Análisis de diversidad metagenómica

Dulce I. Valdivia

Octubre 2022

En este manual se realizarán análisis básicos de diversidad y abundancia de muestras de metagenómica. Para poder llevar a acabo todos los pasos es necesario:

- **Datos**:
  - *Archivo de asignación taxonómica.* Archivo `.json` generado en el últmo paso del pipeline de metagenómica. Este archivo es un parseo del programa `kraken-biom` a la salida de asignación taxonómica generados por kraken (`.kraken.report`)
- **Paquetes de R**:
  - `phyloseq`: contiene las funciones necesarias para realizar los análisis correspondientes.
  - `tidyverse`: manipulación de datos.
  - `ggplot2` y `scico`: para hacer los gráficos de dichos análisis.

  A continuación se muestra cómo hacer la instalación de estos paquetes:

```
if (!require("BiocManager", quietly = TRUE))
  install.packages("BiocManager")
BiocManager::install("phyloseq")
install.packages("ggplot2")
install.packages("scico")
install.packages("tidyverse")
```

Carga de datos

Iniciamos carando los paquetes que necesitaremos:

```
library(phyloseq)
library(ggplot2)
library(tidyverse)
library(scico)
```

Cargamos la asignación taxonómica generada:

```
taxonomy <- import_biom("taxonomy_kraken.json")
```

Exploración inicial y limpieza de datos

Checamos el contenido del objeto taxonomy:

```
# Información general:
  taxonomy
```

```
## phyloseq-class experiment-level object
## otu_table()   OTU Table:         [ 7831 taxa and 3 samples ]
## sample_data() Sample Data:       [ 3 samples by 1 sample variables ]
```

```
## tax_table()   Taxonomy Table:    [ 7831 taxa by 7 taxonomic ranks ]
```

```
# Conteo de taxa por muestra:
  taxonomy@otu_table %>% head()
```

```
## OTU Table:          [6 taxa and 3 samples]
##                      taxa are rows
##       SRR11131028.kraken SRR11131029.kraken SRR11131030.kraken
## 38820               6766                 13               4606
## 4479              386519                712             286578
## 4577            15943012                655            9943380
## 4558               67788               8600              50759
## 4539                9239                 12               6524
## 38727             179296                169             111688
```

```
# Descripción de taxones:
  taxonomy@tax_table %>% head()
```

```
## Taxonomy Table:     [6 taxa by 7 taxonomic ranks]:
##       Rank1          Rank2             Rank3              Rank4
## 38820 "k__Eukaryota" "p__Streptophyta" "c__Magnoliopsida" "o__Poales"
## 4479  "k__Eukaryota" "p__Streptophyta" "c__Magnoliopsida" "o__Poales"
## 4577  "k__Eukaryota" "p__Streptophyta" "c__Magnoliopsida" "o__Poales"
## 4558  "k__Eukaryota" "p__Streptophyta" "c__Magnoliopsida" "o__Poales"
## 4539  "k__Eukaryota" "p__Streptophyta" "c__Magnoliopsida" "o__Poales"
## 38727 "k__Eukaryota" "p__Streptophyta" "c__Magnoliopsida" "o__Poales"
##       Rank5        Rank6        Rank7
## 38820 "f__"        "g__"        "s__"
## 4479  "f__Poaceae" "g__"        "s__"
## 4577  "f__Poaceae" "g__Zea"     "s__mays"
## 4558  "f__Poaceae" "g__Sorghum" "s__bicolor"
## 4539  "f__Poaceae" "g__Panicum" "s__"
## 38727 "f__Poaceae" "g__Panicum" "s__virgatum"
```

```
  # O bien:
  taxonomy@tax_table@.Data %>% head()
```

```
##       Rank1          Rank2             Rank3              Rank4
## 38820 "k__Eukaryota" "p__Streptophyta" "c__Magnoliopsida" "o__Poales"
## 4479  "k__Eukaryota" "p__Streptophyta" "c__Magnoliopsida" "o__Poales"
## 4577  "k__Eukaryota" "p__Streptophyta" "c__Magnoliopsida" "o__Poales"
## 4558  "k__Eukaryota" "p__Streptophyta" "c__Magnoliopsida" "o__Poales"
## 4539  "k__Eukaryota" "p__Streptophyta" "c__Magnoliopsida" "o__Poales"
## 38727 "k__Eukaryota" "p__Streptophyta" "c__Magnoliopsida" "o__Poales"
##       Rank5        Rank6        Rank7
## 38820 "f__"        "g__"        "s__"
## 4479  "f__Poaceae" "g__"        "s__"
## 4577  "f__Poaceae" "g__Zea"     "s__mays"
## 4558  "f__Poaceae" "g__Sorghum" "s__bicolor"
## 4539  "f__Poaceae" "g__Panicum" "s__"
## 38727 "f__Poaceae" "g__Panicum" "s__virgatum"
```

