Taxa Plot Tutorial

Laura Schaerer

2/23/2023

Taxa Plot Tutorial

This tutorial shows you how to make a taxa plot from a phyloseq object.

Start by loading the required R packages

(Packages must be installed prior to running this tutorial)

```
library(tidyverse)
```

```
## ── Attaching packages ─────────────────────────────────────── tidyverse 1.3.2 ──
## ✔ ggplot2 3.4.0     ✔ purrr   1.0.1
## ✔ tibble  3.1.8     ✔ dplyr   1.1.0
## ✔ tidyr   1.3.0     ✔ stringr 1.5.0
## ✔ readr   2.1.3     ✔ forcats 1.0.0
## ── Conflicts ────────────────────────────────────────── tidyverse_conflicts() ──
## ✖ dplyr::filter() masks stats::filter()
## ✖ dplyr::lag()    masks stats::lag()
```

```
library(phyloseq)
```

Load your phyloseq object

This tutorial assumes that you have a phyloseq object of the data that you want to plot. The example phyloseq object shown here has 9 samples, 9 sample variables, and 12,003 unique taxa.

```
toy_ps <- read_rds("/Users/lgschaer/Desktop/toy_phyloseq.rds")
toy_ps
```

```
## phyloseq-class experiment-level object
## otu_table()   OTU Table:         [ 12003 taxa and 9 samples ]
## sample_data() Sample Data:       [ 9 samples by 9 sample variables ]
## tax_table()   Taxonomy Table:    [ 12003 taxa by 7 taxonomic ranks ]
```

Explore the sample variables in the test phyloseq object

This tutorial assumes that you have a phyloseq object of the data that you want to plot. The example phyloseq object shown here has 9 samples, 9 sample variables, and 12,003 unique taxa.

```
head(sample_data(toy_ps))
```

```
##          SampleID Environment Sample_Type Replicate Substrate Transfer
## CCONR1T1 CCONR1T1     Compost  Enrichment        R1   Control       T1
## CCONR2T1 CCONR2T1     Compost  Enrichment        R2   Control       T1
## CCONR3T1 CCONR3T1     Compost  Enrichment        R3   Control       T1
## CTAR1T1   CTAR1T1     Compost  Enrichment        R1        TA       T1
## CTAR2T1   CTAR2T1     Compost  Enrichment        R2        TA       T1
## CTAR3T1   CTAR3T1     Compost  Enrichment        R3        TA       T1
##          Substrate_Label Environment_Label_Location Environment_Label
## CCONR1T1         Control       Compost\nCalumet, MI           Compost
## CCONR2T1         Control       Compost\nCalumet, MI           Compost
## CCONR3T1         Control       Compost\nCalumet, MI           Compost
## CTAR1T1  Terephthalamide       Compost\nCalumet, MI           Compost
## CTAR2T1  Terephthalamide       Compost\nCalumet, MI           Compost
## CTAR3T1  Terephthalamide       Compost\nCalumet, MI           Compost
```

STEP #1: Convert the phyloseq object into a dataframe that can be more easily manipulated

