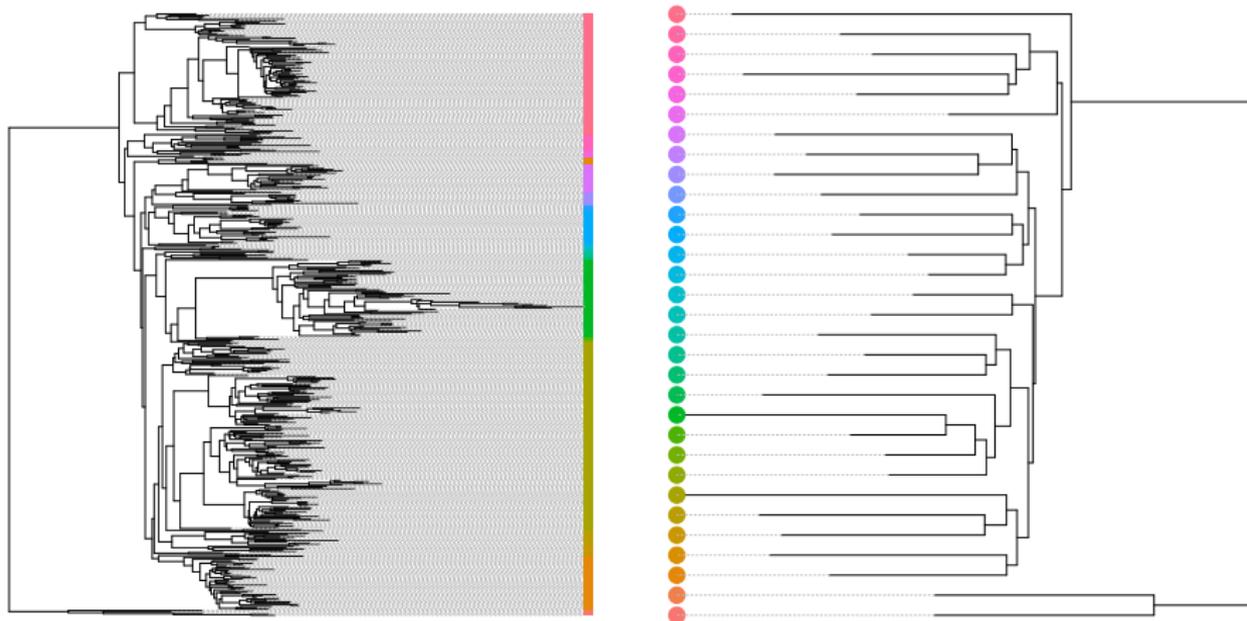
## [1] 11

tax_table(mergedData)[1:2, c("Phylum", "Order", "Class")]

## Taxonomy Table:      [2 taxa by 3 taxonomic ranks]:
##           Phylum      Order Class
## otu_01101 "Proteobacteria" NA     NA
## otu_01152 "Actinobacteria" NA     NA
```

Smoothing with `tax_glom` (II)

The effect of `tax_glom` is most obvious and best understood on the phylogenetic tree (otus are colored by phylum).



1 Goals of the tutorial

2 phyloseq

- About phyloseq
- phyloseq data structure
- Other accessors
- Manipulating a phyloseq object: Filtering
- Manipulating a phyloseq object: Smoothing
- **Manipulating a phyloseq object: Abundance counts**
- Importing a phyloseq object

3 Biodiversity indices

4 Exploring the structure

Rarefaction with `rarefy_even_depth`

`rarefy_even_depth` **downsamples** all samples to the same depth and **prunes** otus that disappear from all samples as a result.

```
foodRare <- rarefy_even_depth(food, rngseed = 1121983)

## 'set.seed(1121983)' was used to initialize repeatable random
## subsampling.
## Please record this for your records so others can reproduce.
## Try 'set.seed(1121983); .Random.seed' for the full vector
## ...
## 10TUs were removed because they are no longer
## present in any sample after random subsampling
##
## ...

sample_sums(foodRare)[1:5]

## DLTO.LOT08 DLTO.LOT05 DLTO.LOT03 DLTO.LOT07 DLTO.LOT06
##      11718      11718      11718      11718      11718
```

Transforming abundance counts with `transform_sample_counts`

`transform_sample_counts` applies a function to the **abundance vector** of each sample. It can be useful for normalization. For example:

```
count_to_prop <- function(x) { return( x / sum(x) ) }
```

transforms counts to proportions.

```
foodTrans <- transform_sample_counts(food, count_to_prop)
sample_sums(foodTrans)[1:5] ## should be 1

## DLT0.LOT08 DLT0.LOT05 DLT0.LOT03 DLT0.LOT07 DLT0.LOT06
##           1           1           1           1           1
```

1 Goals of the tutorial

2 phyloseq

- About phyloseq
- phyloseq data structure
- Other accessors
- Manipulating a phyloseq object: Filtering
- Manipulating a phyloseq object: Smoothing
- Manipulating a phyloseq object: Abundance counts
- **Importing a phyloseq object**

3 Biodiversity indices

4 Exploring the structure

From a biom dataset: `import_biom`

The biom format **natively** supports

- otu count tables (the `otu_table`)
- otu description (the `tax_table`)
- sample description (the `sample_data`)

The other components are optional and must be stored in separate files

- phylogenetic tree in Newick format (the `phy_tree`)
- sequences in fasta format ((the `refseq`))

For our purpose, the taxonomy should be in greengenes (*à la* qiime) format (FROGS does it): "k__Bacteria", "p__Proteobacteria", "c__Gammaproteobacteria", "o__Enterobacteriales"

From a biom dataset: `import_biom`

The biom format natively supports

- otu count tables (the `otu_table`)
- otu description (the `tax_table`)
- sample description (the `sample_data`)

The other components are **optional** and must be stored in separate files

- phylogenetic tree in Newick format (the `phy_tree`)
- sequences in fasta format ((the `refset`))

For our purpose, the taxonomy should be in greengenes (*à la* qiime) format (FROGS does it): "k__Bacteria", "p__Proteobacteria", "c__Gammaproteobacteria", "o__Enterobacteriales"

From a biom dataset: `import_biom`

The biom format natively supports

- otu count tables (the `otu_table`)
- otu description (the `tax_table`)
- sample description (the `sample_data`)

The other components are optional and must be stored in separate files

- phylogenetic tree in Newick format (the `phy_tree`)
- sequences in fasta format ((the `refseq`))

For our purpose, the taxonomy should be in `greengenes` (*à la qiime*) format (FROGS does it): "k__Bacteria", "p__Proteobacteria", "c__Gammaproteobacteria", "o__Enterobacteriales"

import_biom: example

Our toy dataset includes a biom, a tree and a set of references sequences.

```
biomfile <- "data/chaillou/chaillou.biom"  
treefile <- "data/chaillou/tree.nwk"
```

The import is quite easy (our specific `parseFunction` is used for greengenes formatted taxonomy)

```
manual.food <- import_biom(biomfile, treefile,  
                           parseFunction = parse_taxonomy_greengenes)  
manual.food  
  
## phyloseq-class experiment-level object  
## otu_table() OTU Table: [ 508 taxa and 64 samples ]  
## sample_data() Sample Data: [ 64 samples by 3 sample variables ]  
## tax_table() Taxonomy Table: [ 508 taxa by 7 taxonomic ranks ]  
## phy_tree() Phylogenetic Tree: [ 508 tips and 507 internal nodes ]
```

A nice data structure to store the **count table**, **taxonomic information**, **contextual data** and **phylogenetic tree** as different components of a single R object .

- Functions to import data from `biom` files, `qiime` output files or plain tabular files.
- Accessors to access different component of your dataset
- Samples and taxa names are coherent between the different components.
- Filters to keep only part of the dataset.
- Smoothers to aggregate parts of the dataset.
- Manipulators to rarefy and transform samples

A nice data structure to store the **count table**, **taxonomic information**, **contextual data** and **phylogenetic tree** as different components of a single R object .

- Functions to **import** data from `biom` files, `qiime` output files or plain tabular files.
- **Accessors** to access different component of your dataset
- Samples and taxa names are **coherent** between the different components.
- **Filters** to keep only part of the dataset.
- **Smoothers** to aggregate parts of the dataset.
- **Manipulators** to rarefy and transform samples

A nice data structure to store the **count table**, **taxonomic information**, **contextual data** and **phylogenetic tree** as different components of a single R object .

- Functions to **import** data from biom files, qiime output files or plain tabular files.
- **Accessors** to access different component of your dataset
- Samples and taxa names are **coherent** between the different components.
- **Filters** to keep only part of the dataset.
- **Smoothers** to aggregate parts of the dataset.
- **Manipulators** to rarefy and transform samples

A nice data structure to store the **count table**, **taxonomic information**, **contextual data** and **phylogenetic tree** as different components of a single R object .

- Functions to **import** data from biom files, qiime output files or plain tabular files.
- **Accessors** to access different component of your dataset
- Samples and taxa names are **coherent** between the different components.
- **Filters** to keep only part of the dataset.
- **Smoothers** to aggregate parts of the dataset.
- **Manipulators** to rarefy and transform samples

A nice data structure to store the **count table**, **taxonomic information**, **contextual data** and **phylogenetic tree** as different components of a single R object .

- Functions to **import** data from biom files, qiime output files or plain tabular files.
- **Accessors** to access different component of your dataset
- Samples and taxa names are **coherent** between the different components.
- **Filters** to keep only part of the dataset.
- **Smoothers** to aggregate parts of the dataset.
- **Manipulators** to rarefy and transform samples

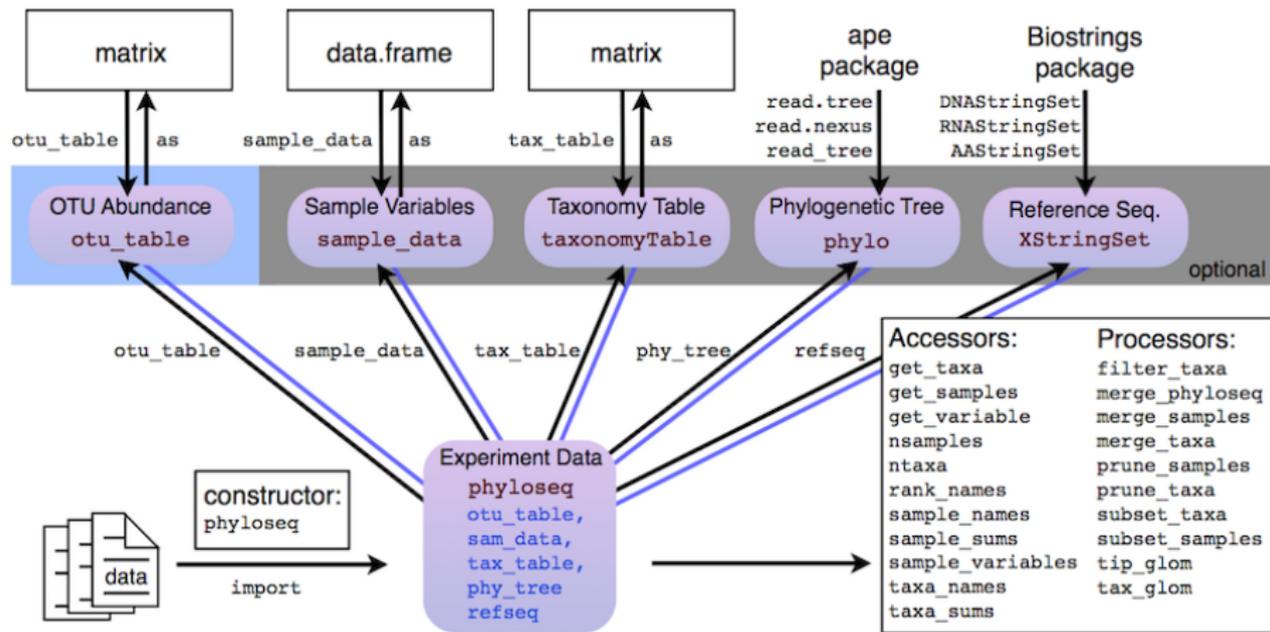
A nice data structure to store the **count table**, **taxonomic information**, **contextual data** and **phylogenetic tree** as different components of a single R object .

- Functions to **import** data from biom files, qiime output files or plain tabular files.
- **Accessors** to access different component of your dataset
- Samples and taxa names are **coherent** between the different components.
- **Filters** to keep only part of the dataset.
- **Smoothers** to aggregate parts of the dataset.
- **Manipulators** to rarefy and transform samples

A nice data structure to store the **count table**, **taxonomic information**, **contextual data** and **phylogenetic tree** as different components of a single R object .

- Functions to **import** data from biom files, qiime output files or plain tabular files.
- **Accessors** to access different component of your dataset
- Samples and taxa names are **coherent** between the different components.
- **Filters** to keep only part of the dataset.
- **Smoothers** to aggregate parts of the dataset.
- **Manipulators** to rarefy and transform samples

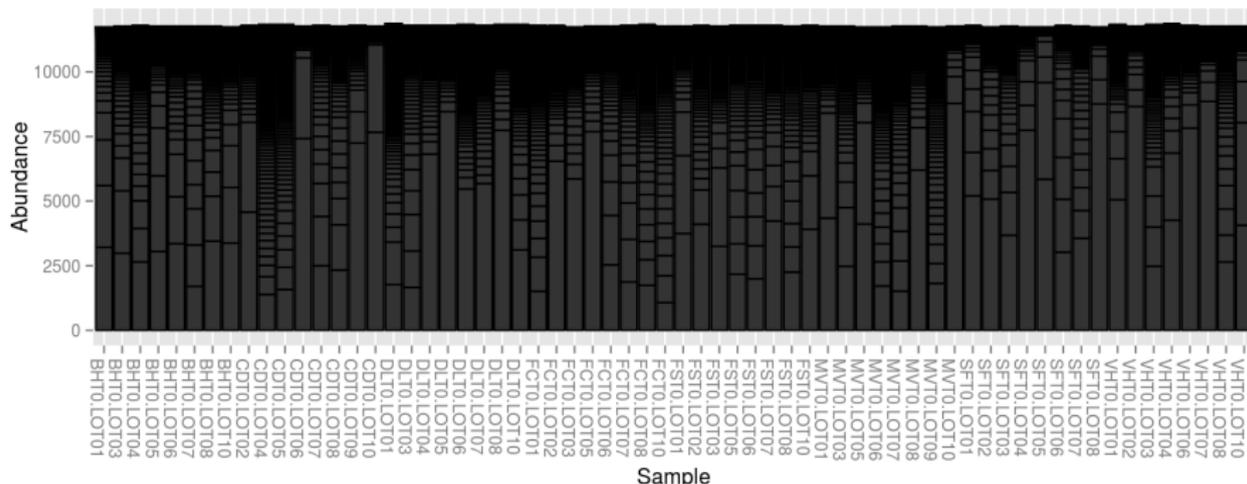
phyloseq recap (II)



- 1 Goals of the tutorial
- 2 phyloseq
- 3 **Biodiversity indices**
 - Exploring the samples composition
 - Notions of biodiversity
 - α -diversity
 - β -diversity
- 4 Exploring the structure
- 5 Diversity Partitioning
- 6 To Go Further

Looking at your samples (`plot_bar`)

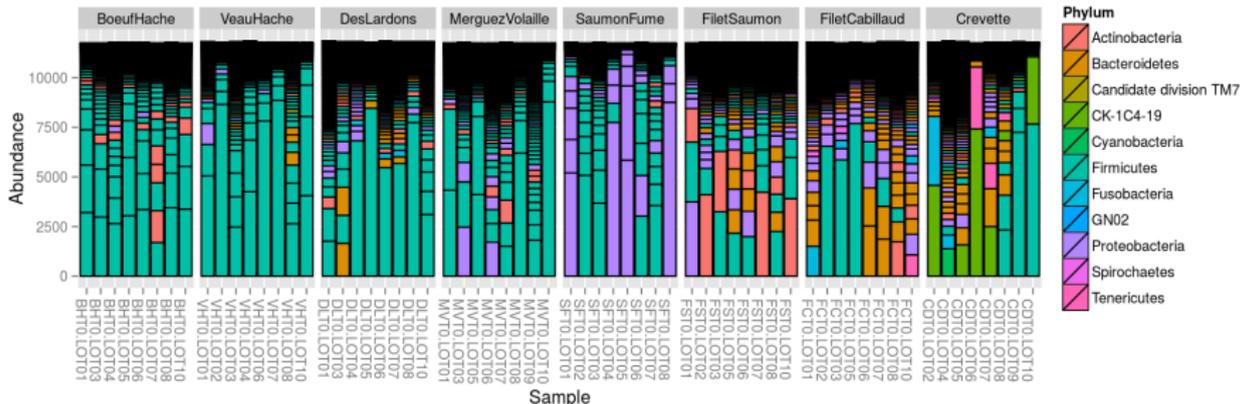
```
p <- plot_bar(food)
plot(p) ## Base graphic, ugly
```



Looking at your samples (`plot_bar`)

Organize samples and color otu by Phylum

