Microbial community dynamics based on amplicon sequence variants (ASVs)

Cristian Díaz-Muñoz & Florac De Bruyne

v1.0

Contents

Introduction

This is a guide to plot the relative abundance of genera/species of interest based on amplicon sequencing data. The script is based on a more basic, previous one made by Florac De Bruyne. The script is meant for amplicon sequences that have been processed using `DADA2` (Callahan *et al.,* 2017, 2019). Therefore, you will need two files to start this script:

1. **Sequence table**. A tab separated file containing the amplicon sequence variants (ASVs) inferred using `DADA2` and the number of reads per sample. Every column is a sample. It is possible that the name of the first column is not present in the text file. If so, edit it and include "Sequence" (+ a tab) at the very beginning. This will be the column name of the first column.
2. **Taxonomy table**. A tab separated file containing the ASVs and the taxonomy classification of each one, from kingdom to species. Again, the first column name may be missing, so add "Sequence" again at the very beginning of the file. We will use these names to join the two data frames in R.

In this example, we will use a dataset of 12 different cocoa fermentation processes with around 9 timepoint per fermentation. We extracted the whole community DNA, performed a PCR with primers that amplify the full bacterial 16S rRNA gene, and sequenced the amplicons using PacBio circular consensus sequencing (*i.e.,* PacBioHiFi) at the university facility of Leuven. However, this script is valid and can be adapted for any kind of amplicon sequences (16S, V4, V3-V4, ITS1, ITS2, *etc.*) as long as they have been processed using `DADA2`.

Microbial community dynamics at genus level

This part of the script will execute some commands to adapt the two tables described above to a format that we can work with, transform the data, group the ASVs at genus level, and generate a barplot of relative abundances.

Set up the basics of the script

First of all, let's set the working directory and the load the libraries needed for this script.

```
setwd("C:/Users/cdiazmun/OneDrive - Vrije Universiteit Brussel/IMDO/R scripts IMDO/Amplicon sequencing/

library(reshape2) # Flexibly Reshape Data. To manipulate data.
library(tidyverse) # To manipulate data to make it look easy and tidy.
library(ggthemes) # Extra Themes, Scales and Geoms for 'ggplot2'.
```

Next, we are going to load two functions that will be used afterwards. The first one is to calculate the per-column percentage of relative abundance of every row and the second is to group very low occuring rows through the whole dataset under "Minorities". Why do I speak about rows and not ASVs? Because the functions do not care about what is in there, it will take whatever is in the rows, and this can be ASvs themselves, species or genera. And this is very important for the interpretation of the data, but you will see it more clearly when we actually use the functions.

```
# Function to calculate the relative abundances
percentage <- function(d){
  d_per <- apply(d[2:ncol(d)], 2, function(x) x/sum(x)*100)
  d_per <- cbind(d[1],d_per)
  return(d_per)
}
# Function to put the genus/species that appear in less than "threshold" in others/minorities
minoritize_loose <- function(d,threshold){
  d_min <- d[!(apply(d[2:ncol(d)] < threshold, 1, function(x) all(x))),]
  minorities <- c("Minorities", apply(d_min[2:ncol(d_min)], 2, function(x) 100-sum(x)))
  d_min <- rbind(d_min, minorities)
  d_min[,2:ncol(d_min)] <- apply(d_min[,2:ncol(d_min)], 2, as.numeric)
  return(d_min)
}
```

Import the input files and get the relative abundance table

Read the taxonomy table (`bacteria_taxa`) and the sequence table (`bacteria_reads`).

```
bacteria_taxa <- read_tsv("Test data/Taxonomy_pseudo",col_names=TRUE)
bacteria_reads <- read_tsv("Test data/Reads_pseudo",col_names=TRUE)
```

Merge the two data frames using the first column (`Sequence`) as the common vector for the merging.

```
bacteria <- merge(bacteria_taxa, bacteria_reads,
                  by="Sequence",
                  all=T)
```

In my dataset there are some mistakes in the labelings and some samples that had to be repeated. As this is a quite often ocurring mistake, I decided to also show how to deal with these issues. In the next code line we change the positions between F02T96 and F04T96 and remove repeated and failed samples (F03T006, F03T024, F03T048, F12T072).

```
bacteria <- bacteria[c(1:22,43,24,25,26,28,29,32:42,23,44:107,109,110,111)]
```

Since we are only interested in the microbial composition at genus-level, we are going to select only the `Genus` column and discard the `Species` column. It is in general a **good practice** to create a new object every time we make a change to the data frame. This way if there is an error in the next steps or if we something weird in the data frame, we can go back step by step and see where the mistake was. Also, I **recommend** viewing the data frames as we transform them by clicking on them at the right-hand side, under Global Environment (in RStudio).

```
bacteria_genus <- bacteria[,7:ncol(bacteria)]
bacteria_genus <- bacteria_genus[,-2]
```

Here we are going to do a rather important step. We have sequenced the 16S, to get a view of all bacteria in the sample. However, there are more things than microorganisms in the fermentation matrix. In cocoa, there is plant material, from the cocoa tree. In sourdough, there is also plant material, from wheat. And plant cells have chloroplasts and mitochondria, which are ancient bacteria. Therefore, they also harbour a 16S gene. In this analysis we are not interested on watching the proportion of plant material in our samples, so we will filter them out. If you want to have a look, you are always free to keep them.

Reclassify the `Genus` with mithocondria or chloroplast as plant material

```
bacteria_genus$Genus[bacteria$Order=="Rickettsiales"] <- "Plant material"
bacteria_genus$Genus[bacteria$Order=="Chloroplast"] <- "Plant material"
```

Reclassify the `Family` name Enterobacteriaceae as `Genus` to be able to appear on the legend and be classified as that This is normally a necessary step if you're sequencing small fragments of the 16S, such as the V3-V4 region, as these regions have not enough discriminatory power for the Enterobacteria lineage. If your ASVs originate from full 16S amplicons, you don't need to run the following line.

```
bacteria_genus$Genus[is.na(bacteria_genus$Genus) & bacteria$Family=="Enterobacteriaceae"] <- "Enterobac
```

All the rest rows that are unclassified will be as "Above genus".

```
bacteria_genus$Genus[is.na(bacteria_genus$Genus)] <- "Above genus"
```

Finally, let's remove the plant material from the dataset. It is important that we do this step before calculating the relative abundances. If you want to keep going with the plant material in, just continue the next steps using `bacteria_genus` instead of `bacteria_genus_noplants`.

```
bacteria_genus_noplants <- bacteria_genus %>%
  filter(!Genus == "Plant material")
```

In the next steps we are going to perform some calculations, so we better transform any `NA` value to 0 to avoid problems. We do this step now, because there are not any NA anymore at the Genus column.

```
bacteria_genus_noplants[is.na(bacteria_genus_noplants)] <- 0
```

Transform the reads in percentage of relative abundance per sample and merge the ASV that correspond to the same genus.

```
bacteria_final <- percentage(bacteria_genus_noplants)
bacteria_final <- aggregate(bacteria_final[,2:ncol(bacteria_final)],
                            by=list(bacteria_final[,"Genus"]), FUN=sum)
colnames(bacteria_final)[1] <- "Taxon"
```

Now we can group the genera that are in less than xx % of relative abundance in any sample. In other words, if there is only one sample in which a genus has more than xx %, it will be included as such, otherwise grouped under minorities. We use 1 % when plant material is included and 0.1% when plant material is not included. It is better to apply minoritize after Genus melting, so we can represent more accurately the Genus that are present in the fermentation. The threshold to set depends on your sample diversity and which story you want to tell. If you are interested in watching the big picture and the main microorganisms prevailing over the fermentation, you can increase that threshold. If you want to know all the small microbial groups that are present in, for example, initial samples of a spontaneous process, then you can decrease the threshold to include more genera. Also consider that the more genera are plotted, the more difficult it gets for the reader to interpret the data. and it is also more difficult to choose divergent enough colors, *etc*,.

```
bacteria_final_min <- minoritize_loose(bacteria_final, 0.1)
head(bacteria_final_min[1:10,1:10])
```

```
##                Taxon   F01T000    F01T006  F01T012    F01T024    F01T048
## 1        Above genus 23.655914 21.8045113 1.321586 1.16333725 0.55678117
## 2        Acetobacter  4.659498  0.3759398 0.000000 0.10575793 0.00000000
## 3  Actinomycetospora  0.000000  0.7518797 0.000000 0.00000000 0.00000000
## 5    Brachybacterium  0.000000  0.0000000 0.000000 0.00000000 0.00000000
## 6 Cellulosimicrobium  3.225806  1.8796992 1.101322 0.05875441 0.05252653
## 7       Enterobacter  0.000000  0.0000000 0.000000 0.03525264 0.00000000
##     F01T072     F01T096     F01T120   F02T000
## 1 0.5433552  0.04808463  0.01824818 12.903226
## 2 0.6225945 34.70107389 67.25364964  5.806452
## 3 0.0000000  0.00000000  0.00000000  0.000000
## 5 0.0000000  0.00000000  0.00000000  0.000000
## 6 0.0226398  0.00000000  0.05474453  4.516129
## 7 0.0339597  0.00000000  0.00000000  0.000000
```