The tax_glom() function specifies the most specific taxonomic level you are interested in. Change the taxonomic level to the most specific level you are interested in plotting. The more broad the category, the quicker the code will run. This step will probably run slowly for a large data set.

```
genusabundance <- toy_ps %>%
  tax_glom(taxrank = "Genus") %>%                      # agglomerate at class level
  transform_sample_counts(function(x) {x/sum(x)} ) %>%   # Transform to rel. abundance
  psmelt() %>%                                           # Melt to long format
  arrange(Genus)
head(genusabundance)
```

```
##       OTU   Sample Abundance SampleID Environment Sample_Type Replicate
## 1 sp10420  CTPR3T1         0  CTPR3T1     Compost  Enrichment        R3
## 2 sp10420  CTPR1T1         0  CTPR1T1     Compost  Enrichment        R1
## 3 sp10420  CTAR3T1         0  CTAR3T1     Compost  Enrichment        R3
## 4 sp10420  CTAR2T1         0  CTAR2T1     Compost  Enrichment        R2
## 5 sp10420  CTAR1T1         0  CTAR1T1     Compost  Enrichment        R1
## 6 sp10420 CCONR1T1         0 CCONR1T1     Compost  Enrichment        R1
##   Substrate Transfer Substrate_Label Environment_Label_Location
## 1       TPA       T1   Terephthalate       Compost\nCalumet, MI
## 2       TPA       T1   Terephthalate       Compost\nCalumet, MI
## 3        TA       T1 Terephthalamide       Compost\nCalumet, MI
## 4        TA       T1 Terephthalamide       Compost\nCalumet, MI
## 5        TA       T1 Terephthalamide       Compost\nCalumet, MI
## 6   Control       T1         Control       Compost\nCalumet, MI
##   Environment_Label  Kingdom      Phylum           Class             Order
## 1           Compost Bacteria Chloroflexi Ktedonobacteria Ktedonobacterales
## 2           Compost Bacteria Chloroflexi Ktedonobacteria Ktedonobacterales
## 3           Compost Bacteria Chloroflexi Ktedonobacteria Ktedonobacterales
## 4           Compost Bacteria Chloroflexi Ktedonobacteria Ktedonobacterales
## 5           Compost Bacteria Chloroflexi Ktedonobacteria Ktedonobacterales
## 6           Compost Bacteria Chloroflexi Ktedonobacteria Ktedonobacterales
##               Family  Genus
## 1 Ktedonobacteraceae 1959-1
## 2 Ktedonobacteraceae 1959-1
## 3 Ktedonobacteraceae 1959-1
## 4 Ktedonobacteraceae 1959-1
## 5 Ktedonobacteraceae 1959-1
## 6 Ktedonobacteraceae 1959-1
```

STEP #2: Filter, group and modify data to prepare for plotting.

Here we will make a simplified data frame that only includes the variables we are interested in plotting for all taxonomic levels.

select() function: Choose which variables to include in the plot. Select ALL variables that will be used for x-axis and/or facet labels in addition to all taxonomic levels you are interested in.

filter() function: Filter out all taxa with zero percent abundance. Filter can also be used to remove treatments or conditions that are not to be included in the plot.

mutate() function: Convert the taxonomic level columns to character vectors (they are factors by default), can also be used to add additional columns or modify existing columns as desired.

```
all <- genusabundance %>%
  select(Phylum, Class, Family, Genus, Sample, Abundance, Substrate, Substrate_Label, Re
plicate) %>%
  filter(Abundance != 0) %>%
  mutate(
    Phylum = as.character(Phylum),
    Class = as.character(Class),
    Family = as.character(Family),
    Genus = as.character(Genus))
head(all)
```

```
##             Phylum               Class             Family           Genus
## 1 Abditibacteriota      Abditibacteria Abditibacteriaceae Abditibacterium
## 2       Firmicutes             Bacilli Acholeplasmataceae    Acholeplasma
## 3       Firmicutes             Bacilli Acholeplasmataceae    Acholeplasma
## 4   Proteobacteria Gammaproteobacteria     Alcaligenaceae   Achromobacter
## 5   Proteobacteria Gammaproteobacteria     Alcaligenaceae   Achromobacter
## 6   Proteobacteria Gammaproteobacteria     Alcaligenaceae   Achromobacter
##    Sample    Abundance Substrate Substrate_Label Replicate
## 1 CTAR1T1 0.0008406894        TA Terephthalamide        R1
## 2 CTPR1T1 0.0008461600       TPA   Terephthalate        R1
## 3 CTPR2T1 0.0006498781       TPA   Terephthalate        R2
## 4 CTPR3T1 0.0019787468       TPA   Terephthalate        R3
## 5 CTPR2T1 0.0010289737       TPA   Terephthalate        R2
## 6 CTAR3T1 0.0003358522        TA Terephthalamide        R3
```