```
p <- plot_bar(food, fill = "Phylum") ## aes, fill bar according to phylum  
p <- p + facet_wrap(~EnvType, scales = "free_x", nrow = 1) ## add facets  
plot(p)
```



Limitations of `plot_bar`

`plot_bar`

- `plot_bar` works at the *OTU*-level...
- ...which may lead to graph **cluttering** and useless legends
- No easy way to look at a **subset** of the data
- Works with absolute counts (beware of unequal depths)

Custom function `plot_composition`

- subset otus at a given taxonomic level
- aggregate otus at another taxonomic level
- Show only a given number of otus.
- Works with relative abundances

Limitations of `plot_bar`

`plot_bar`

- `plot_bar` works at the *OTU*-level...
- ...which may lead to graph **cluttering** and useless legends
- No easy way to look at a **subset** of the data
- Works with absolute counts (beware of unequal depths)

Custom function `plot_composition`

- **subset** otus at a given taxonomic level
- **aggregate** otus at another taxonomic level
- Show only a **given number** of otus.
- Works with relative abundances

Looking at your samples (`plot_composition`) (I)

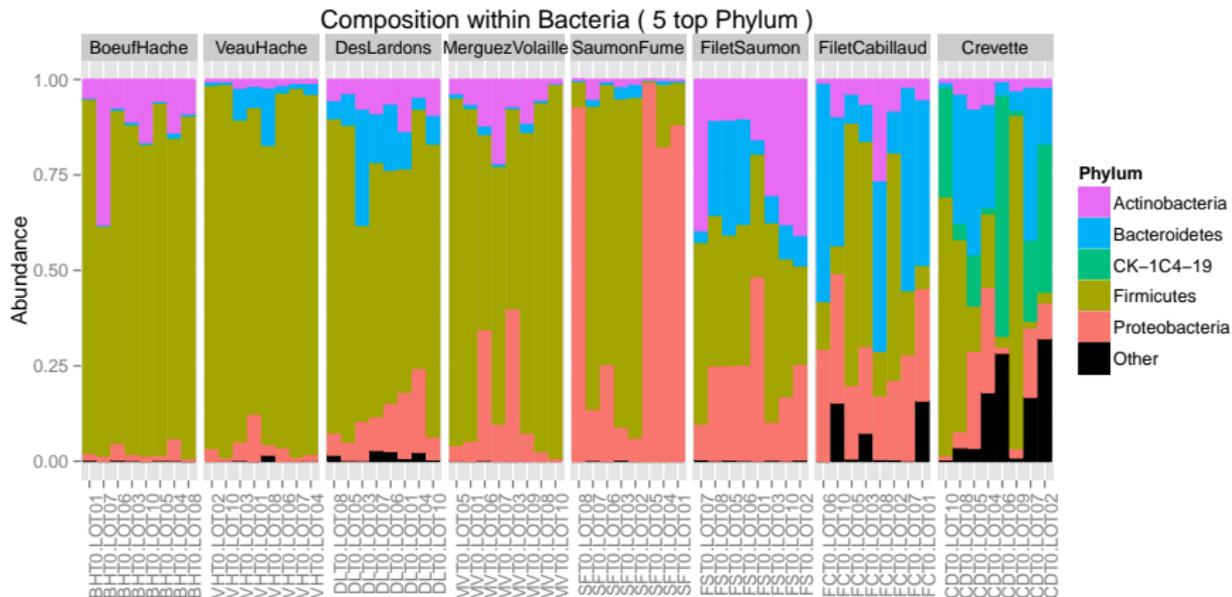
Select **Bacteria** (at **Kingdom** level) and aggregate by **Phylum**.

```
p <- plot_composition(food, "Kingdom", "Bacteria", "Phylum",
                      numberOfTaxa = 5, fill = "Phylum")
p <- p + facet_wrap(~EnvType, scales = "free_x", nrow = 1)
plot(p)
```

Looking at your samples (`plot_composition`) (I)

Select **Bacteria** (at **Kingdom** level) and aggregate by **Phylum**.

```
p <- plot_composition(food, "Kingdom", "Bacteria", "Phylum",  
                      numberOfTaxa = 5, fill = "Phylum")  
p <- p + facet_wrap(~EnvType, scales = "free_x", nrow = 1)  
plot(p)
```



Looking at your samples (`plot_composition`) (II)

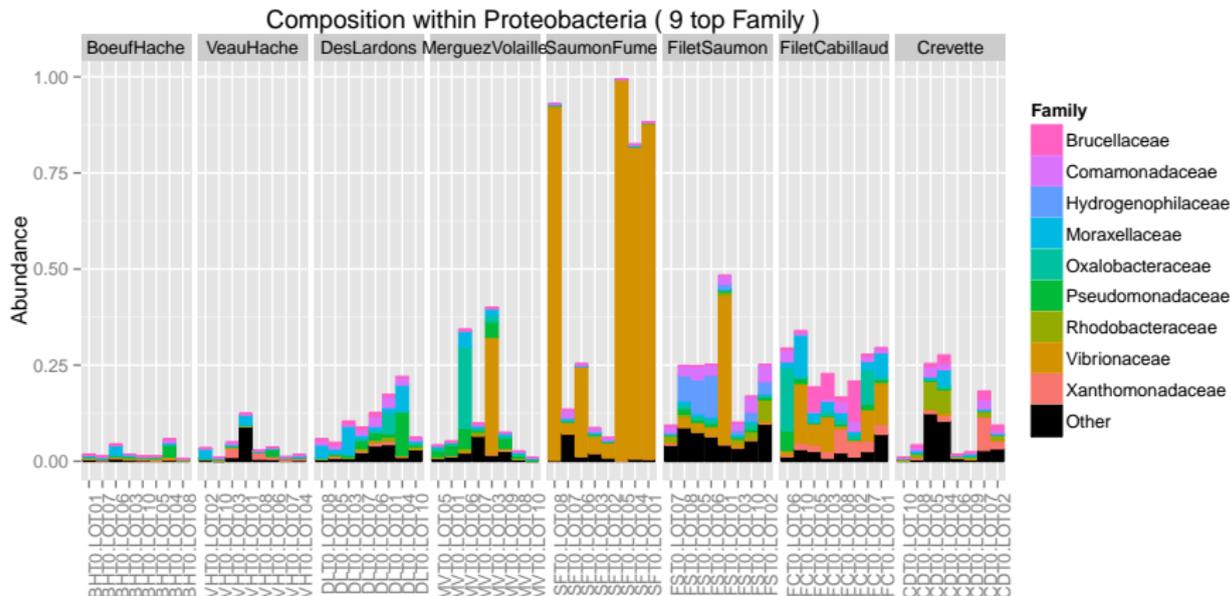
Select `Proteobacteria` (at `Phylum` level) and aggregate by `Family`.

```
p <- plot_composition(food, "Phylum", "Proteobacteria", "Family",
                      numberOfTaxa = 9, fill = "Family")
p <- p + facet_wrap(~EnvType, scales = "free_x", nrow = 1)
plot(p)
```

Looking at your samples (`plot_composition`) (II)

Select **Proteobacteria** (at **Phylum** level) and aggregate by **Family**.

```
p <- plot_composition(food, "Phylum", "Proteobacteria", "Family",  
                      numberOfTaxa = 9, fill = "Family")  
p <- p + facet_wrap(~EnvType, scales = "free_x", nrow = 1)  
plot(p)
```



- 1 Goals of the tutorial
- 2 phyloseq
- 3 Biodiversity indices**
 - Exploring the samples composition
 - Notions of biodiversity**
 - α -diversity
 - β -diversity
- 4 Exploring the structure
- 5 Diversity Partitioning
- 6 To Go Further

Different kinds of biodiversity indices...

16S surveys used to monitor the **bacterial biodiversity**.

Three flavors of diversity

- α -diversity: diversity within a community;
- β -diversity: diversity between communities;
- γ -diversity: diversity at the landscape scale (blurry for bacterial communities);

Diversity decomposition

$$\gamma = \alpha + |\times \beta$$

β -dissimilarities/distances

- Dissimilarities between pairs of communities
- Often used as a first step to compute β -diversity

Different kinds of biodiversity indices...

16S surveys used to monitor the bacterial biodiversity.

Three flavors of diversity

- α -diversity: diversity **within** a community;
- β -diversity: diversity **between** communities;
- γ -diversity: diversity at the **landscape** scale (blurry for bacterial communities);

Diversity decomposition

$$\gamma = \alpha + |\times| \beta$$

β -dissimilarities/distances

- Dissimilarities between pairs of communities
- Often used as a first step to compute β -diversity

Different kinds of biodiversity indices...

16S surveys used to monitor the bacterial biodiversity.

Three flavors of diversity

- α -diversity: diversity within a community;
- β -diversity: diversity between communities;
- γ -diversity: diversity at the landscape scale (blurry for bacterial communities);

Diversity decomposition

$$\gamma = \alpha + |\times \beta$$

β -dissimilarities/distances

- Dissimilarities between pairs of communities
- Often used as a first step to compute β -diversity

Different kinds of biodiversity indices...

16S surveys used to monitor the bacterial biodiversity.

Three flavors of diversity

- α -diversity: diversity within a community;
- β -diversity: diversity between communities;
- γ -diversity: diversity at the landscape scale (blurry for bacterial communities);

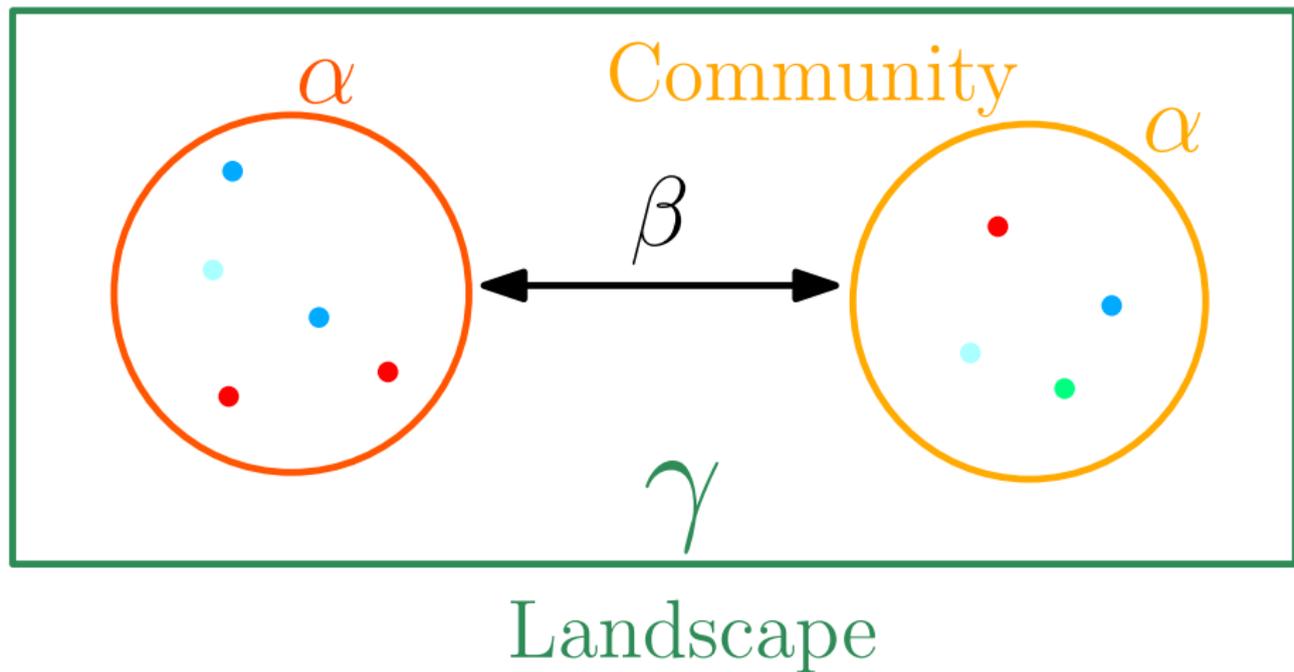
Diversity decomposition

$$\gamma = \alpha + |\times \beta$$

β -dissimilarities/distances

- Dissimilarities between **pairs** of communities
- Often used as a first step to compute β -diversity

A schematic view of diversity



Based on different types of data

Presence/Absence (qualitative) vs. Abundance (quantitative)

- Presence/Absence gives less weight to **dominant** species;
- is more **sensitive** to differences in sampling depths;
- emphasizes difference in taxa diversity rather than differences in composition.

Compositional vs. Phylogenetic

- Compositional does not require a phylogenetic tree;
- is more sensitive to erroneous otu picking;
- gives the same importance to all otus.

Based on different types of data

Presence/Absence (qualitative) vs. Abundance (quantitative)

- Presence/Absence gives less weight to dominant species;
- is more sensitive to differences in sampling depths;
- emphasizes difference in taxa diversity rather than differences in composition.

Compositional vs. Phylogenetic

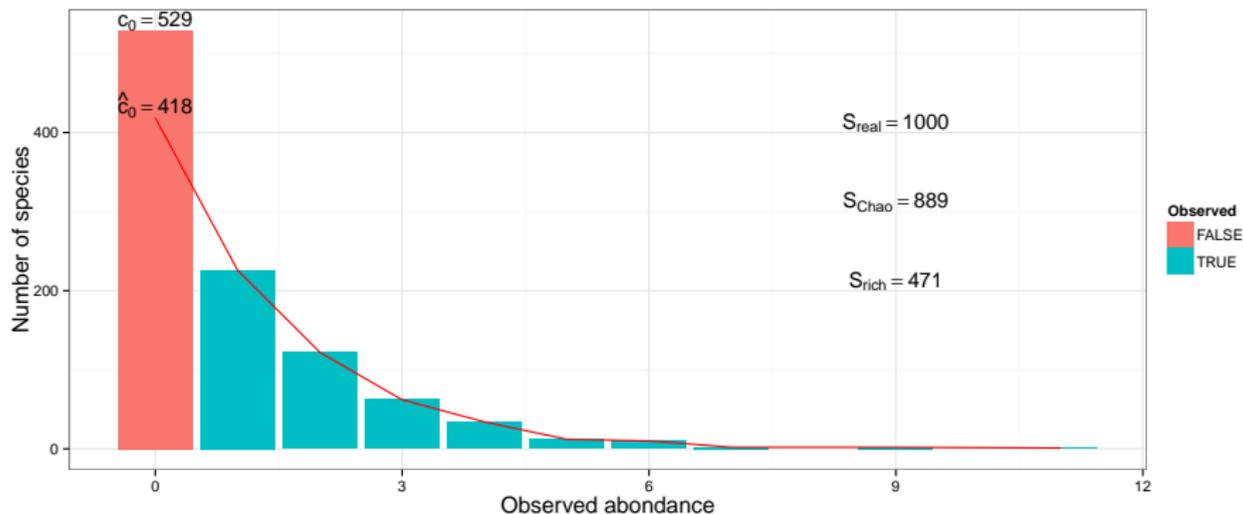
- Compositional does not require a **phylogenetic tree**;
- is more **sensitive** to erroneous otu picking;
- gives the **same importance** to all otus.

- 1 Goals of the tutorial
- 2 phyloseq
- 3 **Biodiversity indices**
 - Exploring the samples composition
 - Notions of biodiversity
 - **α -diversity**
 - β -diversity
- 4 Exploring the structure
- 5 Diversity Partitioning
- 6 To Go Further

α -diversity: number of species (richness)

Note c_i the number of species observed i times ($i = 1, 2, \dots$) and p_s the proportion of species s ($s = 1, \dots, S$)

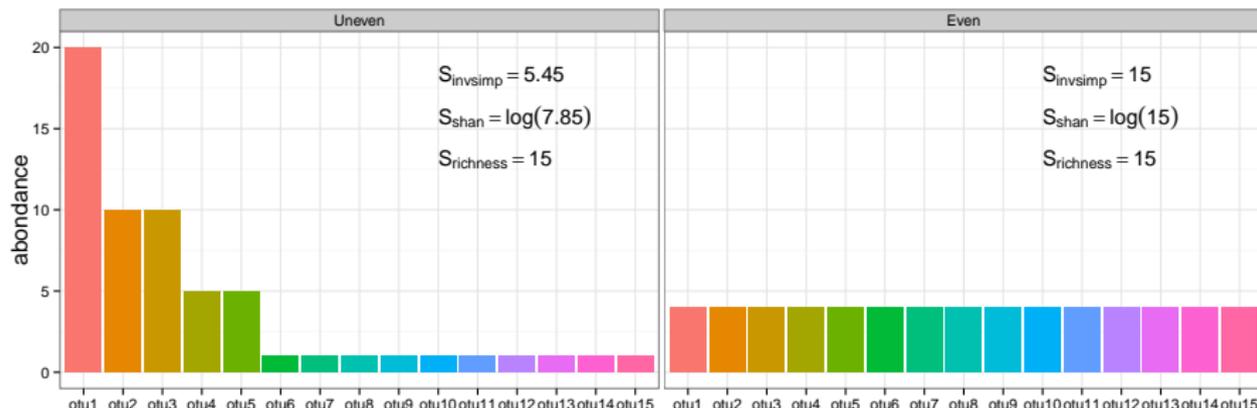
Richness	Chao1
Number of observed species	Richness + (estimated) number of unobserved species
$S_{\text{rich}} = \sum_s 1_{\{p_s > 0\}} = \sum_i c_i$	$S_{\text{Chao}} = S_{\text{rich}} + \hat{c}_0$



α -diversity: evenness of the species distribution

Give more weight to abundant species

Shannon	Inv-Simpson
Evenness of the species abundance distribution	Inverse probability that two sequences sampled at random come from the same species
$S_{\text{Shan}} = - \sum_s p_s \log(p_s) \leq \log(S)$	$S_{\text{Inv-Simp}} = \frac{1}{p_1^2 + \dots + p_S^2} \leq S$



Available in phyloseq

- **Species richness:** number of observed otus
- **Shannon entropy/Jensen:** the *width* of the otu relative abundance distribution. Roughly, it reflects our (in)ability to predict the otu of a randomly picked bacteria.
- **Simpson:** 1 - probability that two bacteria picked at random in the community belong to different otu.
- **Inverse Simpson:** inverse of the probability that two bacteria picked at random belong to the same otu.
- **Chao1:** number of observed otu + estimate of the number of unobserved otus

Available in phyloseq

- **Species richness:** number of observed otus
- **Shannon entropy/Jensen:** the *width* of the otu relative abundance distribution. Roughly, it reflects our (in)ability to predict the otu of a randomly picked bacteria.
- **Simpson:** 1 - probability that two bacteria picked at random in the community belong to different otu.
- **Inverse Simpson:** inverse of the probability that two bacteria picked at random belong to the same otu.
- **Chao1:** number of observed otu + estimate of the number of unobserved otus

Available in phyloseq

- **Species richness:** number of observed otus
- **Shannon entropy/Jensen:** the *width* of the otu relative abundance distribution. Roughly, it reflects our (in)ability to predict the otu of a randomly picked bacteria.
- **Simpson:** 1 - probability that two bacteria picked at random in the community belong to different otu.
- **Inverse Simpson:** inverse of the probability that two bacteria picked at random belong to the same otu.
- **Chao1:** number of observed otu + estimate of the number of unobserved otus

Available in phyloseq

- **Species richness:** number of observed otus
- **Shannon entropy/Jensen:** the *width* of the otu relative abundance distribution. Roughly, it reflects our (in)ability to predict the otu of a randomly picked bacteria.
- **Simpson:** 1 - probability that two bacteria picked at random in the community belong to different otu.
- **Inverse Simpson:** inverse of the probability that two bacteria picked at random belong to the same otu.
- **Chao1:** number of observed otu + estimate of the number of unobserved otus

Available in phyloseq

- **Species richness:** number of observed otus
- **Shannon entropy/Jensen:** the *width* of the otu relative abundance distribution. Roughly, it reflects our (in)ability to predict the otu of a randomly picked bacteria.
- **Simpson:** 1 - probability that two bacteria picked at random in the community belong to different otu.
- **Inverse Simpson:** inverse of the probability that two bacteria picked at random belong to the same otu.
- **Chao1:** number of observed otu + estimate of the number of unobserved otus

α diversity and filtering (I)

Many α diversities (richness, Chao) depend **a lot** on rare otus. Do not **trim** rare otus before computing them as it can **drastically** alter the result (see next slide).

Richness

Richness are plotted with `plot_richness`. Note the `x = "EnvType"` passed on to the `aes` mapping of a `ggplot`.