At this point, we can save the data frame as a text file, if we want to inspect it using Excel or whatever.

```
write_excel_csv2(bacteria_final_min, "Test data/bacteria_genus_relative_abundance_pseudo.csv")
```

Reshape the data frame and incorporate factors for plotting

We are going to plot using `ggplot2` and this package likes dataframes in a "long-table" style. So, the first thing we need to do is to organize the data frame in such way. We will use the function `melt()` from the `Reshape2`package.

```
bacteria_melted <- melt(bacteria_final_min, id.vars="Taxon")
head(bacteria_melted, n=10)
```

```
##                 Taxon variable     value
## 1         Above genus  F01T000 23.655914
## 2         Acetobacter  F01T000  4.659498
## 3   Actinomycetospora  F01T000  0.000000
## 4     Brachybacterium  F01T000  0.000000
## 5  Cellulosimicrobium  F01T000  3.225806
## 6        Enterobacter  F01T000  0.000000
## 7             Erwinia  F01T000  7.526882
## 8           Frateuria  F01T000  0.000000
## 9       Gluconobacter  F01T000  0.000000
## 10         Klebsiella  F01T000  1.792115
```

Now, we are going to add extra columns with any kind of information that we consider of relevance. We can use these new columns as factors or groups, to sort, plot, divide, or filter our data. In my case, I am going to start by adding information about the `Fermentation` process and the `Vessel` in which the fermentation was performed. Then, we will add the time points as a numeric scale. A way of doing is by using the function `rep()`. The logic behind the code you can find below is the following: my samples are well sorted by fermentation and time point by default; in the melted table, every sample is repeated the same number of times as Genera are in the final dataset (24); I have 16 time points for the "Negative control". Then I add 24*16 times "Negative control". And I do the same for the rest of fermentation processes. Why do not I write 384, instead of 24*16? Because if I want to reprocess the data changing the threshold for minorities, I will have a different number of genera represented in my final dataset and this way I can easily substitute 24* by the number that I want, using `Ctrl+F`.

```
bacteria_melted$Fermentation <- c(rep("Negative control",24*16),
                                  rep("Positive control",24*17),
                                  rep("AFSC V",24*16),
                                  rep("AFSC VI",24*16),
                                  rep("AFSC VII",24*18),
                                  rep("AFSC VIII",24*16))

bacteria_melted$Vessel <- c(rep("NC (F01)",24*8), rep("NC (F02)",24*8),
                            rep("PC (F03)",24*8), rep("PC (F04)",24*9),
                            rep("AFSC V (F05)",24*9), rep("AFSC V (F06)",24*7),
                            rep("AFSC VI (F07)",24*8), rep("AFSC VI (F08)",24*8),
                            rep("AFSC VII (F09)",24*9), rep("AFSC VII (F10)",24*9),
                            rep("AFSC VIII (F11)",24*8), rep("AFSC VIII (F12)",24*8))
```

For the time points, we could do the same, but I decided to be a bit inventive this time. Since the information of the time point is embedded in the sample name, I can subtract that information and add it as a new column. Finally, we transform that column to numeric.

```
# We are taking the names in the "variable" column as characters (strings) and
# dividing them using "T" as separator. Then we are selecting the second field.
bacteria_melted$Timepoint <- sapply(strsplit(basename(as.character(bacteria_melted$variable)),
                                             "T"), `[`,2)
# Transform the column into numeric.
bacteria_melted$Timepoint <- as.numeric(as.character(bacteria_melted$Timepoint))
bacteria_melted <- bacteria_melted %>%
  transform(bacteria_melted, Timepoint = as.numeric(Timepoint))
# Remove other columns that were generated in the process and that we don't want.
bacteria_melted <- bacteria_melted[,-c(7:ncol(bacteria_melted))]
```

Good. We added all factors that we wanted. The next code line will tell R in which order we want them to appear when we plot the data:

```
bacteria_melted$Vessel <- factor(bacteria_melted$Vessel,
                                 levels = c("NC (F01)", "NC (F02)",
                                            "PC (F03)", "PC (F04)",
                                            "AFSC V (F05)", "AFSC V (F06)",
                                            "AFSC VI (F07)", "AFSC VI (F08)",
                                            "AFSC VII (F09)", "AFSC VII (F10)",
                                            "AFSC VIII (F11)", "AFSC VIII (F12)"))
```

Next, we define a vector with the genera present in the order that we want them to appear in the plot. My way of sorting the microbial genera/species in a relative abundance plot is taxonomy-based. That way it is easier to make sense of the plot, which group of microorganisms grow and thrive and which ones decline over the fermentation.

```
taxa_genus <- c("Above genus", "Minorities",
                "Cellulosimicrobium", "Actinomycetospora", "Brachybacterium",
                "Erwinia", "Frateuria", "Pseudomonas","Enterobacter", "Klebsiella",
                "Kluyvera", "Pantoea", "Tatumella","Lactococcus","Leuconostoc",
                "Weissella", "Lactiplantibacillus", "Levilactobacillus",
                "Limosilactobacillus", "Liquorilactobacillus", "Paucilactobacillus",
                "Acetobacter", "Gluconobacter", "Komagataeibacter")
```

```r
bacteria_melted <- bacteria_melted %>%
  mutate(Taxon = factor(Taxon, levels = taxa_genus))
```

Next, we set the colors that we want for the genera. I also choose the colors taxonomy-wise. The order of the colors is very important, they need to be in the same order as the genera.

```r
taxa_last <- c("azure4", "antiquewhite2")
taxa_rest <- c('#ffffd4','#fee391','#fec44f','#fe9929','#d95f0e','#993404') # Yellows
taxa_entero <- c('#fcbba1','#fc9272','#fb6a4a','#de2d26','#a50f15') # Reds
taxa_lactoc <- "#FF99FF" # Pink
taxa_leucon <- c("#CC99FF","#9933CC") # Purples
taxa_lactob <- c('#c6dbef','#9ecae1','#6baed6','#3182bd','#08519c') # Blues
taxa_aab <- c('#74c476','#31a354','#006d2c') # Greens
# Now we join all the vectors in a single one
taxa_col <- c(taxa_last,  taxa_rest, taxa_entero, taxa_lactoc,
              taxa_leucon, taxa_lactob, taxa_aab)
```

Finally, the last touch. We define a vector to write the names in *italic*. Again, the order of the genera needs to be the same as in the `taxa_genus` column.

```r
mylabels <- c("Above genus", "Minorities",
              expression(italic("Cellulosimicrobium")),
              expression(italic("Actinomycetospora")),
              expression(italic("Brachybacterium")),
              expression(italic("Erwinia")),
              expression(italic("Frateuria")),
              expression(italic("Pseudomonas")),
              expression(italic("Enterobacter")),
              expression(italic("Klebsiella")),
              expression(italic("Kluyvera")),
              expression(italic("Pantoea")),
              expression(italic("Tatumella")),
              expression(italic("Lactococcus")),
              expression(italic("Leuconostoc")),
              expression(italic("Weissella")),
              expression(italic("Lactiplantibacillus")),
              expression(italic("Levilactobacillus")),
              expression(italic("Limosilactobacillus")),
              expression(italic("Liquorilactobacillus")),
              expression(italic("Paucilactobacillus")),
              expression(italic("Acetobacter")),
              expression(italic("Gluconobacter")),
              expression(italic("Komagataeibacter")))
```