STEP #3: Prepare to plot Phylum

Here we will group the data even more to prepare to make a taxa plot at the Phylum level

group_by() function: Groups data by the specified variables. Include ONLY the variables that will be used in the ggplot() command to make the plot. Adding additional variables here can lead to aesthetic problems with the plot.

summarise() function: This function will collapse the data based on the groups specified in the group_by() function above. Here we will calculate the average abundance for each group (sum of the abundance of each phyla for each unique substrate/replicate combination divided by the total number of samples in each group).

```
phylum <- all %>%
  select(Substrate, Substrate_Label, Replicate, Phylum, Abundance) %>%  #choose variable
s to work with
  group_by(Substrate, Substrate_Label, Replicate) %>%                   #group by variab
les used to plot NOT taxonomic level
  mutate(totalSum = sum(Abundance)) %>%                                 #calculate total
abundance of each Phylum
  ungroup() %>%                                                         #remove grouping
variables
  group_by(Substrate, Substrate_Label, Replicate, Phylum) %>%           #now group by sa
me variables as before PLUS taxonomic level
  summarise(
    Abundance = sum(Abundance),                                         #sum abundance i
n each phylum for each unique group of variables
    totalSum,
    RelAb = Abundance/totalSum) %>%                                     #calculate relat
ive abundance
  unique()
```

```
## Warning: Returning more (or less) than 1 row per `summarise()` group was deprecated i
n
## dplyr 1.1.0.
## ℹ Please use `reframe()` instead.
## ℹ When switching from `summarise()` to `reframe()`, remember that `reframe()`
##   always returns an ungrouped data frame and adjust accordingly.
```

```
## `summarise()` has grouped output by 'Substrate', 'Substrate_Label',
## 'Replicate', 'Phylum'. You can override using the `.groups` argument.
```

```
head(phylum)
```

```
## # A tibble: 6 × 7
## # Groups:   Substrate, Substrate_Label, Replicate, Phylum [6]
##   Substrate Substrate_Label Replicate Phylum           Abundance total…¹   RelAb
##   <chr>     <chr>           <chr>     <chr>                <dbl>   <dbl>   <dbl>
## 1 Control   Control         R1        Acidobacteriota    0.0275        1 0.0275
## 2 Control   Control         R1        Actinobacteriota   0.0422        1 0.0422
## 3 Control   Control         R1        Bacteroidota       0.446         1 0.446
## 4 Control   Control         R1        Bdellovibrionota   0.00336       1 0.00336
## 5 Control   Control         R1        Chloroflexi        0.00397       1 0.00397
## 6 Control   Control         R1        Crenarchaeota      0.00275       1 0.00275
## # … with abbreviated variable name ¹totalSum
```

Check data: maximum should be no greater than 1, minimum should be more than zero

```
max(phylum$RelAb)
```

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

```
mean(phylum$RelAb)
```

```
## [1] 0.08108108
```

```
min(phylum$RelAb)
```

```
## [1] 8.058667e-05
```

STEP #4: Prepare colors for the plot

Calculate how many colors will be needed. This is equal to the length of the unique values in the Phylum column of the “phylum” dataframe (18).

Save a vector of color names to be used for plotting.

Check the length to be sure there are enough colors… Here we have 25 colors, more than enough!

```
length(unique(phylum$Phylum))
```

```
## [1] 18
```

```
phylum_colors <- c(
  "grey22", "darkcyan", "orchid1", "green", "orange", "blue", "tomato2", "olivedrab", "g
rey47",
  "cyan", "coral3", "darkgreen", "magenta", "palegoldenrod", "dodgerblue", "firebrick",
"yellow", "purple4",
  "lightblue", "grey77", "mediumpurple1", "tan4", "red", "darkblue", "yellowgreen")

length(phylum_colors)
```

```
## [1] 25
```

STEP #5: Make a plot!

geom_col(): this creates the columns in the plot. Bars should add to exactly 1, if not check grouping variables. If bars add exactly to 1, percentages from the dataframe “phylum” can be interpreted directly as relative abundances. Anything inside aes() will create a legend: Fill = Phylum Anything outside outside aes() will NOT create a legend: Color = “black”: creates black outline around colored blocks. If more than one block of the same color, check grouping variables in Step # 3.

facet_grid(): divides the figure into panels based on the variables provided.