```
p <- plot_richness(food, color = "EnvType", x = "EnvType",
                  measures = c("Observed", "Chao1", "Shannon",
                              "Simpson", "InvSimpson"))
p <- p + geom_boxplot()
plot(p)
```

α diversity and filtering (I)

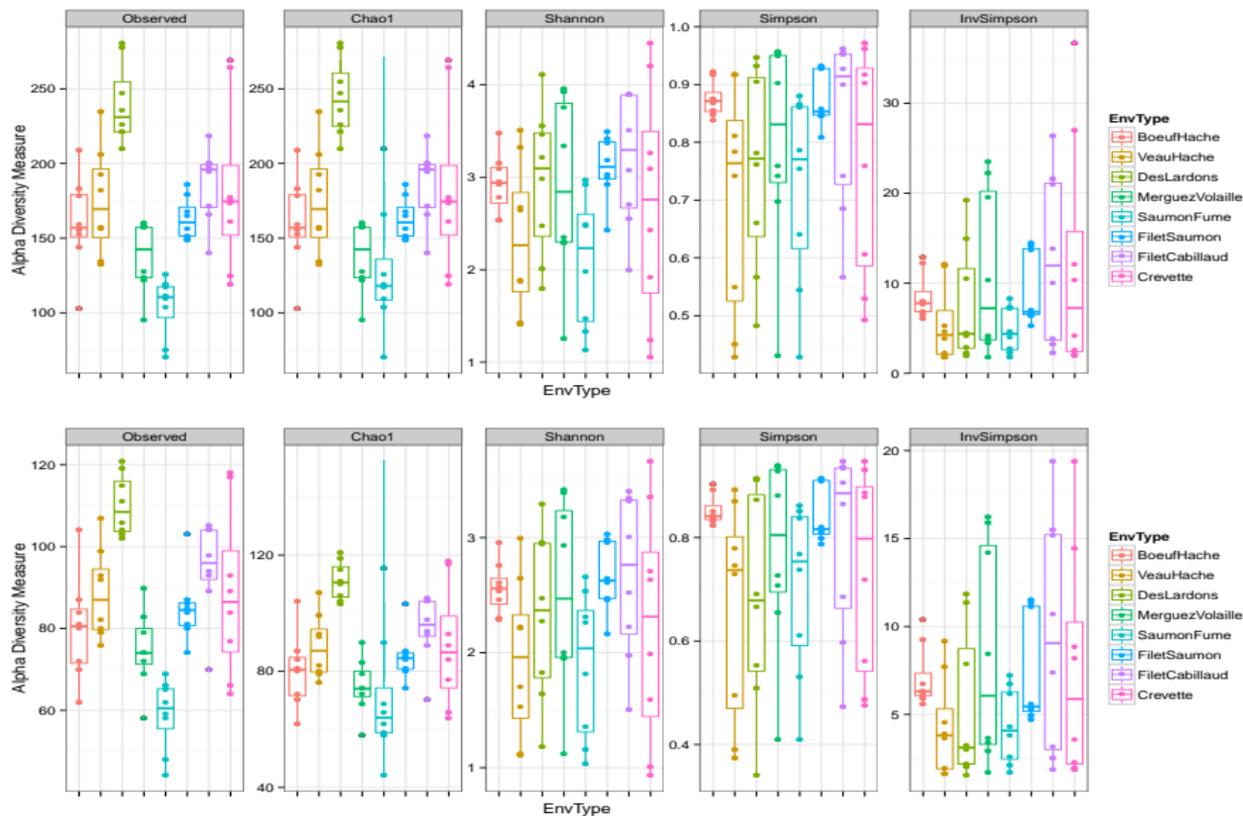
Many α diversities (richness, Chao) depend **a lot** on rare otus. Do not **trim** rare otus before computing them as it can **drastically** alter the result (see next slide).

Richness

Richness are plotted with `plot_richness`. Note the `x = "EnvType"` passed on to the `aes` mapping of a `ggplot`.

```
p <- plot_richness(food, color = "EnvType", x = "EnvType",
                  measures = c("Observed", "Chao1", "Shannon",
                              "Simpson", "InvSimpson"))
p <- p + geom_boxplot()
plot(p)
```

α diversity: without (top) and with (bottom) trimming



α diversity: numeric values

Numeric values of α -diversities are given by `estimate_richness` (used internally by `plot_richness`)

```
alpha.diversity <- estimate_richness(food,  
                                   measures = c("Observed", "Chao1", "Shannon"))  
head(alpha.diversity)
```

##	Observed	Chao1	se.chao1	Shannon
## DLTO.LOT08	210	210.0000	0.0000	2.016038
## DLTO.LOT05	221	254.7857	13.3895	1.798009
## DLTO.LOT03	226	226.0000	0.0000	3.455284
## DLTO.LOT07	221	221.0000	0.0000	2.982161
## DLTO.LOT06	278	278.0000	0.0000	3.209521
## DLTO.LOT01	281	281.0000	0.0000	4.106852

```
write.table(alpha.diversity, "myfile.txt")
```

α diversity: A quick ANOVA

```
data <- cbind(sample_data(food), alpha.diversity)
food.anova <- aov(Observed ~ EnvType, data)
summary(food.anova) ## significant effect of environment type on richness
```

##		Df	Sum Sq	Mean Sq	F value	Pr(>F)			
##	EnvType	7	86922	12417	12.49	1.63e-09 ***			
##	Residuals	56	55686	994					
##	---								
##	Signif. codes:	0	'***'	0.001	'**'	0.01	'*' 0.05	'.' 0.1	' ' 1

```
food.anova <- aov(Shannon ~ EnvType, data)
summary(food.anova) ## effect on Shannon diversity is not significant
```

##		Df	Sum Sq	Mean Sq	F value	Pr(>F)
##	EnvType	7	7.98	1.139	1.767	0.112
##	Residuals	56	36.12	0.645		

α diversity: A quick ANOVA

```
data <- cbind(sample_data(food), alpha.diversity)
food.anova <- aov(Observed ~ EnvType, data)
summary(food.anova) ## significant effect of environment type on richness
```

##		Df	Sum Sq	Mean Sq	F value	Pr(>F)			
##	EnvType	7	86922	12417	12.49	1.63e-09 ***			
##	Residuals	56	55686	994					
##	---								
##	Signif. codes:	0	'***'	0.001	'**'	0.01	'*' 0.05	'.' 0.1	' ' 1

```
food.anova <- aov(Shannon ~ EnvType, data)
summary(food.anova) ## effect on Shannon diversity is not significant
```

##		Df	Sum Sq	Mean Sq	F value	Pr(>F)
##	EnvType	7	7.98	1.139	1.767	0.112
##	Residuals	56	36.12	0.645		

Interpretation

- Many taxa observed in **Deslardons** (high Chao1, high Observed)...
- ...but low Shannon and Inverse-Simpson
- ⇒ communities dominated by a few abundant taxa

Interpretation

- Environments differ a lot in terms of richness...
- ...but not so much in terms of Shannon diversity
- ⇒ *Effective* diversities are quite similar

Interpretation

- Many taxa observed in **Deslardons** (high Chao1, high Observed)...
- ...but low Shannon and Inverse-Simpson
- \Rightarrow communities dominated by a few abundant taxa

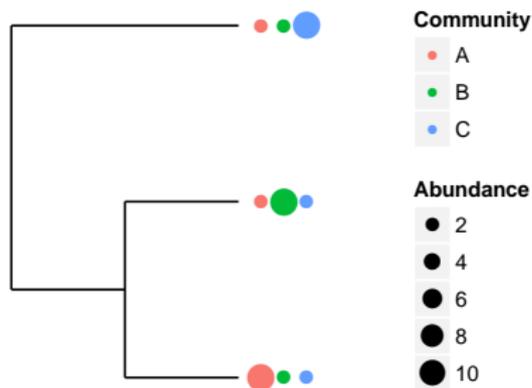
Interpretation

- Environments differ a lot in terms of richness...
- ...but not so much in terms of Shannon diversity
- \Rightarrow *Effective* diversities are quite similar

- 1 Goals of the tutorial
- 2 phyloseq
- 3 Biodiversity indices**
 - Exploring the samples composition
 - Notions of biodiversity
 - α -diversity
 - β -diversity**
- 4 Exploring the structure
- 5 Diversity Partitioning
- 6 To Go Further

β dissimilarities

- Many β diversities (both compositional and phylogenetic) offered by phyloseq through the **generic** distance function.
- Different dissimilarities capture different **features** of the communities.



β -diversity: compositional

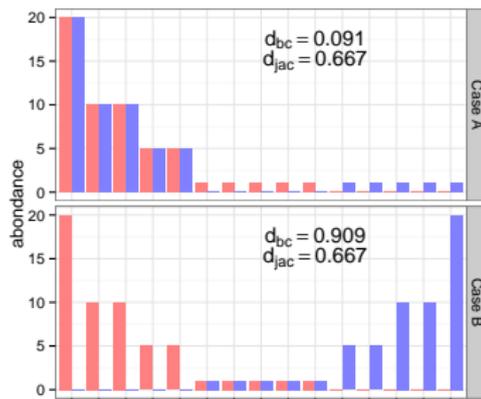
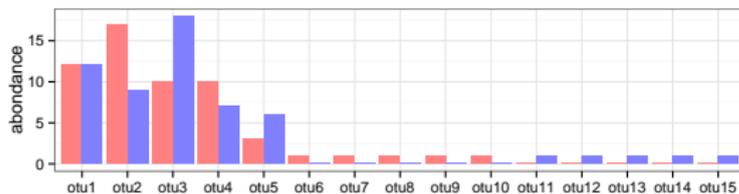
Note n_s^1 the count of species s ($s = 1, \dots, S$) in **community 1** and n_s^2 the count in **community 2**. We focus on **shared** features.

Jaccard	Bray-Curtis
Fraction of species specific to either 1 or 2	Fraction of the community specific to 1 or to 2
$d_{\text{Jac}} = \frac{\sum_s 1_{\{n_s^1 > 0, n_s^2 = 0\}} + 1_{\{n_s^2 > 0, n_s^1 = 0\}}}{\sum_s 1_{\{n_s^1 + n_s^2 > 0\}}}$	$d_{\text{BC}} = \sum_s n_s^1 - n_s^2 / \sum_s n_s^1 + n_s^2 $

β -diversity: compositional

Note n_s^1 the count of species s ($s = 1, \dots, S$) in **community 1** and n_s^2 the count in **community 2**. We focus on **shared** features.

Jaccard	Bray-Curtis
Fraction of species specific to either 1 or 2	Fraction of the community specific to 1 or to 2
$d_{\text{Jac}} = \frac{\sum_s 1_{\{n_s^1 > 0, n_s^2 = 0\}} + 1_{\{n_s^2 > 0, n_s^1 = 0\}}}{\sum_s 1_{\{n_s^1 + n_s^2 > 0\}}}$	$d_{\text{BC}} = \frac{\sum_s n_s^1 - n_s^2 }{\sum_s n_s^1 + n_s^2 }$



β -diversity: phylogenetic

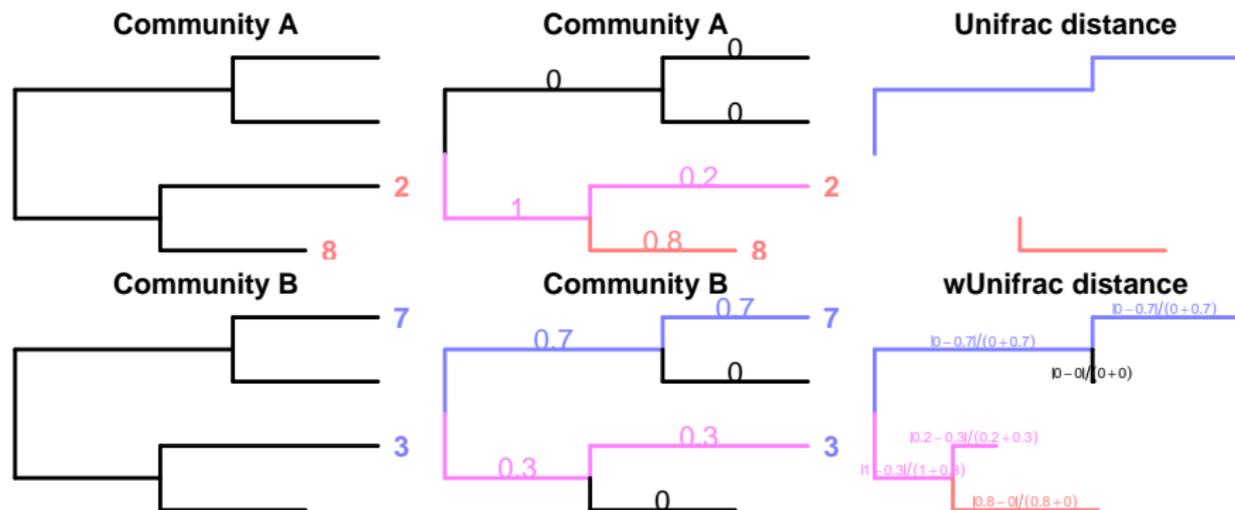
For each branch e , note l_e its length and p_e (resp. q_e) the fraction of **community 1** (resp. **community 2**) below branch e . We focus on **shared** features.

Unifrac	Weighted Unifrac
Fraction of the tree specific to either 1 or 2	Fraction of the diversity specific to 1 or to 2
$d_{\text{UF}} = \frac{\sum_e l_e [1_{\{p_e > 0, q_e = 0\}} + 1_{\{q_e > 0, p_e = 0\}}]}{\sum_e l_e \times 1_{\{p_e + q_e > 0\}}}$	$d_{\text{UF}} = \frac{\sum_e l_e p_e - q_e }{\sum_e l_e (p_e + q_e)}$

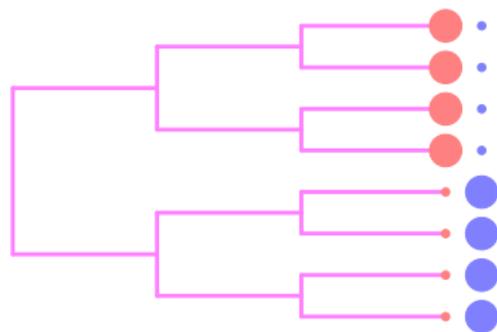
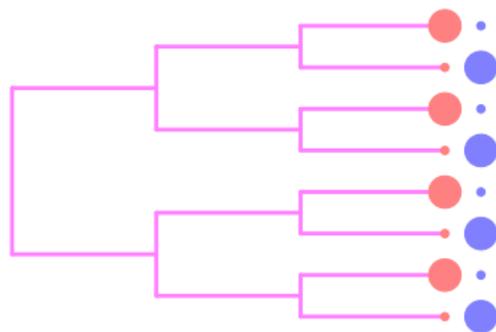
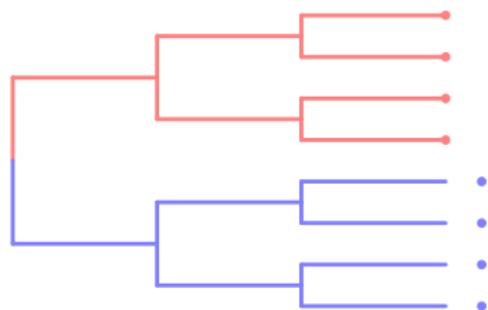
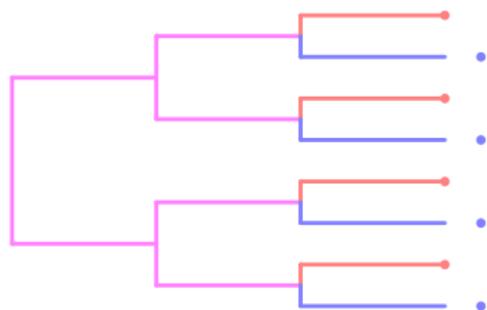
β -diversity: phylogenetic

For each branch e , note l_e its length and p_e (resp. q_e) the fraction of **community 1** (resp. **community 2**) below branch e . We focus on **shared** features.

Unifrac	Weighted Unifrac
Fraction of the tree specific to either 1 or 2	Fraction of the diversity specific to 1 or to 2
$d_{UF} = \frac{\sum_e l_e [1_{\{p_e > 0, q_e = 0\}} + 1_{\{q_e > 0, p_e = 0\}}]}{\sum_e l_e \times 1_{\{p_e + q_e > 0\}}}$	$d_{UF} = \frac{\sum_e l_e p_e - q_e }{\sum_e l_e (p_e + q_e)}$

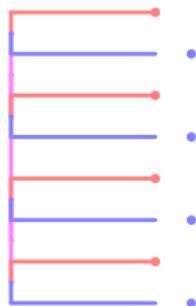


Differences between the β -dissimilarities

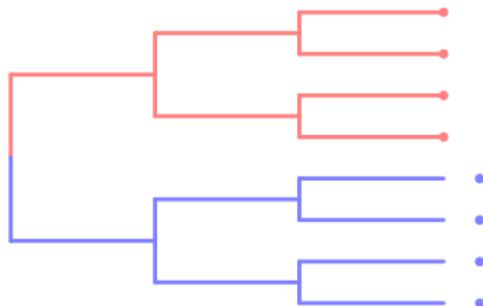


Differences between the β -dissimilarities

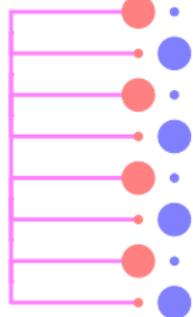
low UF, high Jac



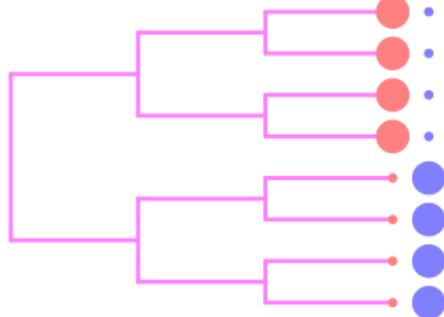
high UF, high Jac



low wUF, high BC



high wUF, high BC



β -dissimilarities/distances in phyloseq

β dissimilarities are computed with distance

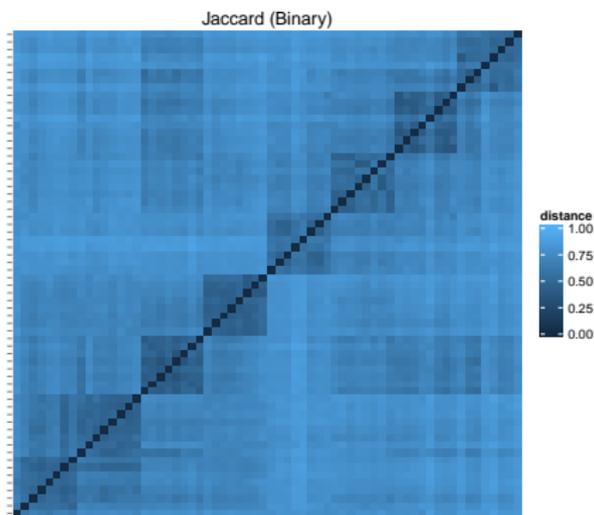
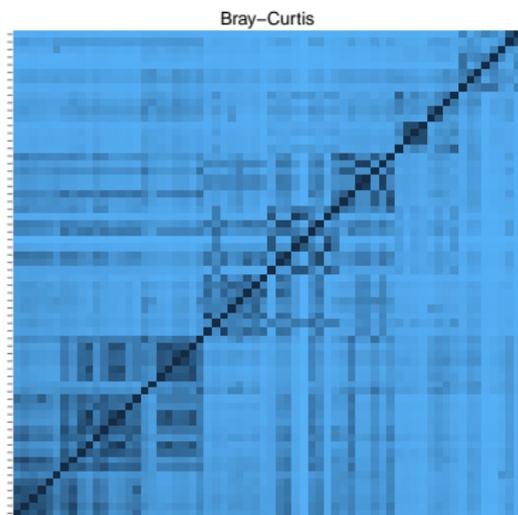
```
dist.bc <- distance(food, method = "bray") ## Bray-Curtis
```

All available distances are available with