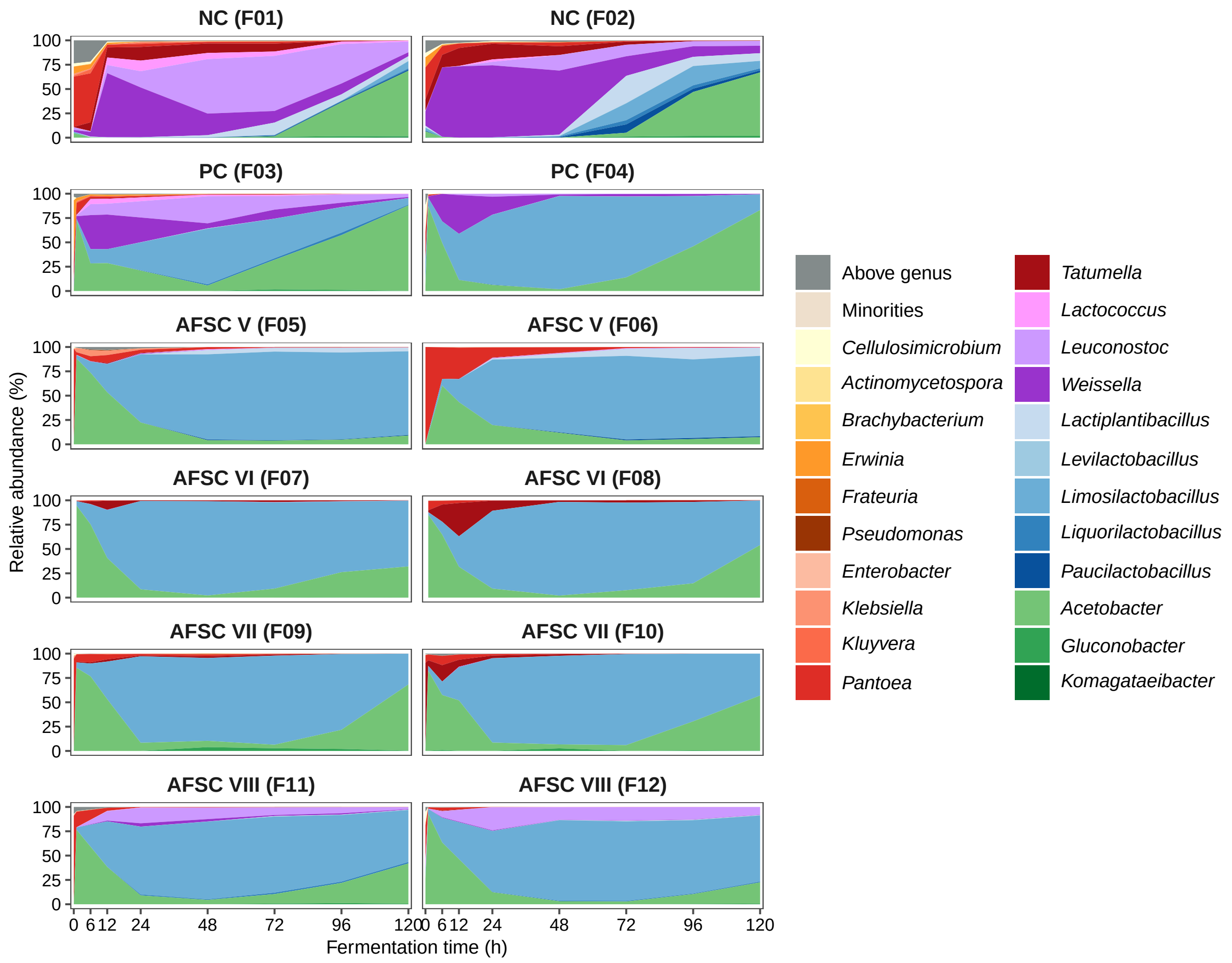
We can finally create a plot!

Plotting a relative abundance figure

First, we are going to plot the relative abundance of every genus as a function of the fermentation time. For that, we supply the `Timepoint`column as a numerical vector to `aes()`. The colors are defined by `fill` or `color` depending on the `geom` used. When the x-axis is a continuous scale and the y-axis always goes from

0 to 100, is a nice idea to use `geom_area()`. We use then `facet_wrap()` to separate each `Vessel` separately, inside the same plot. All the other things are pure aesthetics.

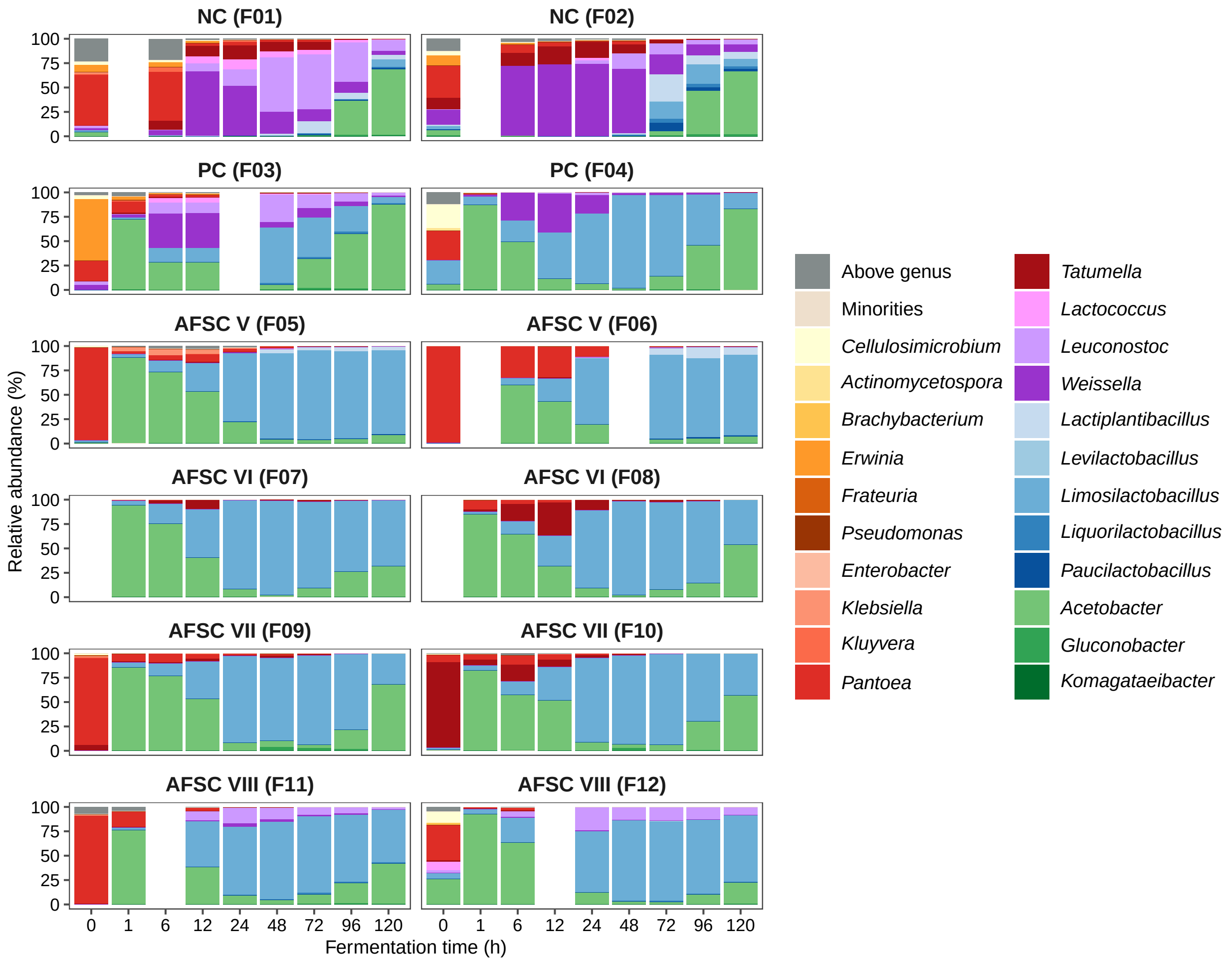
```r
ggplot(bacteria_melted, aes(x = Timepoint, y = value, fill = Taxon)) +

  geom_area(stat = "identity", position = position_stack()) +

  facet_wrap(.~Vessel, ncol = 2) +

  scale_x_continuous(breaks = c(0,6,12,24,48,72,96,120),
                     expand = c(0.01, 0.01)) +

  scale_fill_manual(labels = mylabels,
                    values = taxa_col) +

  theme_few() +

  theme(legend.title = element_blank(),
        legend.text = element_text(size=11),
        legend.position = "right",
        axis.text.x = element_text(size = 10, colour = "black"),
        axis.text.y = element_text(size = 10, colour = "black"),
        axis.title = element_text(size=11),
        legend.key.height = unit(1.5,"line"),
        legend.key.width = unit(1.5,"line"),
        legend.text.align = 0,
        strip.text = element_text(face = "bold", size = 12)) +

  labs(title = element_blank(),
       labels = mylabels,
       y = "Relative abundance (%)",
       x = "Fermentation time (h)")
```