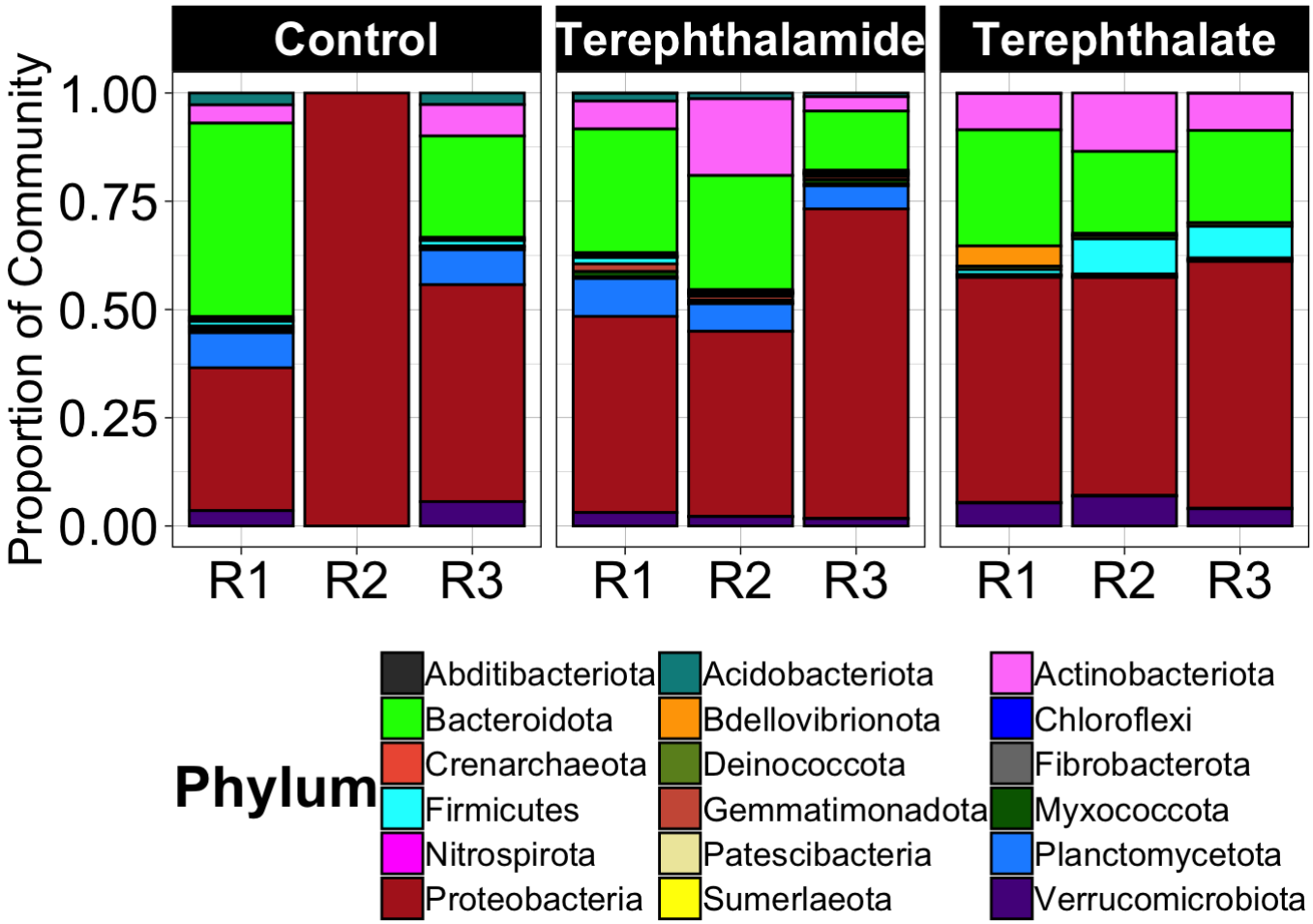
scale_fill_manual(): specifies vector of colors to use instead of default colors.