```
distanceMethodList
```

```
## $UniFrac
## [1] "unifrac" "wunifrac"
##
## $DPCoA
## [1] "dpcoa"
##
## $JSD
## [1] "jsd"
##
## $vegdist
## [1] "manhattan" "euclidean" "canberra" "bray" "kulczynski"
## [6] "jaccard" "gower" "altGower" "morisita" "horn"
## [11] "mountford" "raup" "binomial" "chao" "cao"
##
## $betadiver
## [1] "w" "-1" "c" "wb" "r" "I" "e" "t" "me" "j" "sor"
```

β -dissimilarities/distances in phyloseq (II)

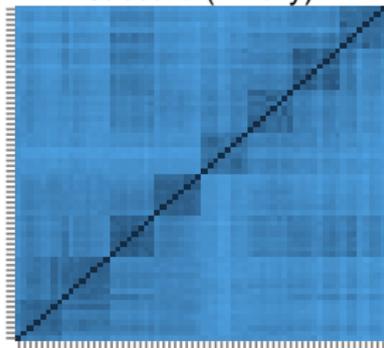


Phylogenetic β -dissimilarities/distances in phyloseq (II)

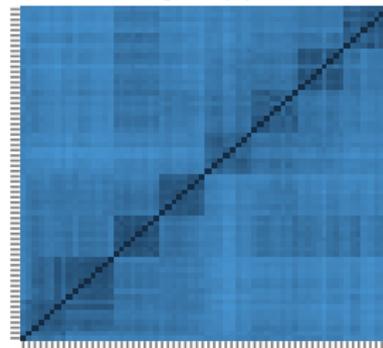
```
dist.uf <- distance(food, method = "unifrac") ## Unifrac  
dist.wuf <- distance(food, method = "wunifrac") ## Weighted Unifrac
```

Compositional vs Qualitative

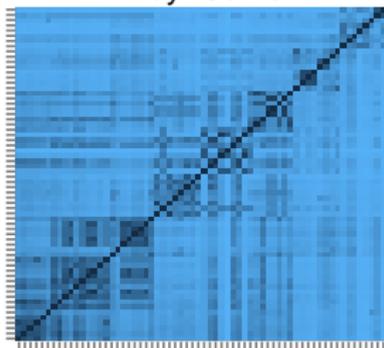
Jaccard (Binary)



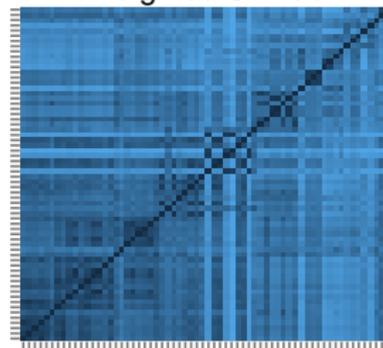
Unifrac



Bray-Curtis



Weighted Unifrac



Compositional vs Qualitative (II)

- Jaccard lower than Bray-Curtis \Rightarrow abundant taxa are not shared
- Jaccard higher than Unifrac \Rightarrow communities' taxa are distinct but phylogenetically related
- Unifrac higher than weighted Unifrac \Rightarrow abundant taxa in both communities are phylogenetically close.

General remarks about β diversity

In general, **qualitative** diversities are most sensitive to factors that affect presence/absence of organisms (such as pH, salinity, depth, etc) and therefore useful to study and define **bioregions** (regions with little or no flow between them)...

... whereas **quantitative** distances focus on factors that affect **relative** changes (seasonal changes, nutrient availability, concentration of oxygen, depth, etc) and therefore useful to monitor communities **over time** or **along an environmental gradient**.

Different distances capture different features of the samples. There is no "one size fits all"

General remarks about β diversity

In general, **qualitative** diversities are most sensitive to factors that affect presence/absence of organisms (such as pH, salinity, depth, etc) and therefore useful to study and define **bioregions** (regions with little or no flow between them)...

... whereas **quantitative** distances focus on factors that affect **relative** changes (seasonal changes, nutrient availability, concentration of oxygen, depth, etc) and therefore useful to monitor communities **over time** or **along an environmental gradient**.

Different distances capture different features of the samples. There is no "one size fits all"

- 1 Goals of the tutorial
- 2 phyloseq
- 3 Biodiversity indices
- 4 Exploring the structure**
 - Ordination
 - Clustering
 - Heatmap
- 5 Diversity Partitioning
- 6 To Go Further

Principal Component Analysis (PCA)

- Each community is described by **otus abundances**
- Otus abundance maybe **correlated**
- PCA finds **linear combinations** of otus that
 - are uncorrelated
 - capture well the variance of community composition

But variance is not a very good measure of β -diversity.

Principal Component Analysis (PCA)

- Each community is described by otus abundances
- Otus abundance maybe correlated
- PCA finds linear combinations of otus that
 - are uncorrelated
 - capture well the variance of community composition

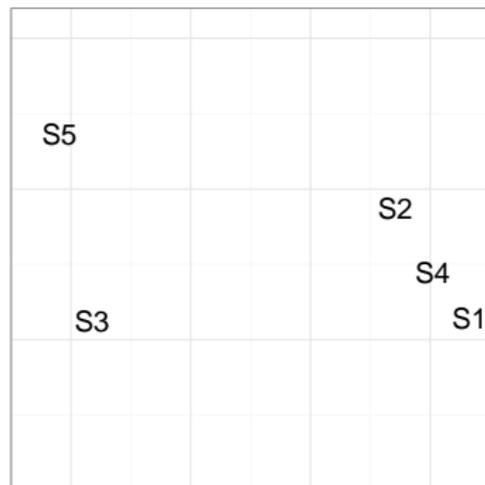
But **variance** is not a very good measure of β -diversity.

MultiDimensional Scaling (MDS/PCoA)

MDS/PCoA

- Start from a distance matrix $D = (d_{ij})$
- Project the communities $\text{Com}_i \mapsto X_i$ in a euclidian space such that distances are preserved $\|X_i - X_j\| \simeq d_{ij}$

	S1	S2	S3	S4	S5
S1	0.00	2.21	6.31	0.99	7.50
S2	2.21	0.00	5.40	1.22	5.74
S3	6.31	5.40	0.00	5.75	3.16
S4	0.99	1.22	5.75	0.00	6.64
S5	7.50	5.74	3.16	6.64	0.00



MultiDimensional Scaling (MDS/PCoA)

MDS/PCoA

- Start from a distance matrix $D = (d_{ij})$
- Project the communities $\text{Com}_i \mapsto X_i$ in a euclidian space such that distances are preserved $\|X_i - X_j\| \simeq d_{ij}$

	S1	S2	S3	S4	S5
S1	0.00	2.21	6.31	0.99	7.50
S2	2.21	0.00	5.40	1.22	5.74
S3	6.31	5.40	0.00	5.75	3.16
S4	0.99	1.22	5.75	0.00	6.64
S5	7.50	5.74	3.16	6.64	0.00

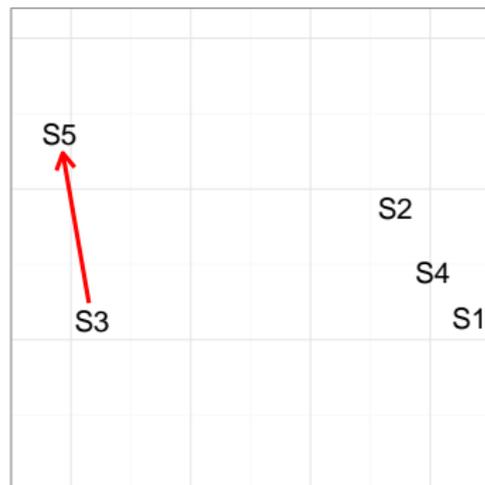


MultiDimensional Scaling (MDS/PCoA)

MDS/PCoA

- Start from a distance matrix $D = (d_{ij})$
- Project the communities $\text{Com}_i \mapsto X_i$ in a euclidian space such that distances are preserved $\|X_i - X_j\| \simeq d_{ij}$

	S1	S2	S3	S4	S5
S1	0.00	2.21	6.31	0.99	7.50
S2	2.21	0.00	5.40	1.22	5.74
S3	6.31	5.40	0.00	5.75	3.16
S4	0.99	1.22	5.75	0.00	6.64
S5	7.50	5.74	3.16	6.64	0.00



Ordination in phyloseq : `ordinate`

Ordination is done through the `ordinate` function:

Ordination

You can pass the distance either by name (and phyloseq will call `distance`)...

```
ord <- ordinate(food, method = "MDS", distance = "bray")
```

or by passing a distance matrix directly (useful if you already computed it)

```
dist.bc <- distance(food, method = "bray")  
ord <- ordinate(food, method = "MDS", distance = dist.bc)
```

The graphic is then produced with `plot_ordination`

```
p <- plot_ordination(food, ord, color = "EnvType")  
p <- p + theme_bw() + ggtitle("MDS + BC") ## add title and plain background  
plot(p)
```

Ordination in phyloseq : `ordinate`

Ordination is done through the `ordinate` function:

Ordination

You can pass the distance either by name (and phyloseq will call `distance`)...

```
ord <- ordinate(food, method = "MDS", distance = "bray")
```

or by passing a distance matrix directly (useful if you already computed it)

```
dist.bc <- distance(food, method = "bray")  
ord <- ordinate(food, method = "MDS", distance = dist.bc)
```

The graphic is then produced with `plot_ordination`

```
p <- plot_ordination(food, ord, color = "EnvType")  
p <- p + theme_bw() + ggtitle("MDS + BC") ## add title and plain background  
plot(p)
```

Ordination in phyloseq : `ordinate`

Ordination is done through the `ordinate` function:

Ordination

You can pass the distance either by name (and phyloseq will call `distance`)...

```
ord <- ordinate(food, method = "MDS", distance = "bray")
```

or by passing a distance matrix directly (useful if you already computed it)

```
dist.bc <- distance(food, method = "bray")  
ord <- ordinate(food, method = "MDS", distance = dist.bc)
```

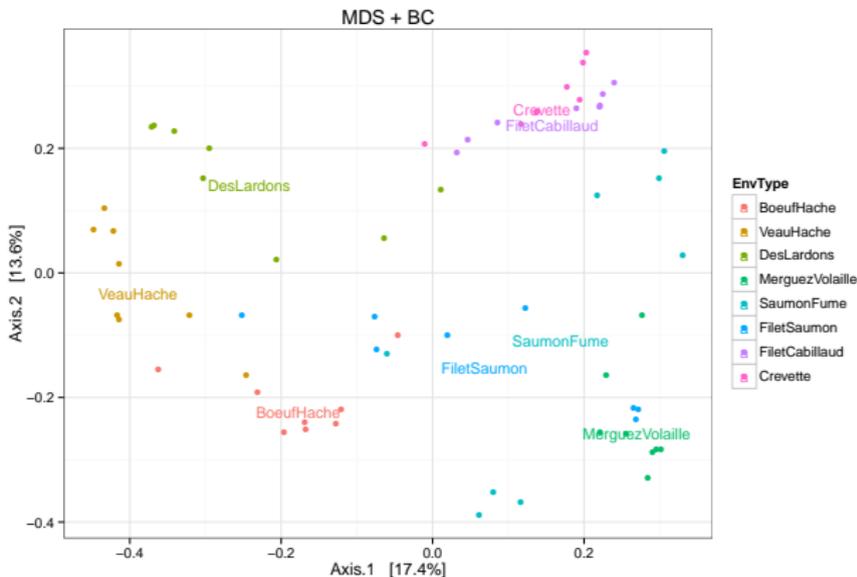
The graphic is then produced with `plot_ordination`

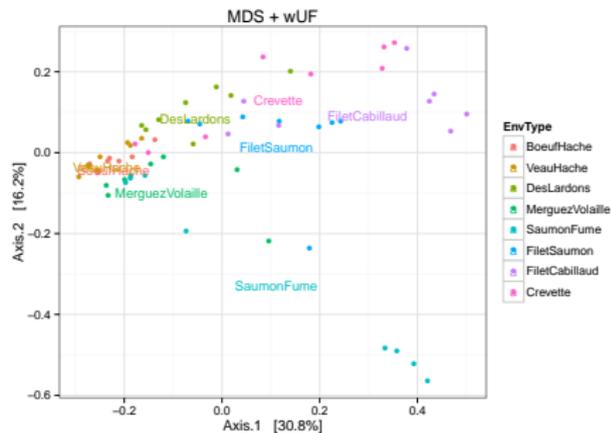
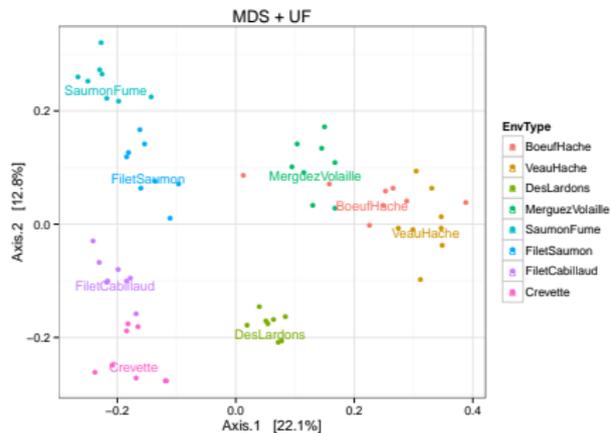
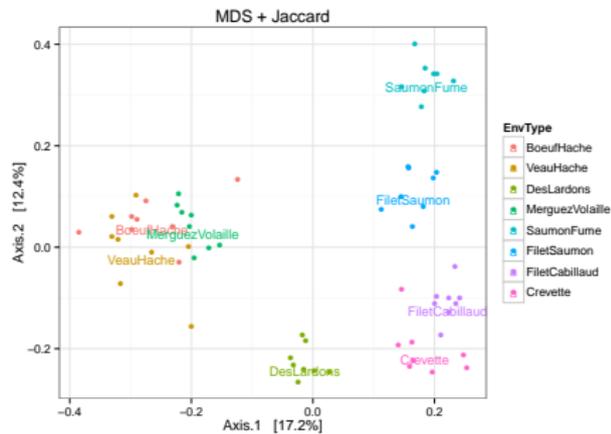
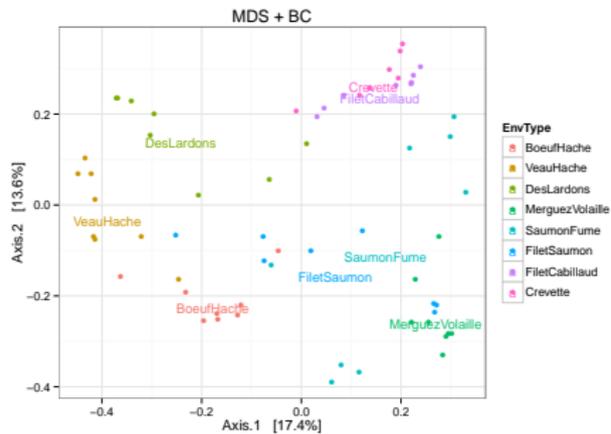
```
p <- plot_ordination(food, ord, color = "EnvType")  
p <- p + theme_bw() + ggtitle("MDS + BC") ## add title and plain background  
plot(p)
```


Ordination in phyloseq : `plot_samples`

Custom `plot_samples` function built around `plot_ordination` to represent groups (extra `replicate` parameter)