Fantasties. But we have many time points during the first 24 h of fermentation, these samples are kind of unermined with this representation. Let's plot now the same, but using the `Timepoint` column as a factor and every time point represented with `geom_bar()`.

```
ggplot(bacteria_melted, aes(x = as.factor(Timepoint), y = value, fill = Taxon)) +

  geom_bar(stat = "identity", position = position_stack()) +

  facet_wrap(.~Vessel, ncol = 2) +

  scale_fill_manual(labels = mylabels,
                    values = taxa_col) +

  theme_few() +

  theme(legend.title = element_blank(),
        legend.text = element_text(size=11),
        legend.position = "right",
        axis.text.x = element_text(size = 10, colour = "black"),
        axis.text.y = element_text(size = 10, colour = "black"),
        axis.title = element_text(size=11),
        legend.key.height = unit(1.5,"line"),
```

```r
        legend.key.width = unit(1.5,"line"),
        legend.text.align = 0,
        strip.text = element_text(face = "bold", size = 12)) +

  labs(title = element_blank(),
       labels = mylabels,
       y = "Relative abundance (%)",
       x = "Fermentation time (h)")
```



Voilà, your marvelous relative abundance figure representing the microbial community dynamics at genus level is finished! Let's save the plot. As you may have seen previously, we are going to control the size of the items by playing with the `width` and `height` options of `ggsave()`. I always like to generate my figures in a different folder, named "Plots". Keep the dpi at 300, as this is the optimal resolution for publications.

```r
ggsave("Bacteria_barplot.tiff", path = "Plots", width = 10, height = 8, units = "in", dpi = 300)
```

Microbial community dynamics at species level

The most straight-forward option would just be to join the `Species` column to the `Genus` column at the beginning of the pipeline and generate a data frame named like `bacteria_species` instead of `bacteria_genus` and afterwards aggregate the ASVs by species. But in the example I illustrate below, I prefer to inspect the species diversity by microbial group and don't consider any species in `Minorities`. But you are free to adapt the codes below to represent all species in your samples, using the `minoritize_loose` function. We are going to generate a plot of lactic acid bacteria (LAB) dynamics at species level. Let's start from the very beginning, to be sure we don't make any mistake with our data frames:

```
bacteria_taxa <- read_tsv("Test data/Taxonomy_pseudo",col_names=TRUE)
bacteria_reads<- read_tsv("Test data/Reads_pseudo",col_names=TRUE)
bacteria <- merge(bacteria_taxa, bacteria_reads,
                  by="Sequence",
                  all=T)
bacteria <- bacteria[c(1:22,43,24,25,26,28,29,32:42,23,44:107,109,110,111)]
bacteria_genus<-bacteria[,7:ncol(bacteria)]
```

Now, we are going to filter the data to get only LAB and then join the species and genus names:

```
# Filter-in all LAB genera in your dataset
lab <- bacteria_genus %>%
  filter(Genus %in% c("Lactococcus", "Leuconostoc", "Weissella",
                      "Lactiplantibacillus","Levilactobacillus","Limosilactobacillus",
                      "Liquorilactobacillus","Paucilactobacillus"))
# Join the two columns
lab$Genus <- paste(lab$Genus, lab$Species)
# Rename those ASVs that did not have any species information
lab$Genus[lab$Genus =="Leuconostoc NA"] <- "Leuconostoc"
lab$Genus[lab$Genus =="Liquorilactobacillus NA"] <- "Liquorilactobacillus"
lab$Genus[lab$Genus =="Weissella NA"] <- "Weissella"
# Remove species column
lab <- lab[,-2]
```

Let's now group the ASVs by the species they belong to and calculate the relative abundance. Keep in mind that now, 100 % is equivalent to all LAB present in the sample, not all bacteria present in the sample.

```
lab_final <- aggregate(lab[,2:ncol(lab)],by=list(lab[,"Genus"]),FUN=sum)
colnames(lab_final)[1] <- "Taxon"
lab_relative <- percentage(lab_final)
```

```
# We can output this table as we did with the genus table before
write_excel_csv2(lab_relative, "Tables/lab_species_relative_abundance_pseudo.csv")
```

We proceed again with the reshape of the data frame and the addition of all factors for plotting. It becomes quite easy to copy-paste the previous codes and just change some numbers:

```
# Reshape the data frame
lab_melted<- melt(lab_relative,id.vars="Taxon")
# Add Fermentation column
lab_melted$Fermentation <- c(rep("Negative control",19*16),
                             rep("Positive control",19*17),
                             rep("AFSC V",19*16),
                             rep("AFSC VI",19*16),
                             rep("AFSC VII",19*18),
                             rep("AFSC VIII",19*16))
```

```r
# Add Vessel column
lab_melted$Vessel <- c(rep("NC (F01)",19*8), rep("NC (F02)",19*8),
                            rep("PC (F03)",19*8), rep("PC (F04)",19*9),
                            rep("AFSC V (F05)",19*9), rep("AFSC V (F06)",19*7),
                            rep("AFSC VI (F07)",19*8), rep("AFSC VI (F08)",19*8),
                            rep("AFSC VII (F09)",19*9), rep("AFSC VII (F10)",19*9),
                            rep("AFSC VIII (F11)",19*8), rep("AFSC VIII (F12)",19*8))
# Add Time point column
lab_melted$Timepoint <- sapply(strsplit(basename(as.character(lab_melted$variable)), "T"), `[`,2)
lab_melted$Timepoint <- as.numeric(as.character(lab_melted$Timepoint))
lab_melted <- lab_melted %>%
  transform(lab_melted, Timepoint = as.numeric(Timepoint))
lab_melted <- lab_melted[,-c(7:ncol(lab_melted))]
# Sort the data by vessel for plotting purposes
lab_melted$Vessel <- factor(lab_melted$Vessel,
                                 levels = c("NC (F01)", "NC (F02)",
                                            "PC (F03)","PC (F04)",
                                            "AFSC V (F05)","AFSC V (F06)",
                                            "AFSC VI (F07)","AFSC VI (F08)",
                                            "AFSC VII (F09)","AFSC VII (F10)",
                                            "AFSC VIII (F11)","AFSC VIII (F12)"))
```

Now we are going to import the relative abundances of all LAB together within the whole bacterial community. We will need the `bacteria_final_min` data frame from the previous section. In case that you are in a different R session, remember that we exported the data frame in a `csv` format, which we can actually open in Excel and save it as `xlsx`. In Excel I just created a second sheet with the grouped relative abundances of the main microbial groups found. You can have a look at the Excel if you want. We will use the relative abundance of LAB in every sample to add an extra layer of information to our plot.