theme_linedraw(): sets theme for graphic. Additional themes are available.

theme(): aesthetic tweaks to appearance, font size, etc.

guides(): adjusts legend

```
ggplot(phylum)+
  geom_col(mapping = aes(x = Replicate, y = RelAb, fill = Phylum), color = "black", posi
tion = "stack", show.legend = TRUE)+
  facet_grid(cols = vars(Substrate_Label))+
  ylab("Proportion of Community") +
  xlab(NULL)+
  scale_fill_manual(values = phylum_colors) +
  theme_linedraw()+
  theme(axis.text.y = element_text(size = 20, color = "black"),
        axis.title.y = element_text(size = 18, color = "black"),
        axis.text.x = element_text(size = 20, angle = 0, vjust = 1, hjust = 0.5, color =
"black"),
        legend.text = element_text(size = 13),
        legend.position = "bottom",
        legend.spacing.x = unit(0.1, 'mm'),
        legend.spacing.y = unit(0.05, 'mm'),
        plot.margin=grid::unit(c(0.1,0.1,0.1,0.1), "mm"),
        strip.text = element_text(size = 18, face = "bold", angle = 0),
        legend.title = element_text(face="bold", size = 22))+
  guides(fill=guide_legend(ncol=3,byrow=TRUE))
```



STEP #6: Plotting other taxonomic levels, grouping low abundance taxa.

Plotting other taxonomic levels is very similar, starting with the “all” dataframe we created earlier, modify the grouping variables to desired taxonomic level. This example will be grouped by genus.

```
genus <- all %>%
  select(Substrate, Substrate_Label, Replicate, Genus, Abundance) %>%
  group_by(Substrate, Substrate_Label, Replicate) %>%
  mutate(totalSum = sum(Abundance)) %>%
  ungroup() %>%
  group_by(Substrate, Substrate_Label, Replicate, Genus) %>%
  summarise(
    Abundance = sum(Abundance),
    totalSum,
    RelAb = Abundance/totalSum) %>%
  unique()
```

```
## `summarise()` has grouped output by 'Substrate', 'Substrate_Label',
## 'Replicate'. You can override using the `.groups` argument.
```

```
head(genus)
```

```
## # A tibble: 6 × 7
## # Groups:   Substrate, Substrate_Label, Replicate [1]
##   Substrate Substrate_Label Replicate Genus              Abund…¹ total…²   RelAb
##   <chr>     <chr>           <chr>     <chr>                <dbl>   <dbl>   <dbl>
## 1 Control   Control         R1        Adhaeribacter      0.0107         1 0.0107
## 2 Control   Control         R1        Aeromicrobium      0.00122        1 0.00122
## 3 Control   Control         R1        Ahniella           0.00458        1 0.00458
## 4 Control   Control         R1        Algoriphagus       0.00397        1 0.00397
## 5 Control   Control         R1        Altererythrobacter 0.0131         1 0.0131
## 6 Control   Control         R1        Amaricoccus        0.00519        1 0.00519
## # … with abbreviated variable names ¹Abundance, ²totalSum
```

Check distribution of the data, these should all be between 1 and 0. Here we also check the number of unqiue genera, there are 230! It would be impossible to visualize 230 colors in a figure.

```
max(genus$RelAb)
```

```
## [1] 0.6666667
```

```
mean(genus$RelAb)
```

```
## [1] 0.01092233
```

```
min(genus$RelAb)
```

```
## [1] 5.597537e-05
```

```
length(unique(genus$Genus))
```

```
## [1] 230
```

One solution for this is to group low abundance genera into a single category. Here we add this to our grouping of the dataframe “all” created earlier. Here I am grouping all genera present at less than 5% relative abundance into a single group.