```
p <- plot_samples(food, ord, color = "EnvType", replicate = "EnvType")  
p <- p + theme_bw() + ggtitle("MDS + BC")  
plot(p)
```





Interpretation

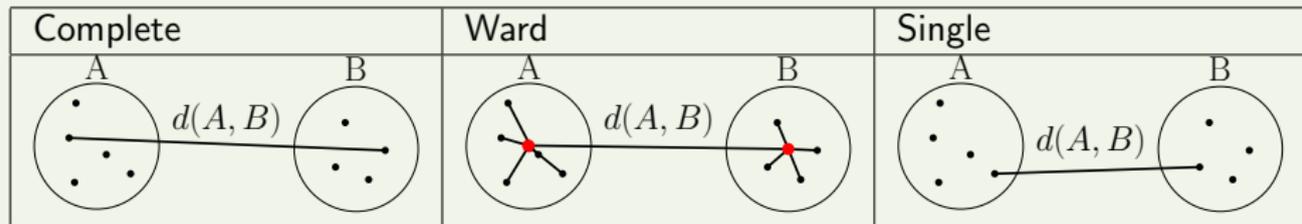
- Qualitative distances (Unifrac, Jaccard) separate meat products from seafood ones \Rightarrow detected taxa segregate by origin
- DesLardons is somewhere in between \Rightarrow contamination induced by sea salt.
- Quantitative distances (wUnifrac) exhibit a gradient meat - seafood (on axis 1) with DesLardons in the middle and a gradient SaumonFume - everything else on axis 2.
- Large overlap between groups in terms of relative composition but less so in term of species composition (a side effect of undersampling?)
- Note the difference between wUniFrac and Bray-Curtis for the distances between BoeufHache and VeauHache
- **Warning** The 2-D representation captures only **part of the original distances**.

Outline

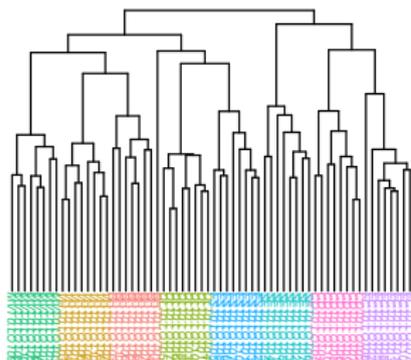
- 1 Goals of the tutorial
- 2 phyloseq
- 3 Biodiversity indices
- 4 Exploring the structure**
 - Ordination
 - Clustering**
 - Heatmap
- 5 Diversity Partitioning
- 6 To Go Further

Hierarchical Clustering

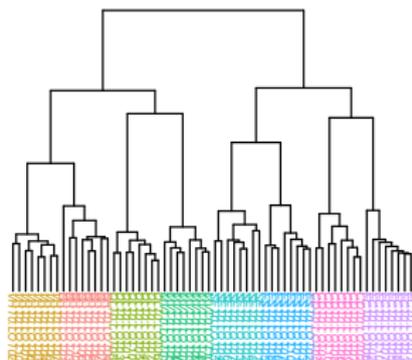
- Merge **closest** communities (according to some distance)
- Update distances between **sets** of communities using **linkage function**
- Repeat until all communities have been merged



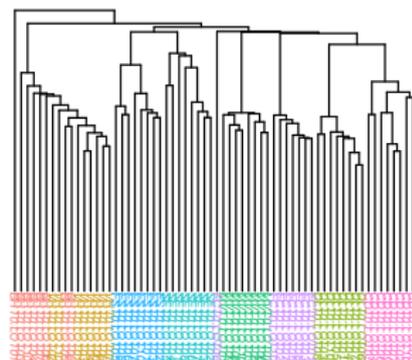
complete linkage



ward.D2 linkage



single linkage



Clustering with hclust

- Choose a **distance** (among Jaccard, Bray-Curtis, Unifrac, etc)
- Choose a **linkage function**

Feed to hclust and plot

```
clustering <- hclust(distance.matrix, method = "linkage.function")  
plot(clustering)
```

linkage function

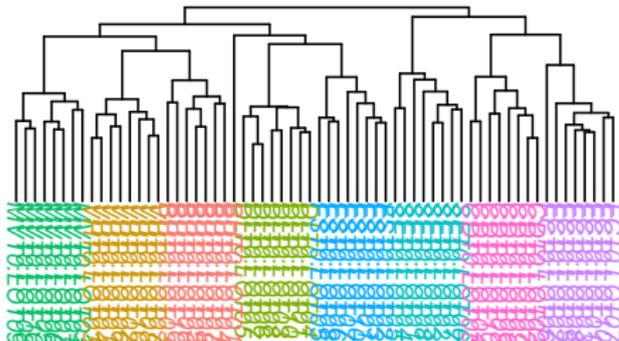
- **complete** (complete): tends to produce **compact**, spherical clusters and guarantees that all samples in a cluster are similar to each other.
- **Ward** (ward.D2): tends to also produces **spherical** clusters but has better theoretical properties than complete linkage.
- **single** (single): friend of friend approach, tends to produce **banana-shaped** or chains-like clusters.

Better dendrograms

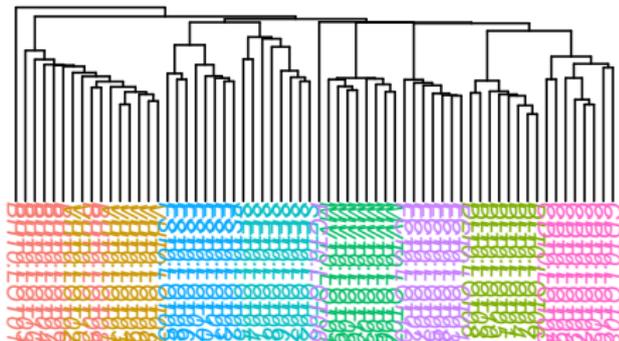
With some effort (see companion R script), we can produce better dendrograms and color sample by food type (appreciate what ggplot does for you behind the hook).

```
## Env types
envtype <- get_variable(food, "EnvType")
## automatic color palette: one color per different sample type
palette <- hue_pal()(length(levels(envtype)))
## Map sample type to color
tipColor = col_factor(palette, levels = levels(envtype))(envtype)
## Change hclust object to phylo object and plot
par(mar = c(0, 0, 2, 0))
clust.uf <- as.phylo(hclust(dist.uf, method = "complete"))
plot(clust.uf, tip.color = tipColor, direction = "downwards",
     main = paste(method, "linkage"))
```

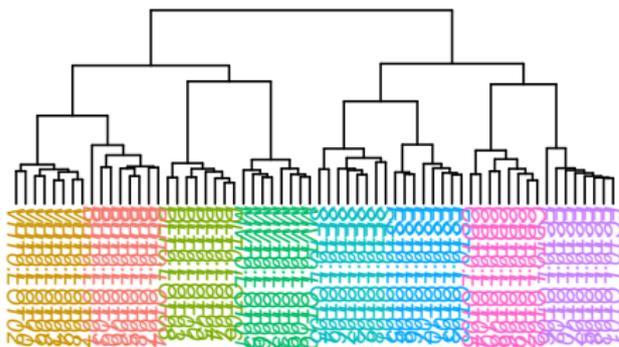
complete linkage



single linkage



ward.D2 linkage



- Crevette
- FiletCabillaud
- FiletSaumon
- SaumonFume
- MerguezVolaille
- DesLardons
- VeauHache
- BoeufHache

- Consistent with the ordination plots, clustering works quite well for the UniFrac distance for some linkage (Ward)
- Clustering is based on the **whole** distance whereas ordination represents **parts** of the distance (the most it can with 2 dimensions)

Outline

- 1 Goals of the tutorial
- 2 phyloseq
- 3 Biodiversity indices
- 4 Exploring the structure**
 - Ordination
 - Clustering
 - Heatmap**
- 5 Diversity Partitioning
- 6 To Go Further

Heatmap with `plot_heatmap`

`plot_heatmap` is a versatile function to visualize the count table.

- Finds a **meaningful order** of the samples and the otus
- Allows the user to choose a **custom** order
- Allows the user to change the color scale
- Produces a `gplot2` object, easy to manipulate and customize

```
p <- plot_heatmap(food, low = "yellow", high = "red", na.value = "white",
                 sample.order = mySampleOrder, taxa.order = myTaxaOrder)
## add faceting
p <- p + facet_grid(~EnvType, scales = "free_x")
plot(p)
```

Heatmap with `plot_heatmap`

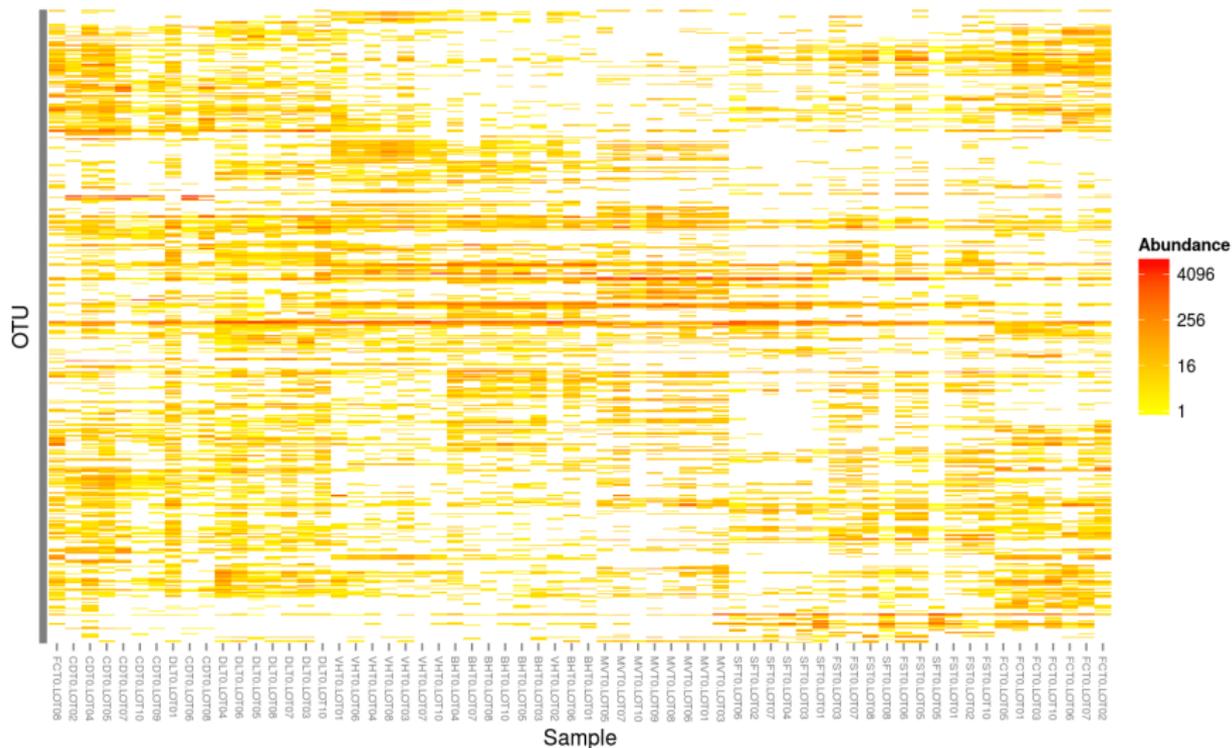
`plot_heatmap` is a versatile function to visualize the count table.

- Finds a **meaningful order** of the samples and the otus
- Allows the user to choose a **custom** order
- Allows the user to change the color scale
- Produces a `ggplot2` object, easy to manipulate and customize

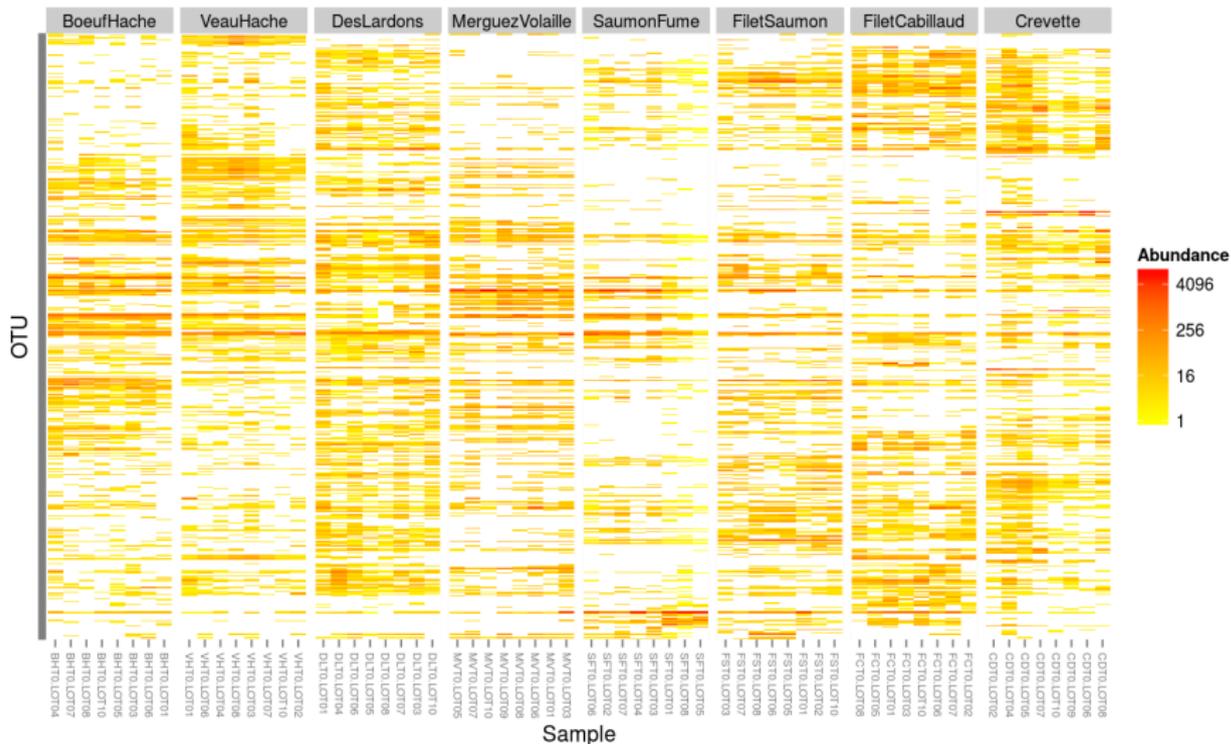
```
p <- plot_heatmap(food, low = "yellow", high = "red", na.value = "white",
                 sample.order = mySampleOrder, taxa.order = myTaxaOrder)
## add facetting
p <- p + facet_grid(~EnvType, scales = "free_x")
plot(p)
```



```
plot_heatmap(food, low = "yellow", high = "red", na.value = "white")
```



```
plot_heatmap(food, low = "yellow", high = "red", na.value = "white") +
  facet_grid(~EnvType, scales = "free_x")
```