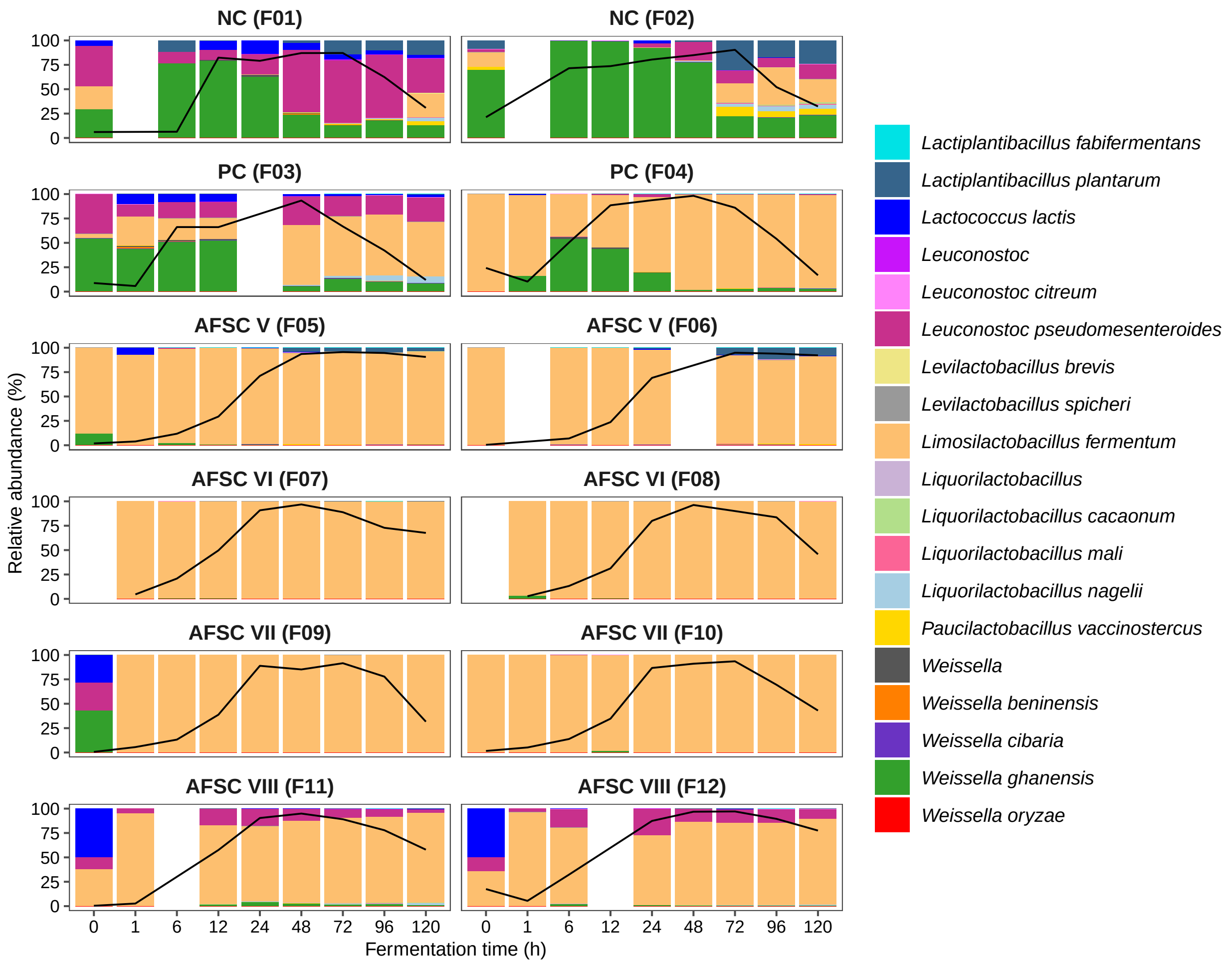
```r
library(readxl)
data <- read_excel("Test data/bacteria_genus_relative_abundance_pseudo.xlsx", sheet = 2)
data.lab <- data %>%
  filter(Taxon == "LAB")
data.t.lab <- as.data.frame(t(data.lab[2:ncol(data.lab)]))
colnames(data.t.lab) <- "Abundance"
```

In order to use two different data frames in the same `ggplot` object, we need to set the same x-axis and if we use `facet_wrap()`, the same divider. That's why we need the `Timepoint` and the `Vessel` columns, respectively.

```r
# Add Timepoint column
data.t.lab$Timepoint <- sapply(strsplit(basename(as.character(row.names(data.t.lab))),
                                        "T"), `[`,2)
data.t.lab$Timepoint <- as.numeric(as.character(data.t.lab$Timepoint))
data.t.lab <- data.t.lab %>%
  transform(data.t.lab, Timepoint = as.numeric(Timepoint))
data.t.lab <- data.t.lab[,-c(3:ncol(data.t.lab))]
# Add Vessel column
data.t.lab$Vessel <- c(rep("NC (F01)",8), rep("NC (F02)",8),
                       rep("PC (F03)",8), rep("PC (F04)",9),
                       rep("AFSC V (F05)",9), rep("AFSC V (F06)",7),
                       rep("AFSC VI (F07)",8), rep("AFSC VI (F08)",8),
                       rep("AFSC VII (F09)",9), rep("AFSC VII (F10)",9),
                       rep("AFSC VIII (F11)",8), rep("AFSC VIII (F12)",8))
# Sort the Vessel column in the same way as before
data.t.lab$Vessel <- factor(data.t.lab$Vessel,
                            levels = c("NC (F01)", "NC (F02)",
                                       "PC (F03)","PC (F04)",
                                       "AFSC V (F05)","AFSC V (F06)",
                                       "AFSC VI (F07)","AFSC VI (F08)",
                                       "AFSC VII (F09)","AFSC VII (F10)",
                                       "AFSC VIII (F11)","AFSC VIII (F12)"))
```

We can now proceed to plot our figure:

```
library(pals)
ggplot(lab_melted, aes(x=as.factor(Timepoint), y=value, fill=Taxon)) +

  geom_bar(stat="identity", position = position_stack()) +

  geom_line(data = data.t.lab, aes(y = Abundance, fill  = NULL, group = Vessel)) +

  facet_wrap(.~Vessel, ncol = 2) +

  theme_few() +

  theme(legend.title = element_blank(),
        legend.text = element_text(size=11, face = "italic"),
        legend.position = "right",
        panel.grid = element_blank(),
        axis.text.x = element_text(size = 10, colour = "black"),
        axis.text.y = element_text(size = 10, colour = "black"),
        axis.title = element_text(size=11),
        legend.key.height=unit(1.5,"line"),
        legend.key.width=unit(1.5,"line"),
        legend.text.align = 0,
        strip.text = element_text(face = "bold", size = 12)) +

  scale_fill_manual(values = as.vector(rev(cols25(20)))) +

  labs(title = element_blank(),
       y="Relative abundance (%)",
       x="Fermentation time (h)")
```



Amplicon sequence variant inspection of a species of interest (*strain-level*)

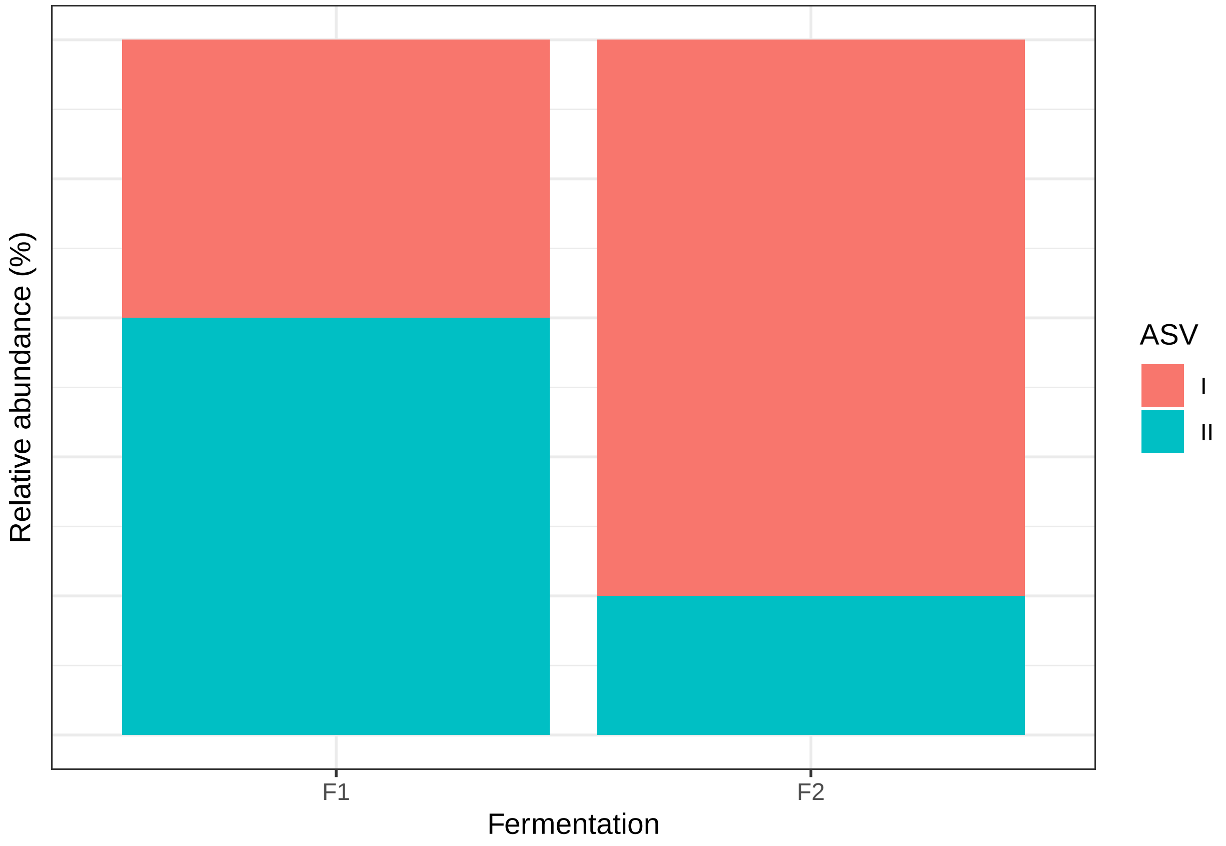
One of the most useful things of ASVs is the ability to inspect them at sub-species level. Every ASV is a unique sequence from a genome. It can be that a single, unique ASV is present in all genomes from a strain present in the microbial community. But there are far many more scenarios. To explain them I will not talk about strains, because that is a controversial terminology that can have many interpretations (see Van Rossum *et al.*, 2020) but I will talk about **populations** instead.