Using mutate() paired with ifelse() allows us to group any taxa with low (< 5% relative abundance) into a single category, reducing the number of colors needed.

```
genus <- all %>%
  select(Substrate, Substrate_Label, Replicate, Genus, Abundance) %>%
  group_by(Substrate, Substrate_Label, Replicate) %>%
  mutate(totalSum = sum(Abundance)) %>%
  ungroup() %>%
  group_by(Substrate, Substrate_Label, Replicate, Genus, totalSum) %>%
  summarise(
    Abundance = sum(Abundance),
    Genus = ifelse(Abundance < 0.05, "< 5 %", Genus)) %>%               #change Genus la
bel to group low abundance taxa together
  group_by(Substrate, Substrate_Label, Replicate, Genus, totalSum) %>%  #now group and s
ummarize again to group newly labeled low abundance taxa together
  summarise(
    Abundance = sum(Abundance),
    RelAb = Abundance/totalSum) %>%
  unique()
```

```
## `summarise()` has grouped output by 'Substrate', 'Substrate_Label',
## 'Replicate', 'Genus'. You can override using the `.groups` argument.
```

```
## Warning: Returning more (or less) than 1 row per `summarise()` group was deprecated i
n
## dplyr 1.1.0.
## ℹ Please use `reframe()` instead.
## ℹ When switching from `summarise()` to `reframe()`, remember that `reframe()`
##   always returns an ungrouped data frame and adjust accordingly.
```

```
## `summarise()` has grouped output by 'Substrate', 'Substrate_Label',
## 'Replicate', 'Genus', 'totalSum'. You can override using the `.groups`
## argument.
```

```
head(genus)
```

```
## # A tibble: 6 × 7
## # Groups:   Substrate, Substrate_Label, Replicate, Genus, totalSum [6]
##   Substrate Substrate_Label Replicate Genus          totalSum Abundance  RelAb
##   <chr>     <chr>           <chr>     <chr>             <dbl>     <dbl>  <dbl>
## 1 Control   Control         R1        < 5 %                 1    0.723  0.723
## 2 Control   Control         R1        Edaphobaculum         1    0.226  0.226
## 3 Control   Control         R1        Hassallia             1    0.0501 0.0501
## 4 Control   Control         R2        Ochrobactrum          1    0.667  0.667
## 5 Control   Control         R2        Steroidobacter        1    0.333  0.333
## 6 Control   Control         R3        < 5 %                 1    0.713  0.713
```

Now check the number of unique genera we have… 17 unique genera! Much easier to visualize!

```
length(unique(genus$Genus))
```

```
## [1] 17
```