```

plot_heatmap(food) +
  scale_fill_gradient2(low = "#1a9850", mid = "#ffffbf", high = "#d73027"
    na.value = "white", trans = log_trans(4),
    midpoint = log(100, base = 4)) +
  facet_grid(~EnvType, scales = "free_x")

```

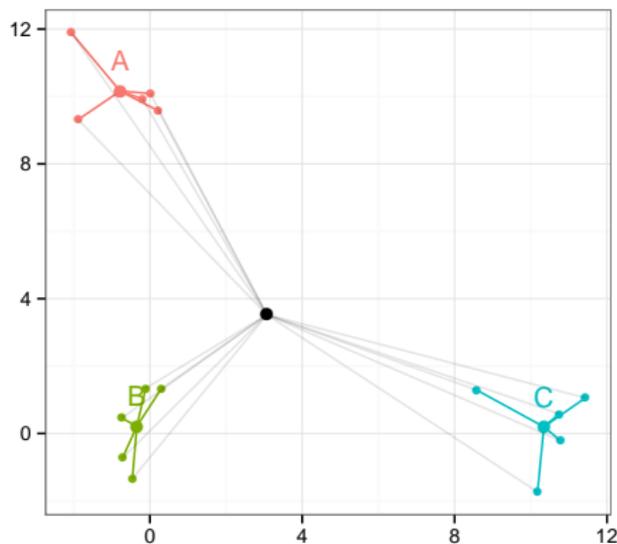
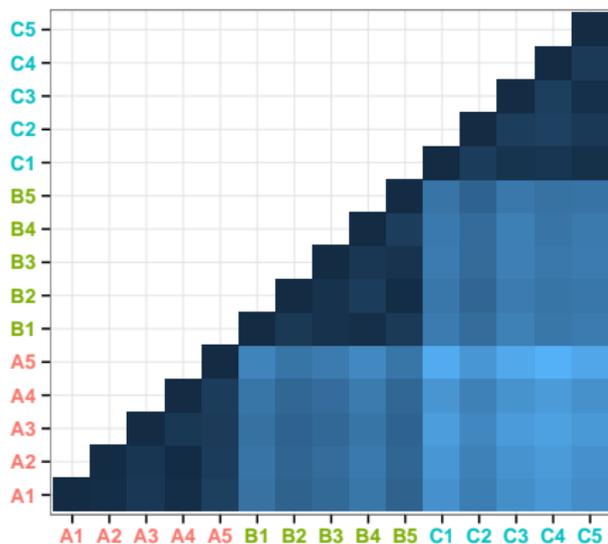


- **Block-like** structure of the abundance table
- **Interaction** between (groups of) taxa and (groups of) samples
- **Core** and **condition-specific** microbiota
- \Rightarrow Classification of taxa and use of custom taxa order to highlight structure

- 1 Goals of the tutorial
- 2 phyloseq
- 3 Biodiversity indices
- 4 Exploring the structure
- 5 Diversity Partitioning
 - Multivariate Analysis
 - Constrained Analysis of Principal Coordinates (CAP)
 - Permutational Multivariate ANOVA
- 6 To Go Further

Idea

- Test **composition differences** of communities from **different groups** using a **distance matrix**
- Compare **within group** to **between group** distances



- 1 Goals of the tutorial
- 2 phyloseq
- 3 Biodiversity indices
- 4 Exploring the structure
- 5 Diversity Partitioning**
 - Multivariate Analysis
 - Constrained Analysis of Principal Coordinates (CAP)**
 - Permutational Multivariate ANOVA
- 6 To Go Further

Constrained Analysis of Principal Coordinates (CAP)

Idea

- Find **associations** between **community composition** and **environmental variables** (pH, group)
- Quantify differences between groups of samples

Method	Input	Steps	Axis	Variation explained
PCA	X (sample \times var.)	$X \xrightarrow{PCA} \text{Axis}$	Lin. comb. of var. (columns of X)	Variance of samples (rows of X)
RDA	X (sample \times var.) Y (sample \times otus)	$(Y, X) \xrightarrow{Proj.} \hat{Y}(X)$ $\hat{Y}(X) \xrightarrow{PCA} \text{Axis}$	Lin. comb. of var. (columns of X)	Variance of projected samples (rows of $\hat{Y}(X)$)
CAP	X (sample \times var.) D (samp. \times samp.)	$D \xrightarrow{PCoA/MDS} Y$ $(Y, X) \xrightarrow{Proj.} \hat{Y}(X)$ $\hat{Y}(X) \xrightarrow{PCA} \text{Axis}$	Lin. comb. of var. (columns of X)	Distance between samples

Constrained Analysis of Principal Coordinates (CAP)

Idea

- Find **associations** between **community composition** and **environmental variables** (pH, group)
- Quantify differences between groups of samples

Method	Input	Steps	Axis	Variation explained
PCA	X (sample \times var.)	$X \xrightarrow{PCA} \text{Axis}$	Lin. comb. of var. (columns of X)	Variance of samples (rows of X)
RDA	X (sample \times var.) Y (sample \times otus)	$(Y, X) \xrightarrow{Proj.} \hat{Y}(X)$ $\hat{Y}(X) \xrightarrow{PCA} \text{Axis}$	Lin. comb. of var. (columns of X)	Variance of projected samples (rows of $\hat{Y}(X)$)
CAP	X (sample \times var.) D (samp. \times samp.)	$D \xrightarrow{PCoA/MDS} Y$ $(Y, X) \xrightarrow{Proj.} \hat{Y}(X)$ $\hat{Y}(X) \xrightarrow{PCA} \text{Axis}$	Lin. comb. of var. (columns of X)	Distance between samples

CAP with `capscale` (I)

Regress a **distance matrix** against some **covariates** using the standard R syntax for linear models.

```
metadata <- as(sample_data(food), "data.frame") ## convert sample_data to data.frame
cap <- capscale(dist.uf ~ EnvType,
                data = metadata)
```

CAP with `capscale` (II)

Sample type explains roughly 63% of the total variation between samples
(as measured by Unifrac)

```
cap
## Call: capscale(formula = dist.uf ~ EnvType, data = metadata)
##
##           Inertia Proportion Rank
## Total          12.1280
## Real Total     12.1600      1.0000
## Constrained    7.6570      0.6297      7
## Unconstrained  4.5030      0.3703     56
## Imaginary     -0.0320
## Inertia is squared Unknown distance
##
## Eigenvalues for constrained axes:
##   CAP1  CAP2  CAP3  CAP4  CAP5  CAP6  CAP7
## 2.5546 1.4630 1.1087 0.8954 0.7159 0.4940 0.4255
##
## Eigenvalues for unconstrained axes:
##   MDS1  MDS2  MDS3  MDS4  MDS5  MDS6  MDS7  MDS8
## 0.4161 0.2908 0.2540 0.2111 0.2066 0.2011 0.1675 0.1562
## (Showed only 8 of all 56 unconstrained eigenvalues)
```

CAP with `capscale` (III)

```
cap <- capscale(dist.uf ~ EnvType, data = metadata)
anova <- anova(cap, permutations = 999)

## Permutation test for capscale under reduced model
## Permutation: free
## Number of permutations: 999
##
## Model: capscale(formula = dist.uf ~ EnvType, data = metadata)
##           Df Variance      F Pr(>F)
## Model      7   7.6571 13.603 0.001 ***
## Residual 56   4.5032
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Assumptions

- Community composition responds **linearly** to environmental changes
- Permutation test can accommodate complex designs

Caveats

- Inadequate for non-linear responses
- Permutation should preserve the design (nestedness)

Assumptions

- Community composition responds **linearly** to environmental changes
- Permutation test can accommodate complex designs

Caveats

- Inadequate for **non-linear responses**
- Permutation should **preserve** the design (nestedness)

- 1 Goals of the tutorial
- 2 phyloseq
- 3 Biodiversity indices
- 4 Exploring the structure
- 5 Diversity Partitioning**
 - Multivariate Analysis
 - Constrained Analysis of Principal Coordinates (CAP)
 - **Permutational Multivariate ANOVA**
- 6 To Go Further

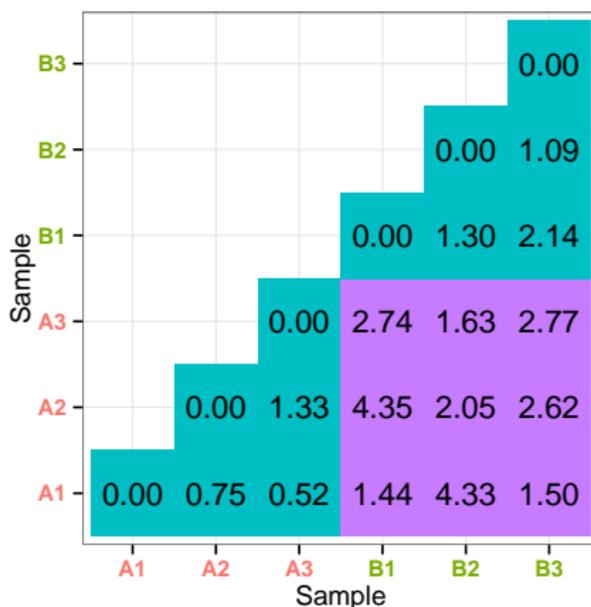
Idea

Test **differences** in the community composition of communities from **different groups** using a **distance matrix**.

Multivariate ANOVA

Idea

Test **differences** in the community composition of communities from **different groups** using a **distance matrix**.



Multivariate ANOVA with `adonis`

Sample type explains again roughly 63% of the total variation.

```
metadata <- as(sample_data(food), "data.frame")
adonis(dist.uf ~ EnvType, data = metadata, perm = 9999)

##
## Call:
## adonis(formula = dist.uf ~ EnvType, data = metadata, permutations = 9999)
##
## Permutation: free
## Number of permutations: 9999
##
## Terms added sequentially (first to last)
##
##           Df SumsOfSqs MeanSqs F.Model      R2 Pr(>F)
## EnvType    7     7.6565  1.09379  13.699 0.63132 1e-04 ***
## Residuals 56     4.4713  0.07984           0.36868
## Total     63    12.1278           1.00000
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Assumptions behind Multivariate ANOVA

Assumptions

- PERMANOVA tests **location** effect (\simeq mean)
- PERMANOVA assumes equal **dispersions** (\simeq variance)

Limitations

- If groups have **different** dispersions, p -value are not adequate.
- (Not a problem if differences in dispersion matter as much as differences in location)
- p -values computed using permutations, permutations must **respect the design**.

Assumptions behind Multivariate ANOVA

Assumptions

- PERMANOVA tests **location** effect (\simeq mean)
- PERMANOVA assumes equal **dispersions** (\simeq variance)

Limitations

- If groups have **different** dispersions, p -value are not adequate.
- (Not a problem if differences in dispersion matter as much as differences in location)
- p -values computed using permutations, permutations must **respect the design**.

Chaillou, S., Chaulot-Talmon, A., Caekebeke, H., Cardinal, M., Christeans, S., Denis, C., Desmonts, M. H., Dousset, X., Feurer, C., Hamon, E., Joffraud, J.-J., La Carbona, S., Leroi, F., Leroy, S., Lorre, S., Macé, S., Pilet, M.-F., Prévost, H., Rivollier, M., Roux, D., Talon, R., Zagorec, M., and Champomier-Vergès, M.-C. (2015). Origin and ecological selection of core and food-specific bacterial communities associated with meat and seafood spoilage. *ISME J*, 9(5):1105–1118.

McMurdie, P. J. and Holmes, S. (2013). phyloseq: An r package for reproducible interactive analysis and graphics of microbiome census data. *PLoS ONE*, 8(4):e61217.

Shade, A., Jones, S. E., Caporaso, J. G., Handelsman, J., Knight, R., Fierer, N., and Gilbert, J. A. (2014). Conditionally rare taxa disproportionately contribute to temporal changes in microbial diversity. *MBio*, 5(4):e01371–e01314.

- 1 Goals of the tutorial
- 2 phyloseq
- 3 Biodiversity indices
- 4 Exploring the structure
- 5 Diversity Partitioning
- 6 **To Go Further**
 - Advanced Filters
 - Advanced Smoothing
 - Advanced Import
 - Rarefaction curves

Filter with `genefilter_sample` (I)

Filter

- Specify a **sample wise, otu wide condition** (e.g. abundance greater than 2, in the top ten otus, etc)
- Specify a number of samples A
- Select only otus satisfying **condition** in at least A samples (usually to prune them).

Examples (what do they do?)

```
● condition <- function(x) { x > 0 }  
  taxaToKeep <- genefilter_sample(food, condition, 5)  
  prune_taxa(taxaToKeep, food)
```

```
● condition <- function(x) { order(x, decreasing = TRUE) <= 250 }  
  taxaToKeep <- genefilter_sample(food, condition, 3)  
  prune_taxa(taxaToKeep, food)
```

Filter with `genefilter_sample` (I)

Filter

- Specify a sample wise, otu wide **condition** (e.g. abundance greater than 2, in the top ten otus, etc)
- Specify a **number** of samples A
- Select only otus satisfying **condition** in at least A samples (usually to prune them).

Examples (what do they do?)

```
condition <- function(x) { x > 0 }  
taxaToKeep <- genefilter_sample(food, condition, 5)  
prune_taxa(taxaToKeep, food)
```

```
condition <- function(x) { order(x, decreasing = TRUE) <= 250 }  
taxaToKeep <- genefilter_sample(food, condition, 3)  
prune_taxa(taxaToKeep, food)
```

Filter with `genefilter_sample` (I)

Filter

- Specify a sample wise, otu wide **condition** (e.g. abundance greater than 2, in the top ten otus, etc)
- Specify a number of samples A
- **Select** only otus satisfying **condition** in at least A samples (usually to **prune** them).

Examples (what do they do?)

```
• condition <- function(x) { x > 0 }  
  taxaToKeep <- genefilter_sample(food, condition, 5)  
  prune_taxa(taxaToKeep, food)
```

```
• condition <- function(x) { order(x, decreasing = TRUE) <= 250 }  
  taxaToKeep <- genefilter_sample(food, condition, 3)  
  prune_taxa(taxaToKeep, food)
```

Filter with `genefilter_sample` (I)

Filter

- Specify a sample wise, otu wide `condition` (e.g. abundance greater than 2, in the top ten otus, etc)
- Specify a number of samples A
- Select only otus satisfying `condition` in at least A samples (usually to prune them).

Examples (what do they do?)

- ```
condition <- function(x) { x > 0 }
taxaToKeep <- genefilter_sample(food, condition, 5)
prune_taxa(taxaToKeep, food)
```
- ```
condition <- function(x) { order(x, decreasing = TRUE) <= 250 }  
taxaToKeep <- genefilter_sample(food, condition, 3)  
prune_taxa(taxaToKeep, food)
```

Filter with `genefilter_sample` (II)

First example

- `condition` is TRUE if a taxa is present in a sample;
- `condition` must be met 5 times;
- Select taxa that appear in at least 5 samples.

Second example

- `condition` is TRUE if a taxa is among the 250 most abundant ones in the sample;
- `condition` must be met 3 times;
- Select taxa that are very abundant (top 250) in at least 3 samples.

Filter with `genefilter_sample` (II)

First example

- `condition` is TRUE if a taxa is present in a sample;
- `condition` must be met 5 times;
- Select taxa that appear in at least 5 samples.

Second example

- `condition` is TRUE if a taxa is among the 250 most abundant ones in the sample;
- `condition` must be met 3 times;
- Select taxa that are very abundant (top 250) in at least 3 samples.

Filter with `filter_taxa` (I)

Filter

- Specify an **otu wise, sample wide** on the otu abundance vector condition (e.g. overall abundance greater than 3, etc);
- Selects only otus satisfying condition (usually to prune them)
- Works on an otu-by-otu basis. Beware of bad normalizations.

Examples (what do they do?)

```
condition <- function(x) { sum(x > 0) >= 5 }  
taxaToKeep <- filter_taxa(food, condition)  
prune_taxa(taxaToKeep, food)
```

```
condition <- function(x) { sum(x) >= 100 }  
taxaToKeep <- filter_taxa(food, condition)  
prune_taxa(taxaToKeep, food)
```

Filter with `filter_taxa` (I)

Filter

- Specify an otu wise, sample wide on the otu abundance vector condition (e.g. overall abundance greater than 3, etc);
- **Selects** only otus satisfying condition (usually to **prune** them)
- Works on an otu-by-otu basis. Beware of bad normalizations.

Examples (what do they do?)

```
condition <- function(x) { sum(x > 0) >= 5 }  
taxaToKeep <- filter_taxa(food, condition)  
prune_taxa(taxaToKeep, food)
```

```
condition <- function(x) { sum(x) >= 100 }  
taxaToKeep <- filter_taxa(food, condition)  
prune_taxa(taxaToKeep, food)
```

Filter with `filter_taxa` (I)

Filter

- Specify an otu wise, sample wide on the otu abundance vector condition (e.g. overall abundance greater than 3, etc);
- Selects only otus satisfying condition (usually to prune them)
- Works on an **otu-by-otu** basis. Beware of bad normalizations.

Examples (what do they do?)

```
condition <- function(x) { sum(x > 0) >= 5 }  
taxaToKeep <- filter_taxa(food, condition)  
prune_taxa(taxaToKeep, food)
```

```
condition <- function(x) { sum(x) >= 100 }  
taxaToKeep <- filter_taxa(food, condition)  
prune_taxa(taxaToKeep, food)
```

Filter with `filter_taxa` (I)

Filter

- Specify an otu wise, sample wide on the otu abundance vector condition (e.g. overall abundance greater than 3, etc);
- Selects only otus satisfying condition (usually to prune them)
- Works on an otu-by-otu basis. Beware of bad normalizations.

Examples (what do they do?)