a. **An ASV represents a population within a species.** ASVs are made of rRNA genes in most of the cases, since they are conserved by kingdoms of life. Let's keep it to the bacterial 16S rRNA genes. Most bacterial genomes do not have a single 16S gene but, at least, a few of them. Whether there is only one 16S or 5 copies of the same 16S sequence, we will obtain only one ASV for that genome, and therefore, a population can be represented by one single ASV. This is the case for *Acetobacter pasteurianus*, for example. The strain that I inoculate in the cocoa fermentation processes gives only one ASV, since all 16S copies in its genome are identical.
b. **A few ASVs represent a population within a species.** It can also be that a single bacterial genome contains, let's say, 5 copies of the 16S rRNA and the 5 are different. In other words, they have small variations in their nucleotide sequence, most of the times consisting of single nucleotide

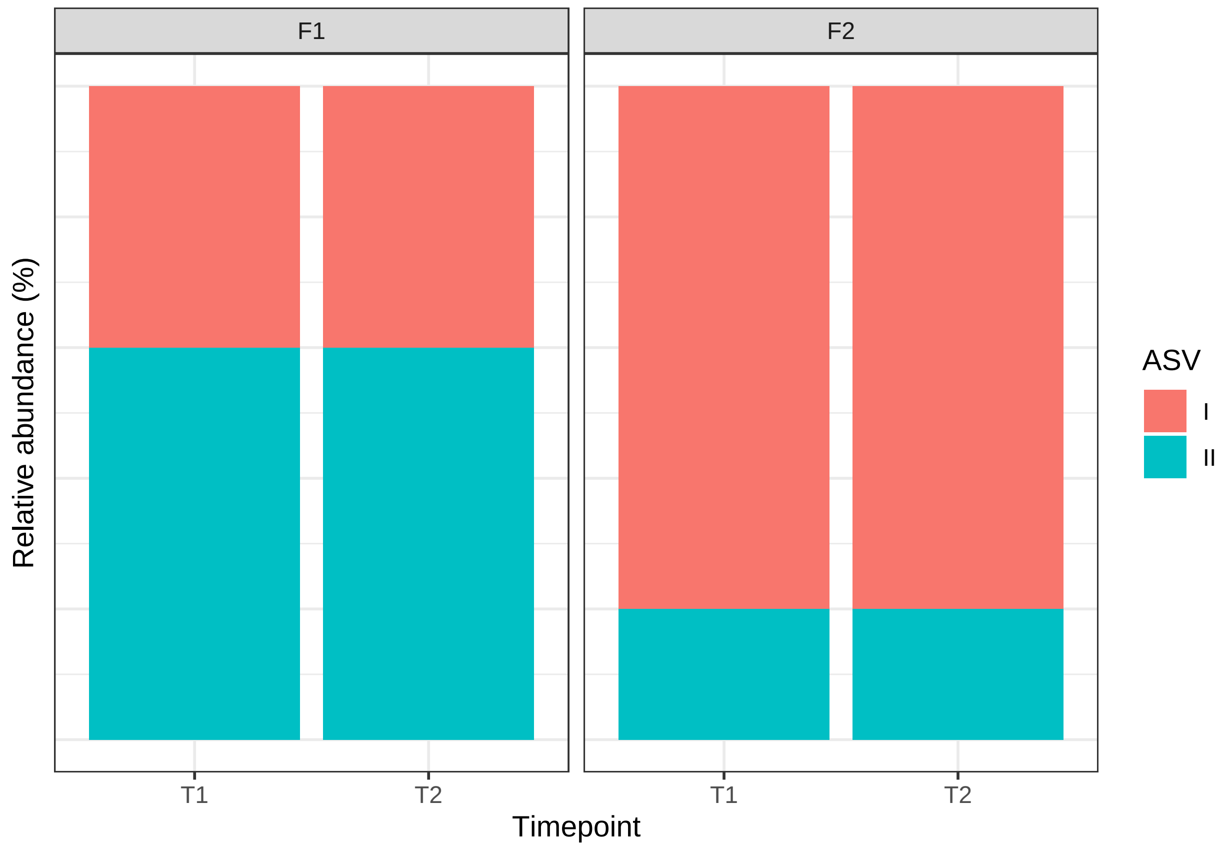
variants (SNVs). In that case, there will be 5 ASVs representing one population. This is the case for the *Liml. fermentum* that we inoculate. It has 5 copies and 4 of them are different, so this population is represented by 4 ASVs, which appear in this ratio of relative abundance (20%, 20%, 20%, 40%).

c. **An ASV represents more than one population.** We go back to the example of *A. pasteurianus.* I saw that the ASV appearing in spontaneous cocoa fermentation processes and the ones we inoculate was exactly the same. There was only one ASV. Here we have two possible scenario: 1. The spontaneous processes got contaminated with our strain. 2. The spontaneous population of *A. pasteurianus* from Costa Rica harbours the same 16S sequence than the inoculated strain (isolated from Ghana). In this case, several populations share the same ASV.