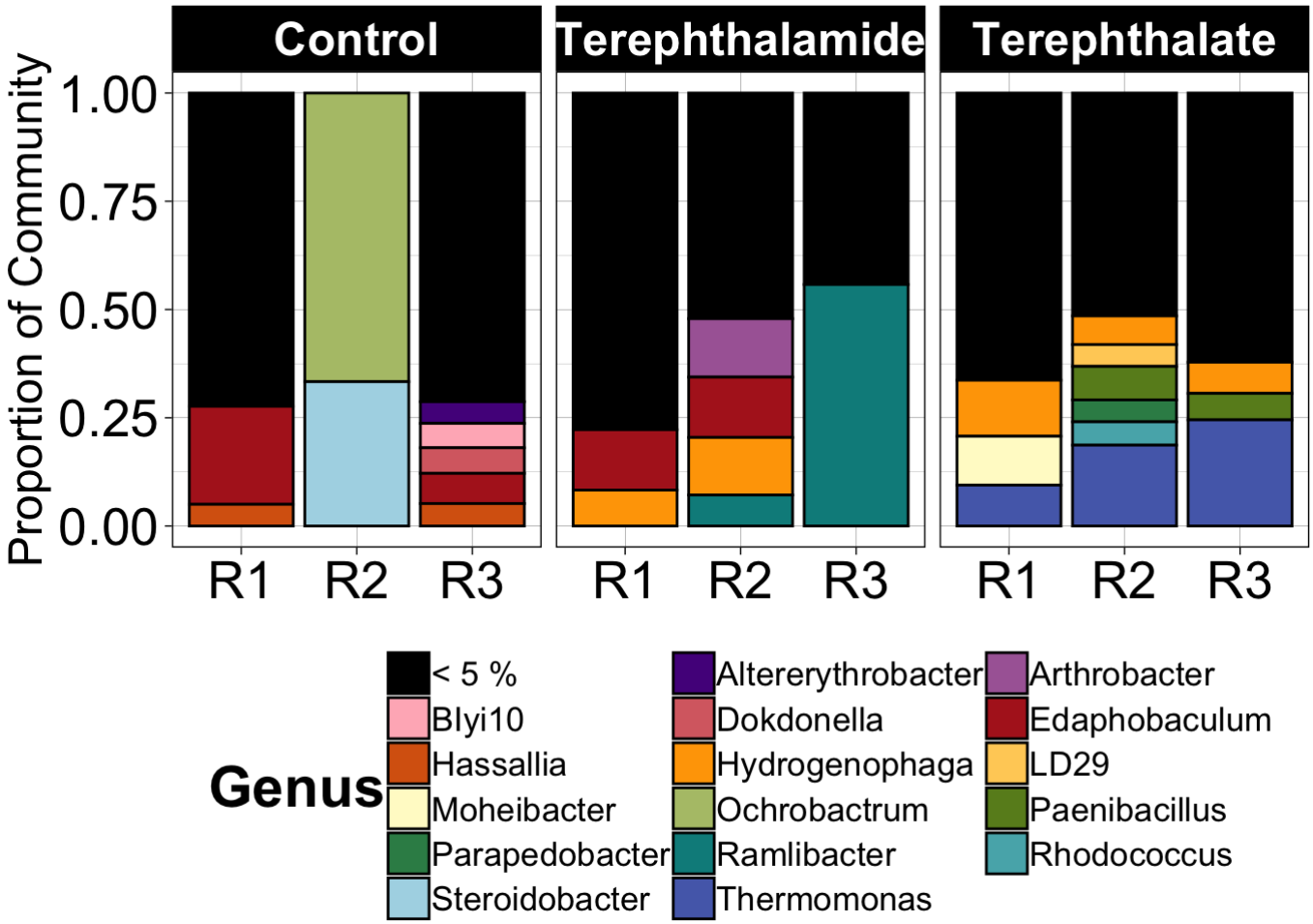
One way to get a long vector of colors is to use colorRampPalette to make a color palette based on a few supplied colors.

```
colFunc <- colorRampPalette(c("purple4", "lightpink", "firebrick", "orange", "lemonchiff
on", "olivedrab4", "darkcyan", "lightblue", "darkblue"))
color_list <- colFunc(length(unique(genus$Genus)))
genus_colors <- c("black", color_list)
length(genus_colors)
```

```
## [1] 18
```

Now we can plot the data! This same process can be used to plot any taxonomic level where there are too many unique genera to visualize all of them.

```
ggplot(genus)+
  geom_col(mapping = aes(x = Replicate, y = RelAb, fill = Genus), color = "black", posit
ion = "stack", show.legend = TRUE)+
  facet_grid(cols = vars(Substrate_Label))+
  ylab("Proportion of Community") +
  xlab(NULL)+
  scale_fill_manual(values = genus_colors) +
  theme_linedraw()+
  theme(axis.text.y = element_text(size = 20, color = "black"),
        axis.title.y = element_text(size = 18, color = "black"),
        axis.text.x = element_text(size = 20, angle = 0, vjust = 1, hjust = 0.5, color =
"black"),
        legend.text = element_text(size = 13),
        legend.position = "bottom",
        legend.spacing.x = unit(0.1, 'mm'),
        legend.spacing.y = unit(0.05, 'mm'),
        plot.margin=grid::unit(c(0.1,0.1,0.1,0.1), "mm"),
        strip.text = element_text(size = 18, face = "bold", angle = 0),
        legend.title = element_text(face="bold", size = 22))+
  guides(fill=guide_legend(ncol=3,byrow=TRUE))
```



Note that the `echo = FALSE` parameter was added to the code chunk to prevent printing of the R code that generated the plot.