```
condition <- function(x) { sum(x > 0) >= 5 }  
taxaToKeep <- filter_taxa(food, condition)  
prune_taxa(taxaToKeep, food)
```

```
condition <- function(x) { sum(x) >= 100 }  
taxaToKeep <- filter_taxa(food, condition)  
prune_taxa(taxaToKeep, food)
```

Filter with `filter_taxa` (II)

First example

- `condition` is TRUE if a taxa has at least 5 positive counts (across samples);
- Select taxa that appear in at least 5 samples;
- Probably better done with `genefilter_sample`

Second example

- `condition` is TRUE if a taxa has global abundance at least 100;
- Select taxa with overall abundance at least 100 (beware of unequal sample sizes).
- Probably done better with `sample_sums`

Filter with `filter_taxa` (II)

First example

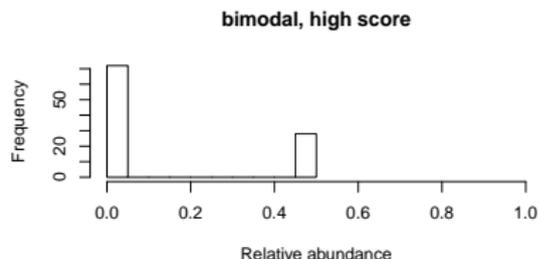
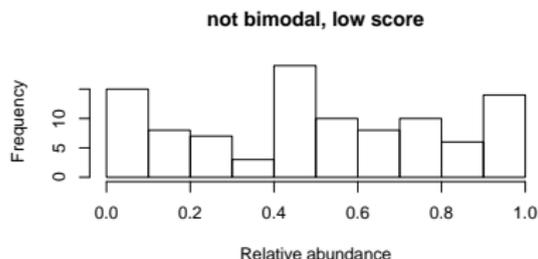
- `condition` is TRUE if a taxa has at least 5 positive counts (across samples);
- Select taxa that appear in at least 5 samples;
- Probably better done with `genefilter_sample`

Second example

- `condition` is TRUE if a taxa has global abundance at least 100;
- Select taxa with overall abundance at least 100 (beware of unequal sample sizes).
- Probably done better with `sample_sums`

A more interesting example

```
library(moments) ## for skewness and kurtosis
## bimodality score, between 0 (not bimodal) and 1 (completely bimodal)
score <- function(x) { (1 + skewness(x)) / (kurtosis(x) + 3) }
condition <- function(x) { score(x) >= 0.9 }
taxaToKeep <- filter_taxa(food, condition)
prune_taxa(taxaToKeep, food)
```



A more interesting example

```
library(moments) ## for skewness and kurtosis
## bimodality score, between 0 (not bimodal) and 1 (completely bimodal)
score <- function(x) { (1 + skewness(x)) / (kurtosis(x) + 3) }
condition <- function(x) { score(x) >= 0.9 }
taxaToKeep <- filter_taxa(food, condition)
prune_taxa(taxaToKeep, food)
```

Conditionally Rare Taxa (CRT) Shade et al. (2014)

- score quantifies the bimodality of a taxa abundance distribution
- condition is TRUE for taxa with score higher than 0.9;
- Selects taxa with bimodal abundance distribution: Conditionally Rare Taxa (CRT) of Shade et al. (2014)

- 1 Goals of the tutorial
- 2 phyloseq
- 3 Biodiversity indices
- 4 Exploring the structure
- 5 Diversity Partitioning
- 6 To Go Further**
 - Advanced Filters
 - Advanced Smoothing**
 - Advanced Import
 - Rarefaction curves

Smoothing with merge_samples (I)

merge_samples merges samples according to a **factor** by **summing** their abundances (beware of different group sizes and library sizes)

```
mergedData <- merge_samples(food, "EnvType")
```

```
## Warning in asMethod(object): NAs introduits lors de la conversion  
automatique
```

```
## Warning in asMethod(object): NAs introduits lors de la conversion  
automatique
```

```
mergedData
```

```
## phyloseq-class experiment-level object
```

```
## otu_table() OTU Table: [ 508 taxa and 8 samples ]
```

```
## sample_data() Sample Data: [ 8 samples by 3 sample variables ]
```

```
## tax_table() Taxonomy Table: [ 508 taxa by 7 taxonomic ranks ]
```

```
## phy_tree() Phylogenetic Tree: [ 508 tips and 507 internal nodes ]
```

```
sample_names(mergedData)
```

```
## [1] "BoeufHache"
```

```
"VeauHache"
```

```
"DesLardons"
```

```
"MerguezVolail"
```

```
## [5] "SaumonFume"
```

```
"FiletSaumon"
```

```
"FiletCabillaud"
```

```
"Crevette"
```

Smoothing with merge_samples (I)

merge_samples merges samples according to a **factor** by **summing** their abundances (beware of different group sizes and library sizes)

```
mergedData <- merge_samples(food, "EnvType")
```

```
## Warning in asMethod(object): NAs introduits lors de la conversion  
automatique
```

```
## Warning in asMethod(object): NAs introduits lors de la conversion  
automatique
```

```
mergedData
```

```
## phyloseq-class experiment-level object
```

```
## otu_table() OTU Table: [ 508 taxa and 8 samples ]
```

```
## sample_data() Sample Data: [ 8 samples by 3 sample variables ]
```

```
## tax_table() Taxonomy Table: [ 508 taxa by 7 taxonomic ranks ]
```

```
## phy_tree() Phylogenetic Tree: [ 508 tips and 507 internal nodes ]
```

```
sample_names(mergedData)
```

```
## [1] "BoeufHache"
```

```
"VeauHache"
```

```
"DesLardons"
```

```
"MerguezVolail
```

```
## [5] "SaumonFume"
```

```
"FiletSaumon"
```

```
"FiletCabillaud"
```

```
"Crevette"
```

Smoothing with merge_samples (I)

merge_samples merges samples according to a **factor** by **summing** their abundances (beware of different group sizes and library sizes)

```
mergedData <- merge_samples(food, "EnvType")
```

```
## Warning in asMethod(object): NAs introduits lors de la conversion  
automatique
```

```
## Warning in asMethod(object): NAs introduits lors de la conversion  
automatique
```

```
mergedData
```

```
## phyloseq-class experiment-level object
```

```
## otu_table() OTU Table: [ 508 taxa and 8 samples ]
```

```
## sample_data() Sample Data: [ 8 samples by 3 sample variables ]
```

```
## tax_table() Taxonomy Table: [ 508 taxa by 7 taxonomic ranks ]
```

```
## phy_tree() Phylogenetic Tree: [ 508 tips and 507 internal nodes ]
```

```
sample_names(mergedData)
```

```
## [1] "BoeufHache"
```

```
"VeauHache"
```

```
"DesLardons"
```

```
"MerguezVolail
```

```
## [5] "SaumonFume"
```

```
"FiletSaumon"
```

```
"FiletCabillaud"
```

```
"Crevette"
```

Smoothing with merge_samples (II)

Unfortunately, merging the contextual data is hard to do automatically in a meaningful way and information is lost in the process...

```
sample_data(mergedData)[1:2, ]
```

```
## Sample Data:          [2 samples by 3 sample variables]:  
##           EnvType FoodType Description  
## BoeufHache      1      NA          NA  
## VeauHache       2      NA          NA
```

```
sample_data(food)[1:2, ]
```

```
## Sample Data:          [2 samples by 3 sample variables]:  
##           EnvType FoodType Description  
## DLT0.LOT08 DesLardons      Meat      LOT8  
## DLT0.LOT05 DesLardons      Meat      LOT5
```

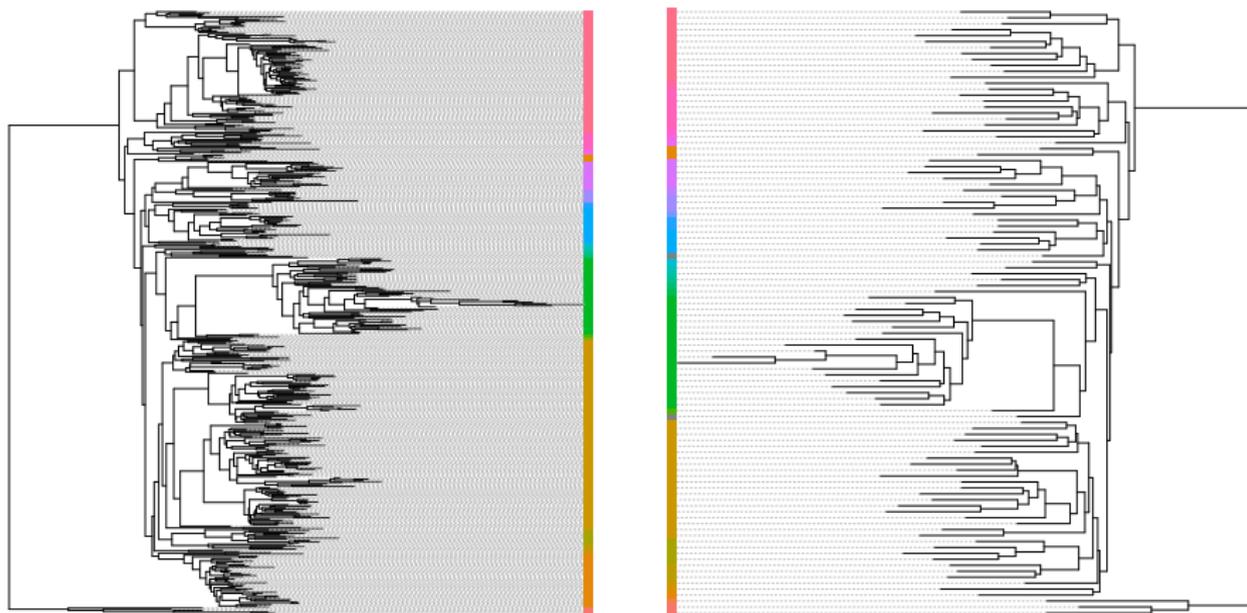
Smoothing with `tip_glom` (I)

`tip_glom` agglomerates otus at a given **height** in the tree.

```
mergedData <- tip_glom(food, h = 0.3)
```

Smoothing with tip_glom (II)

Again, the effect is best understood on the phylogenetic tree.



- 1 Goals of the tutorial
- 2 phyloseq
- 3 Biodiversity indices
- 4 Exploring the structure
- 5 Diversity Partitioning
- 6 **To Go Further**
 - Advanced Filters
 - Advanced Smoothing
 - **Advanced Import**
 - Rarefaction curves

Manual import

- It is possible to build a phyloseq object from **plain tabular files**.
- Since otus/sample names are not always consistent (unlike in a biom), some care must be taken.
- Otherwise the phyloseq objects consists only of otus and samples with **consistent** names and may end up empty.
- Import each component separately;
- Convert to correct base R data type (matrix for `otu_table` and `tax_table`, `data.frame` for `sample_data`)
- Convert to phyloseq data type (`otu_table`, `tax_table`, `sample_data`)
- Check name consistency
- Build phyloseq object

Manual import

- It is possible to build a `phyloseq` object from plain tabular files.
- Since otus/sample names are **not always consistent** (unlike in a `biom`), some care must be taken.
- Otherwise the `phyloseq` objects consists only of otus and samples with **consistent** names and may end up empty.
- Import each component separately;
- Convert to correct base R data type (`matrix` for `otu_table` and `tax_table`, `data.frame` for `sample_data`)
- Convert to `phyloseq` data type (`otu_table`, `tax_table`, `sample_data`)
- Check name consistency
- Build `phyloseq` object

Manual import

- It is possible to build a `phyloseq` object from plain tabular files.
- Since otus/sample names are not always consistent (unlike in a `biom`), some care must be taken.
- Otherwise the `phyloseq` objects consists only of otus and samples with **consistent** names and **may end up empty**.
- Import each component separately;
- Convert to correct base R data type (`matrix` for `otu_table` and `tax_table`, `data.frame` for `sample_data`)
- Convert to `phyloseq` data type (`otu_table`, `tax_table`, `sample_data`)
- Check name consistency
- Build `phyloseq` object

Manual import

- It is possible to build a `phyloseq` object from plain tabular files.
- Since otus/sample names are not always consistent (unlike in a `biom`), some care must be taken.
- Otherwise the `phyloseq` objects consists only of otus and samples with **consistent** names and may end up empty.

- Import each component separately;
 - Convert to correct base R data type (`matrix` for `otu_table` and `tax_table`, `data.frame` for `sample_data`)
 - Convert to `phyloseq` data type (`otu_table`, `tax_table`, `sample_data`)
 - Check name consistency
 - Build `phyloseq` object

Manual import

- It is possible to build a `phyloseq` object from plain tabular files.
- Since otus/sample names are not always consistent (unlike in a `biom`), some care must be taken.
- Otherwise the `phyloseq` objects consists only of otus and samples with **consistent** names and may end up empty.

- Import each component separately;
- Convert to correct base R data type (`matrix` for `otu_table` and `tax_table`, `data.frame` for `sample_data`)
- Convert to `phyloseq` data type (`otu_table`, `tax_table`, `sample_data`)
- Check name consistency
- Build `phyloseq` object

Manual import

- It is possible to build a `phyloseq` object from plain tabular files.
- Since otus/sample names are not always consistent (unlike in a `biom`), some care must be taken.
- Otherwise the `phyloseq` objects consists only of otus and samples with **consistent** names and may end up empty.

- Import each component separately;
- Convert to correct base R data type (`matrix` for `otu_table` and `tax_table`, `data.frame` for `sample_data`)
- Convert to `phyloseq` data type (`otu_table`, `tax_table`, `sample_data`)
- Check name consistency
- Build `phyloseq` object

Manual import

- It is possible to build a `phyloseq` object from plain tabular files.
- Since otus/sample names are not always consistent (unlike in a `biom`), some care must be taken.
- Otherwise the `phyloseq` objects consists only of otus and samples with **consistent** names and may end up empty.

- Import each component separately;
- Convert to correct base R data type (`matrix` for `otu_table` and `tax_table`, `data.frame` for `sample_data`)
- Convert to `phyloseq` data type (`otu_table`, `tax_table`, `sample_data`)
- Check name consistency
- Build `phyloseq` object

Manual import

- It is possible to build a `phyloseq` object from plain tabular files.
- Since otus/sample names are not always consistent (unlike in a `biom`), some care must be taken.
- Otherwise the `phyloseq` objects consists only of otus and samples with **consistent** names and may end up empty.

- Import each component separately;
- Convert to correct base R data type (`matrix` for `otu_table` and `tax_table`, `data.frame` for `sample_data`)
- Convert to `phyloseq` data type (`otu_table`, `tax_table`, `sample_data`)
- Check name consistency
- Build `phyloseq` object

Manual import (II)

Import each component

```
sampladata <- read.csv("data/manual/sampladata.tsv", sep = "\t", row.names = 1)
taxtable <- read.csv("data/manual/taxtable.tsv", sep = "\t", row.names = 1)
otutable <- read.csv("data/manual/otutable.tsv", sep = "\t", row.names = 1)
tree <- read.tree("data/manual/tree.phy")
```

Convert to base R type

```
taxtable <- as.matrix(taxtable)
otutable <- as.matrix(otutable)
```

Convert to phyloseq base type

```
sampladata <- sample_data(sampladata)
taxtable <- tax_table(taxtable)
otutable <- otu_table(otutable, taxa_are_rows = TRUE)
```

Manual import (II)

Import each component

```
sampladata <- read.csv("data/manual/sampladata.tsv", sep = "\t", row.names = 1)
taxtable <- read.csv("data/manual/taxtable.tsv", sep = "\t", row.names = 1)
otutable <- read.csv("data/manual/otutable.tsv", sep = "\t", row.names = 1)
tree <- read.tree("data/manual/tree.phy")
```

Convert to base R type

```
taxtable <- as.matrix(taxtable)
otutable <- as.matrix(otutable)
```

Convert to phyloseq base type

```
sampladata <- sample_data(sampladata)
taxtable <- tax_table(taxtable)
otutable <- otu_table(otutable, taxa_are_rows = TRUE)
```

Manual import (II)

Import each component

```
sampladata <- read.csv("data/manual/sampladata.tsv", sep = "\t", row.names = 1)
taxtable <- read.csv("data/manual/taxtable.tsv", sep = "\t", row.names = 1)
otutable <- read.csv("data/manual/otutable.tsv", sep = "\t", row.names = 1)
tree <- read.tree("data/manual/tree.phy")
```

Convert to base R type

```
taxtable <- as.matrix(taxtable)
otutable <- as.matrix(otutable)
```

Convert to phyloseq base type

```
sampladata <- sample_data(sampladata)
taxtable <- tax_table(taxtable)
otutable <- otu_table(otutable, taxa_are_rows = TRUE)
```

Manual import (III)

Check name consistency

Abundance table and sample data (sample names)