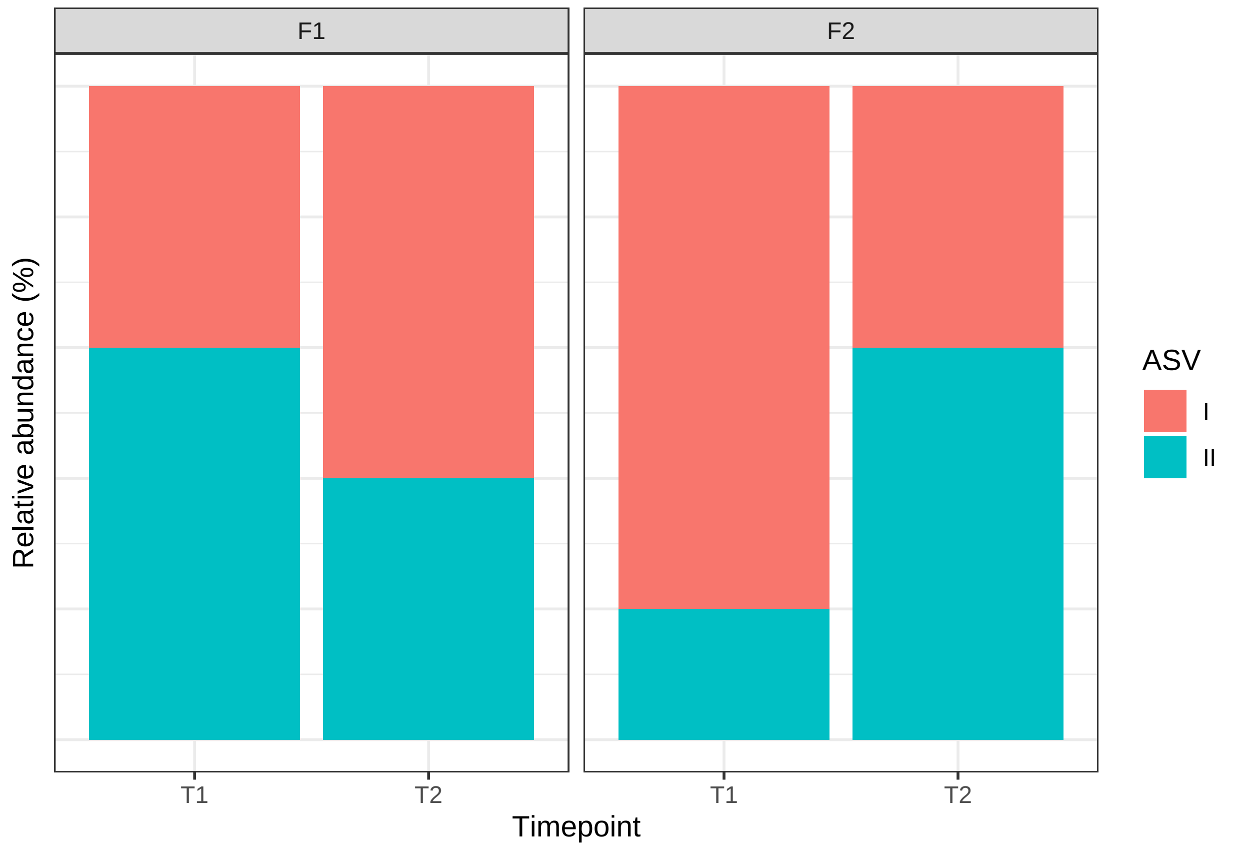
Let's see a real (complex) example. It is also possible that two distinct populations of a given species have the same 16S sequences, but in a different copy number. Let's assume a population of *Acteobacter lambici* with two distinct 16S sequences (I and II). This population has in total 5 copies, 2xI and 3xII. But there is also a *A. lambici* population with 5 copies of the 16S sequence and, interestingly, are also I and II. But in this case the second population has 4xI and 1xII. The ASV distribution of these two populations will be very distinguishable. At least if they are in different samples (see next figure). But let's be honest, if you see such case, you will think that the ASV I represents one *A. lambici* population and ASV II represents another. Unfortunately, in this case we cannot be sure what is the reality.



Unless. . . What if we are a time series in both fermentation processes (F1 and F2)? Then the chances of our initial hypothesis of two *A. lambici* populations having two ASVs is getting strength since it is very unlikely that the ratio of different populations is kept constant through a time series. But, the possibility stills there.



But what if we would have this different scenario? We see that the ratios change, that is an **absolute** certainty that ASV I represent a *A. lambici* population and ASV II a different one. Cool!



Conclusion: We cannot be sure if a single ASV or a group of ASVs are representing single strains/populations in a sample. But by sequencing a time series of a fermentation process, or comparing two different fermentations, we can differentiate populations. If we have fermentation A and fermentation B dominated both by *A. lambici*, by inspecting the ASVs we can see if they are the same or distinct strains/populations. But we cannot be sure about how many there are in each of them.

Uff, that was a lot of theory, let's go back to coding again. As done before, let's import all raw data from the beginning. Please be aware that the last line is slightly different than before.

```
bacteria_taxa <- read_tsv("Test data/Taxonomy_pseudo",col_names=TRUE)
bacteria_reads<- read_tsv("Test data/Reads_pseudo",col_names=TRUE)
bacteria <- merge(bacteria_taxa, bacteria_reads,
                  by="Sequence",
                  all=T)
bacteria <- bacteria[c(1:22,43,24,25,26,28,29,32:42,23,44:107,109,110,111)]
bacteria_genus<-bacteria[,c(1,7:ncol(bacteria))]
```

In this case, we are only going to select the species/genus that we are interested in. In this case it becomes easier for me, since all *Limosilactobacillus* are *fermentum*. But let's assume that there are more species, so that we also need to filter per species:

```
lf <- bacteria_genus %>%
  filter(Genus == "Limosilactobacillus") %>%
  filter(Species == "fermentum")
```

```
# We can output this table as we did with the genus table before
write_excel_csv2(lf, "Tables/Limosilactobacillus_fermentum.csv")
```

Work a bit on the data frame:

```
# Remove sequence column
lf_final <- lf[-1]
# Reformat the name
lf_final$Genus <- "Liml."
# Join genus and species information
lf_final$Genus <- paste(lf_final$Genus, lf_final$Species, row.names(lf_final))
# Remove species column
lf_renamed <- lf_final[,-2]
colnames(lf_renamed)[1]<-"ASV"
```

We don't need to group the ASVs or calculate minorities (unless you have like 50 ASVs), but is typically not the case. So let's proceed with all things we have done previously:

```
lf_relative <- percentage(lf_renamed)
lf_melted<- melt(lf_relative,id.vars="ASV")
lf_melted$Fermentation <- c(rep("Negative control",21*16),
                           rep("Positive control",21*17),
                           rep("AFSC V",21*16),
                           rep("AFSC VI",21*16),
                           rep("AFSC VII",21*18),
                           rep("AFSC VIII",21*16))
lf_melted$Vessel <- c(rep("NC (F01)",21*8), rep("NC (F02)",21*8),
                     rep("PC (F03)",21*8), rep("PC (F04)",21*9),
                     rep("AFSC V (F05)",21*9), rep("AFSC V (F06)",21*7),
                     rep("AFSC VI (F07)",21*8), rep("AFSC VI (F08)",21*8),
                     rep("AFSC VII (F09)",21*9), rep("AFSC VII (F10)",21*9),
                     rep("AFSC VIII (F11)",21*8), rep("AFSC VIII (F12)",21*8))
lf_melted$Timepoint <- sapply(strsplit(basename(as.character(lf_melted$variable)),
                                       "T"), `[`,2)
lf_melted$Timepoint <- as.numeric(as.character(lf_melted$Timepoint))
lf_melted <- lf_melted %>%
  transform(lf_melted, Timepoint = as.numeric(Timepoint))
lf_melted <- lf_melted[,-c(7:ncol(lf_melted))]
lf_melted$Vessel <- factor(lf_melted$Vessel,
                           levels = c("NC (F01)", "NC (F02)",
                                      "PC (F03)","PC (F04)",
                                      "AFSC V (F05)","AFSC V (F06)",
                                      "AFSC VI (F07)","AFSC VI (F08)",
                                      "AFSC VII (F09)","AFSC VII (F10)",
                                      "AFSC VIII (F11)","AFSC VIII (F12)"))
lf_melted$ASV <- factor(lf_melted$ASV,
                        levels = c("Liml. fermentum 1", "Liml. fermentum 2",
                                   "Liml. fermentum 3", "Liml. fermentum 4",
                                   "Liml. fermentum 5", "Liml. fermentum 6",
                                   "Liml. fermentum 7", "Liml. fermentum 8",
                                   "Liml. fermentum 9", "Liml. fermentum 10",
                                   "Liml. fermentum 11", "Liml. fermentum 12",
                                   "Liml. fermentum 13", "Liml. fermentum 14",
                                   "Liml. fermentum 15", "Liml. fermentum 16",
                                   "Liml. fermentum 17", "Liml. fermentum 18",
                                   "Liml. fermentum 19", "Liml. fermentum 20",
                                   "Liml. fermentum 21"))
```

Now we are going to import the relative abundances of all *Liml. fermentum* together within the whole bacterial community. Do as done before for the species-level section.