```
all(colnames(otutable) %in% rownames(sampledata)) ## sample names  
  
## [1] TRUE
```

Abundance table and taxonomy table (taxa names)

```
all(rownames(otutable) %in% rownames(taxtable)) ## taxa names  
  
## [1] TRUE
```

Abundance table and tree leaves (taxa names)

```
all(rownames(otutable) %in% tree$tip.label) ## taxa names  
  
## [1] TRUE
```

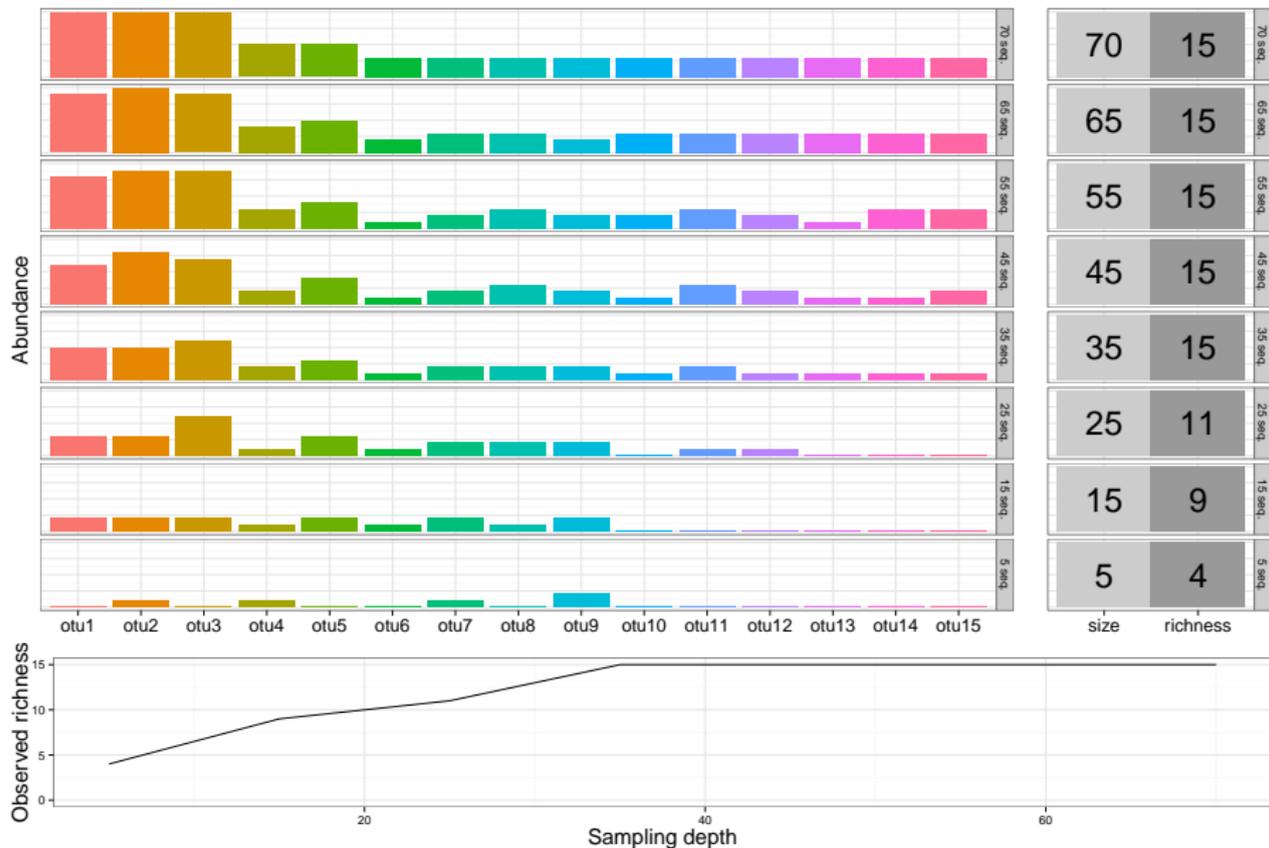
Build object

```
manualData <- phyloseq(sampledata, otutable, taxtable, tree)
manualData

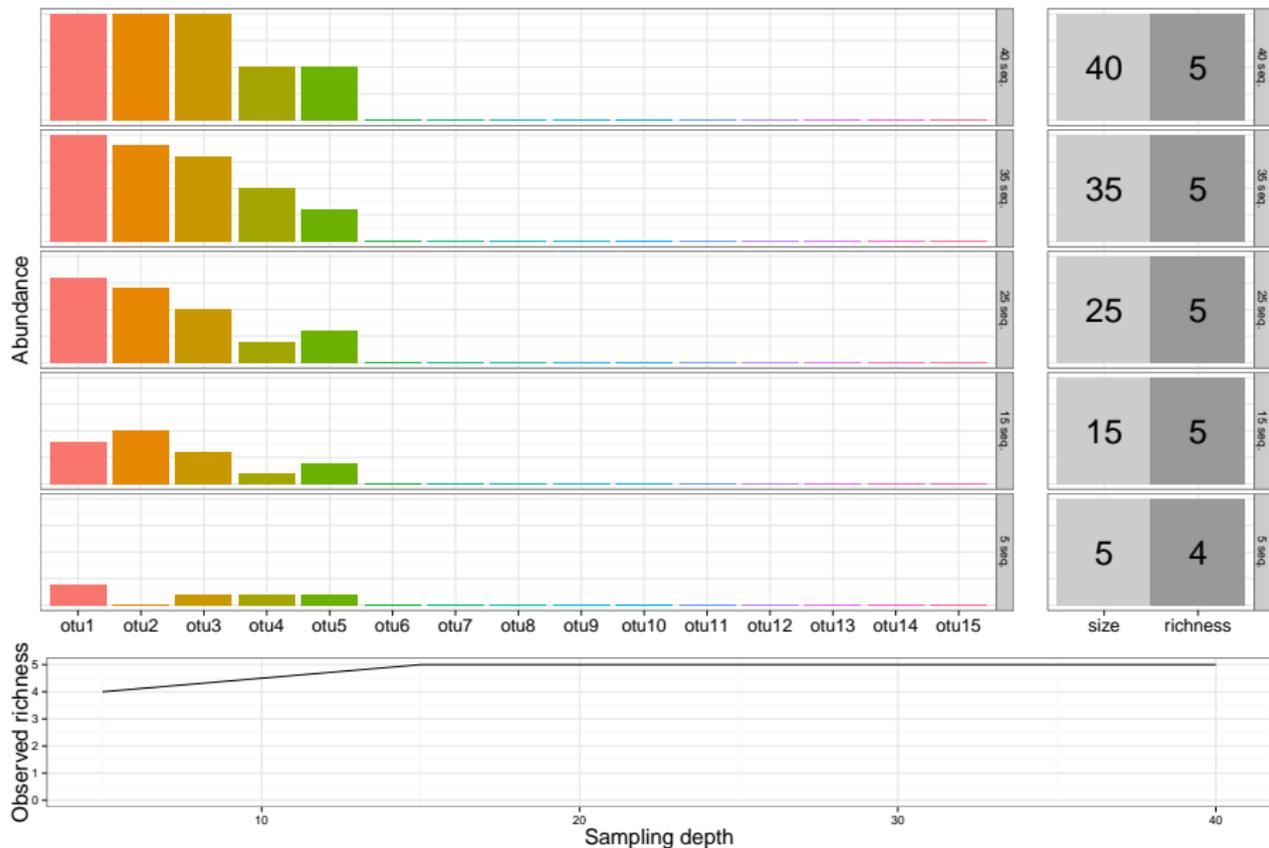
## phyloseq-class experiment-level object
## otu_table()   OTU Table:           [ 500 taxa and 26 samples ]
## sample_data() Sample Data:        [ 26 samples by 6 sample variables ]
## tax_table()   Taxonomy Table:      [ 500 taxa by 7 taxonomic ranks ]
## phy_tree()    Phylogenetic Tree:   [ 500 tips and 499 internal nodes ]
```

- 1 Goals of the tutorial
- 2 phyloseq
- 3 Biodiversity indices
- 4 Exploring the structure
- 5 Diversity Partitioning
- 6 To Go Further**
 - Advanced Filters
 - Advanced Smoothing
 - Advanced Import
 - Rarefaction curves**

Rarefaction curve (I)



Rarefaction curve (after filtering rare otus)



Rarefaction curves (II)

Why?

- α diversity indices are heavily influenced by sampling depths.
- Rarefaction curves assess if sampling has **exhausted** the diversity.

How?

- Rarefy all samples to the same depth (**optional**, for speed here);
- Use custom function **ggrare** and specify a **step** size.

```
food <- rarefy_even_depth(food, rngseed = 1121983)
p <- ggrare(food, step = 1000, color = "SampleType", se = FALSE)
```

- To distinguish different environments easily, use faceting (and plain background)

```
plot(p + facet_wrap(~EnvType) + theme_bw())
```

Rarefaction curves (II)

Why?

- α diversity indices are heavily influenced by sampling depths.
- Rarefaction curves assess if sampling has exhausted the diversity.

How?

- Rarefy all samples to the same depth ([optional](#), for speed here);
- Use custom function `ggrare` and specify a `step` size.

```
food <- rarefy_even_depth(food, rngseed = 1121983)
p <- ggrare(food, step = 1000, color = "SampleType", se = FALSE)
```

- To distinguish different environments easily, use facetting (and plain background)

```
plot(p + facet_wrap(~EnvType) + theme_bw())
```

Rarefaction curves (II)

Why?

- α diversity indices are heavily influenced by sampling depths.
- Rarefaction curves assess if sampling has exhausted the diversity.

How?

- Rarefy all samples to the same depth (`optional`, for speed here);
- Use custom function `ggrare` and specify a `step` size.

```
food <- rarefy_even_depth(food, rngseed = 1121983)
p <- ggrare(food, step = 1000, color = "SampleType", se = FALSE)
```

- To distinguish different environments easily, use facetting (and plain background)

```
plot(p + facet_wrap(~EnvType) + theme_bw())
```

Rarefaction curves (II)

Why?

- α diversity indices are heavily influenced by sampling depths.
- Rarefaction curves assess if sampling has exhausted the diversity.

How?

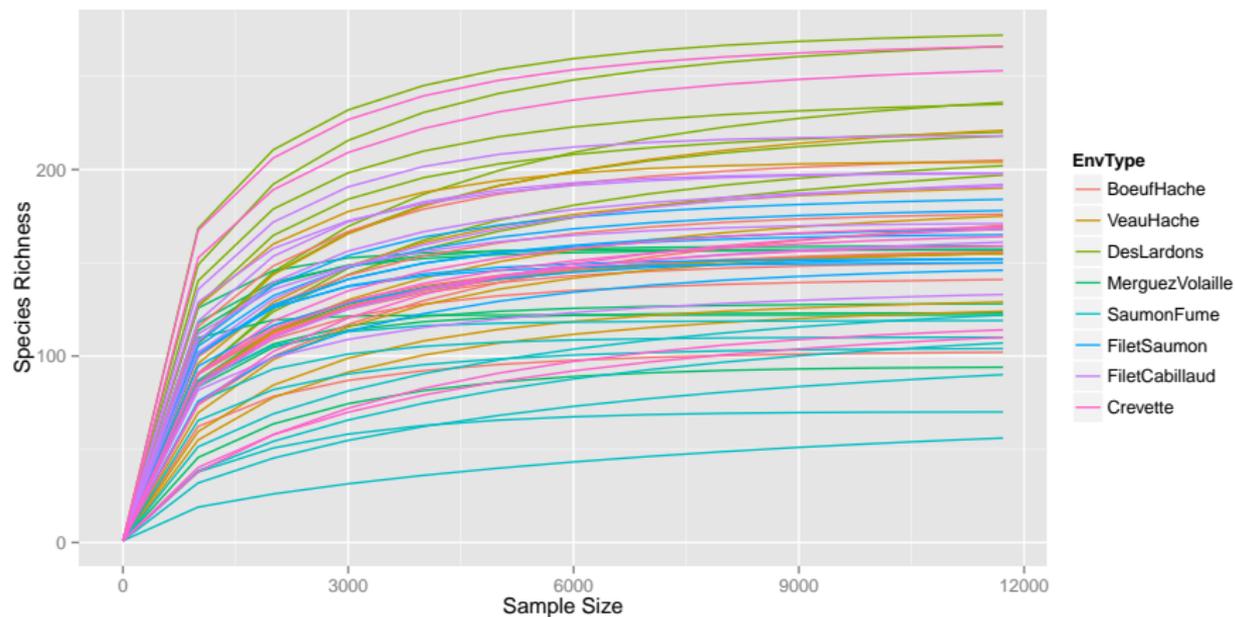
- Rarefy all samples to the same depth (optional, for speed here);
- Use custom function `ggrare` and specify a `step` size.

```
food <- rarefy_even_depth(food, rngseed = 1121983)
p <- ggrare(food, step = 1000, color = "SampleType", se = FALSE)
```

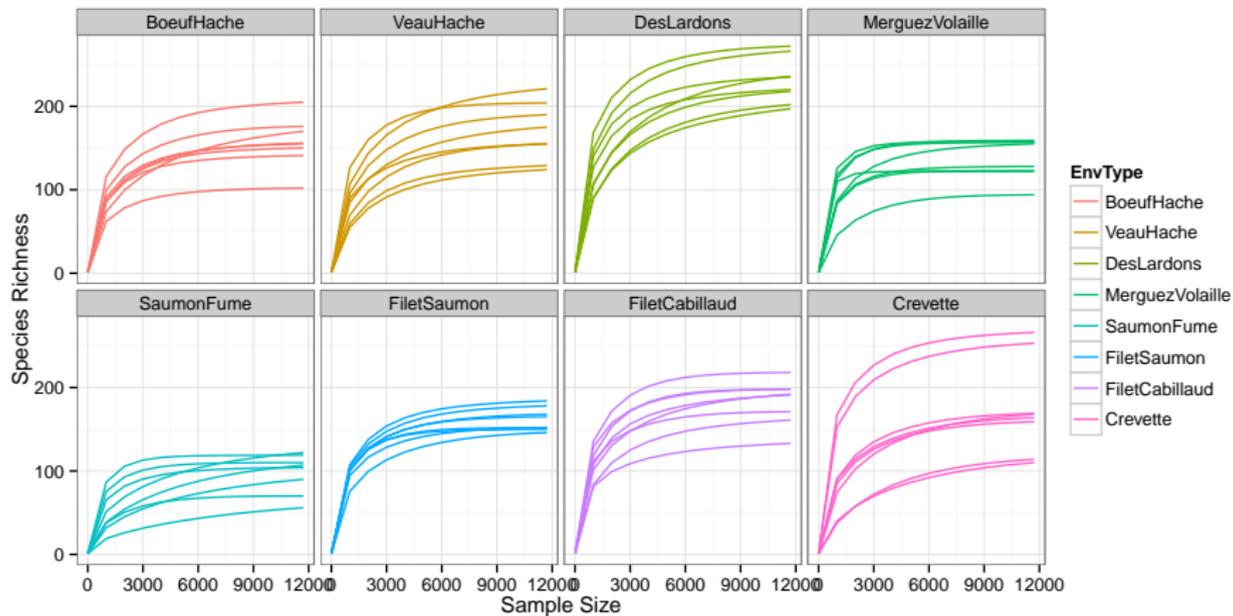
- To distinguish different environments easily, use faceting (and plain background)

```
plot(p + facet_wrap(~EnvType) + theme_bw())
```

Rarefaction curves (III)



Rarefaction curves (IV)



Outline

- 1 Goals of the tutorial
- 2 phyloseq
- 3 Biodiversity indices
- 4 Exploring the structure
- 5 Diversity Partitioning
- 6 To Go Further
 - Advanced Filters
 - Advanced Smoothing
 - Advanced Import
 - Rarefaction curves

Some compositional β dissimilarities

Consider two communities 1 and 2 and note:

- n_s^1 the relative abundance of otu s in community 1;
- n_s^2 the relative abundance of otu s in community 2;
- a the number of otus shared by communities 1 and 2;
- b the number of otus specific to community 1;
- c the number of otus specific to community 2.

Most dissimilarities can be defined in terms of n_s^1 , n_s^2 , a , b , c .

- Bray-Curtis, Jaccard and Kulczynski good at detecting underlying ecological gradients
- Morisita-Horn, Cao and Jensen-Shannon good at handling different sample sizes
- All take value in $[0, 1]$ except JSD and Cao.

Some compositional β dissimilarities

Consider two communities 1 and 2 and note:

- n_s^1 the relative abundance of otu s in community 1;
- n_s^2 the relative abundance of otu s in community 2;
- a the number of otus shared by communities 1 and 2;
- b the number of otus specific to community 1;
- c the number of otus specific to community 2.

Most dissimilarities can be defined in terms of n_s^1 , n_s^2 , a , b , c .

- Bray-Curtis, Jaccard and Kulczynski good at detecting underlying ecological gradients
- Morisita-Horn, Cao and Jensen-Shannon good at handling different sample sizes
- All take value in $[0, 1]$ except JSD and Cao.

A few compositional dissimilarities

Name	Definition
Based on abundance data	
Bray-Curtis	$BC(1, 2) = \frac{\sum_i n_s^1 - n_s^2 }{\sum_i n_s^1 + n_s^2}$
Jaccard	$JC(1, 2) = \frac{2BC(1, 2)}{1 + BC(1, 2)}$
Kulczynski	$KU(1, 2) = 1 - \frac{1}{2} \left(\frac{\sum_i \min(n_s^1, n_s^2)}{\sum_i n_s^1} + \frac{\sum_i \min(n_s^1, n_s^2)}{\sum_i n_s^2} \right)$
Morisita-Horn	$M(1, 2) = 1 - 2 \frac{\sum_i n_s^1 n_s^2}{(\lambda(1) + \lambda(2)) \sum_i n_s^1 n_s^2} \text{ where } \lambda(1) = \frac{\sum_i (n_s^1)^2}{\sum_i n_s^1}$
Cao	$C1O(1, 2) = \frac{1}{a + b + c} \sum_i \left(\log \frac{n_s^1 + n_s^2}{2} - \frac{n_s^1 \log n_s^1 + n_s^2 \log n_s^2}{n_s^1 + n_s^2} \right)$
JSD	$JSD(1, 2) = H(1) + H(2) - H(1 + 2) \text{ where } H(1) = \frac{\sum_i n_s^1 \log \left(\frac{n_s^1}{\sum_i n_s^1} \right)}{\sum_i n_s^1}$
Based on presence/absence data	
Bray-Curtis/Sorensen	$BC(1, 2) = \frac{b + c}{2a + b + c}$
Jaccard	$JC(1, 2) = \frac{b + c}{a + b + c}$