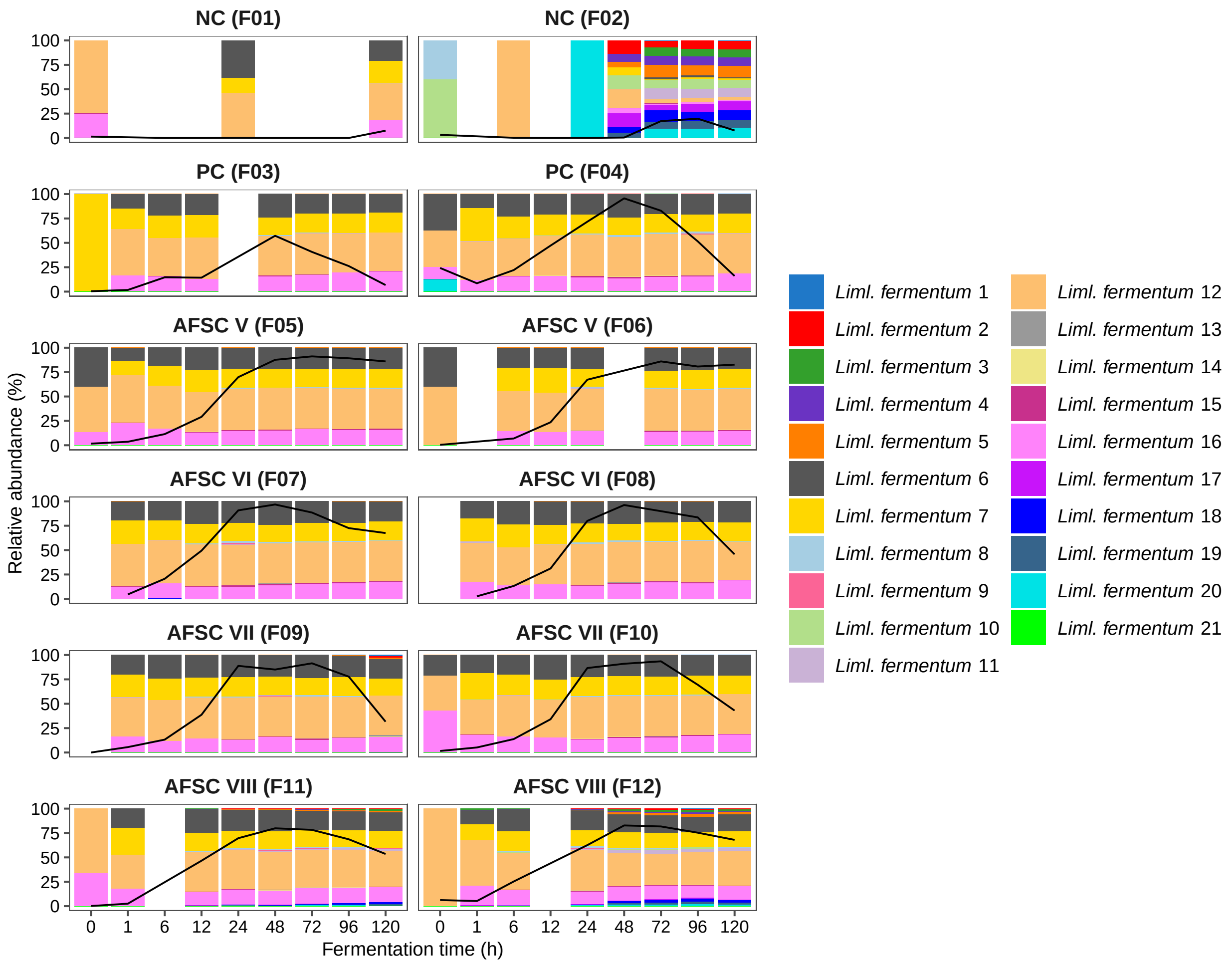
```
# Import relative abundances for LAB
data <- read_excel("Test data/bacteria_genus_relative_abundance_pseudo.xlsx")
data.lab <- data %>%
  filter(Taxon == "Limosilactobacillus")
data.t.lab <- as.data.frame(t(data.lab[2:ncol(data.lab)]))
colnames(data.t.lab) <- "Abundance"
# Add Timepoint column
data.t.lab$Timepoint <- sapply(strsplit(basename(as.character(row.names(data.t.lab))),
                                        "T"), `[`,2)
data.t.lab$Timepoint <- as.numeric(as.character(data.t.lab$Timepoint))
data.t.lab <- data.t.lab %>%
  transform(data.t.lab, Timepoint = as.numeric(Timepoint))
data.t.lab <- data.t.lab[,-c(3:ncol(data.t.lab))]
# Add Vessel column
data.t.lab$Vessel <- c(rep("NC (F01)",8), rep("NC (F02)",8),
                       rep("PC (F03)",8), rep("PC (F04)",9),
                       rep("AFSC V (F05)",9), rep("AFSC V (F06)",7),
                       rep("AFSC VI (F07)",8), rep("AFSC VI (F08)",8),
                       rep("AFSC VII (F09)",9), rep("AFSC VII (F10)",9),
                       rep("AFSC VIII (F11)",8), rep("AFSC VIII (F12)",8))
data.t.lab$Vessel <- factor(data.t.lab$Vessel,
                            levels = c("NC (F01)", "NC (F02)",
                                       "PC (F03)","PC (F04)",
                                       "AFSC V (F05)","AFSC V (F06)",
                                       "AFSC VI (F07)","AFSC VI (F08)",
                                       "AFSC VII (F09)","AFSC VII (F10)",
                                       "AFSC VIII (F11)","AFSC VIII (F12)"))
```

Let's add this ugly and long code to put the names in *italic* but not the numbers:

```
mylabels <- c(expression(paste(italic("Liml. fermentum"), " 1")),
              expression(paste(italic("Liml. fermentum"), " 2")),
              expression(paste(italic("Liml. fermentum"), " 3")),
              expression(paste(italic("Liml. fermentum"), " 4")),
              expression(paste(italic("Liml. fermentum"), " 5")),
              expression(paste(italic("Liml. fermentum"), " 6")),
              expression(paste(italic("Liml. fermentum"), " 7")),
              expression(paste(italic("Liml. fermentum"), " 8")),
              expression(paste(italic("Liml. fermentum"), " 9")),
              expression(paste(italic("Liml. fermentum"), " 10")),
              expression(paste(italic("Liml. fermentum"), " 11")),
              expression(paste(italic("Liml. fermentum"), " 12")),
              expression(paste(italic("Liml. fermentum"), " 13")),
              expression(paste(italic("Liml. fermentum"), " 14")),
              expression(paste(italic("Liml. fermentum"), " 15")),
              expression(paste(italic("Liml. fermentum"), " 16")),
              expression(paste(italic("Liml. fermentum"), " 17")),
              expression(paste(italic("Liml. fermentum"), " 18")),
              expression(paste(italic("Liml. fermentum"), " 19")),
              expression(paste(italic("Liml. fermentum"), " 20")),
              expression(paste(italic("Liml. fermentum"), " 21")))
```

Let's plot!

```
ggplot(lf_melted, aes(x=as.factor(Timepoint),y=value,fill=ASV)) +

  geom_bar(stat="identity", position = position_stack()) +

  geom_line(data = data.t.lab, aes(y = Abundance, fill  = NULL, group = Vessel)) +

  facet_wrap(.~Vessel, ncol = 2) +

  theme_few() +

  theme(legend.title = element_blank(),
        legend.text = element_text(size=11),
        legend.position = "right",
        panel.grid = element_blank(),
        axis.text.x = element_text(size = 10, colour = "black"),
        axis.text.y = element_text(size = 10, colour = "black"),
        axis.title = element_text(size=11)) +

  theme(legend.key.height=unit(1.5,"line"),
        legend.key.width=unit(1.5,"line"),
        legend.text.align = 0,
        strip.text = element_text(face = "bold", size = 12)) +

  scale_fill_manual(values = as.vector(cols25(21)),
                    labels = mylabels) +

  labs(title = element_blank(),
       y="Relative abundance (%)",
       x="Fermentation time (h)")
```



NC (F01)
NC (F02)
PC (F03)
PC (F04)
AFSC V (F05)
AFSC V (F06)
AFSC VI (F07)
AFSC VI (F08)
AFSC VII (F09)
AFSC VII (F10)
AFSC VIII (F11)
AFSC VIII (F12)
100
75
50
25
0
Relative abundance (%)
0
1
6
12
24
48
72
96
120
Fermentation time (h)
Liml. fermentum 1
Liml. fermentum 2
Liml. fermentum 3
Liml. fermentum 4
Liml. fermentum 5
Liml. fermentum 6
Liml. fermentum 7
Liml. fermentum 8
Liml. fermentum 9
Liml. fermentum 10
Liml. fermentum 11
Liml. fermentum 12
Liml. fermentum 13
Liml. fermentum 14
Liml. fermentum 15
Liml. fermentum 16
Liml. fermentum 17
Liml. fermentum 18
Liml. fermentum 19
Liml. fermentum 20
Liml. fermentum 21