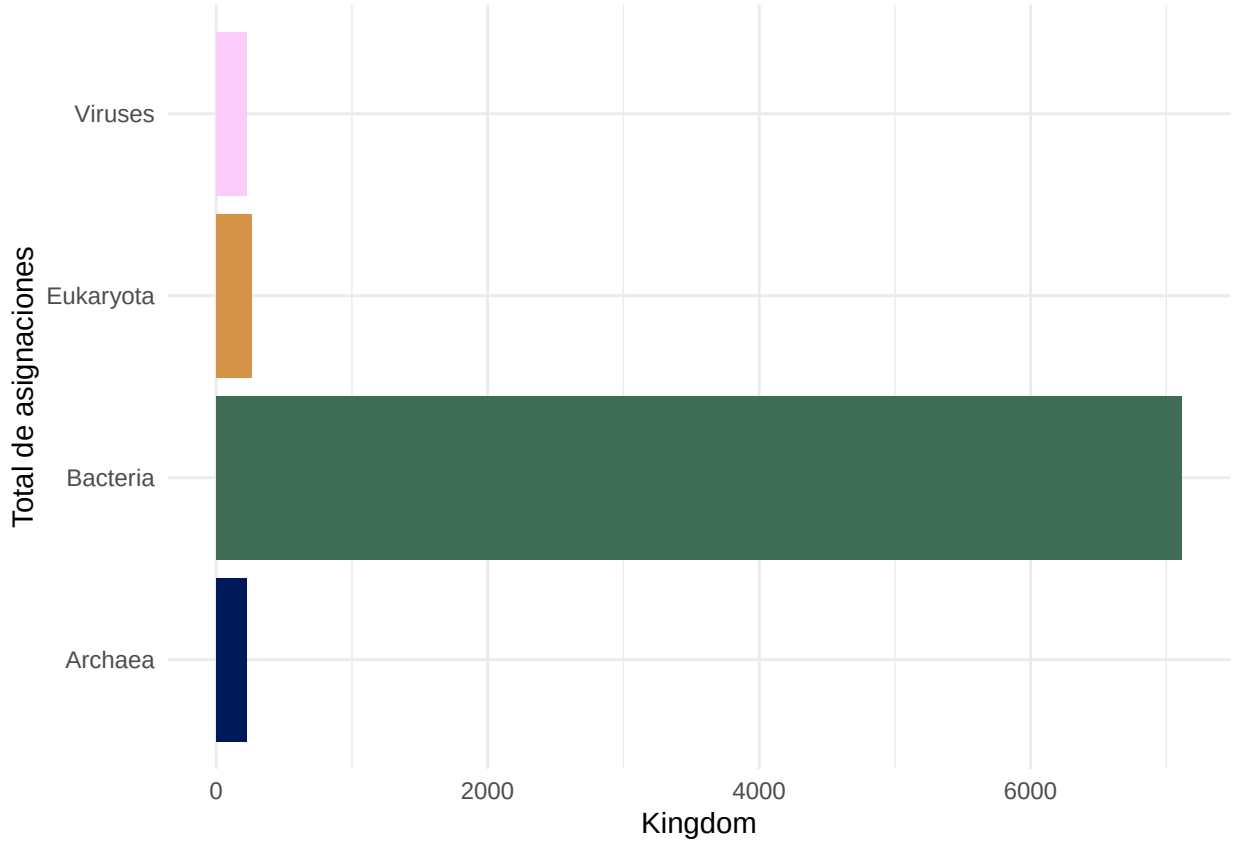
Vamos a limpiar los datos de dos maneras distintas. Primero, como observamos en la exploración del objeto tax_table, los taxones tienen al inicio una etiqueta de cuatro caracteres que corresponde al rango taxonómico al que corresponden. Eliminaremos estas etiquetas para tener una mejor visualización y reasignaremos los nombres de las columnas por los rangos taxonómicos correspondientes:

```
  # Limpiar etiquetas:
  taxonomy@tax_table@.Data <- substring(taxonomy@tax_table@.Data, 4)

  # Renombrar columnas:
  colnames(taxonomy@tax_table@.Data) <- c("Kingdom",
                                          "Phylum",
                                          "Class",
                                          "Order",
                                          "Family",
                                          "Genus",
                                          "Species")

  # Explorar resultado:
  taxonomy@tax_table@.Data %>% head()
```

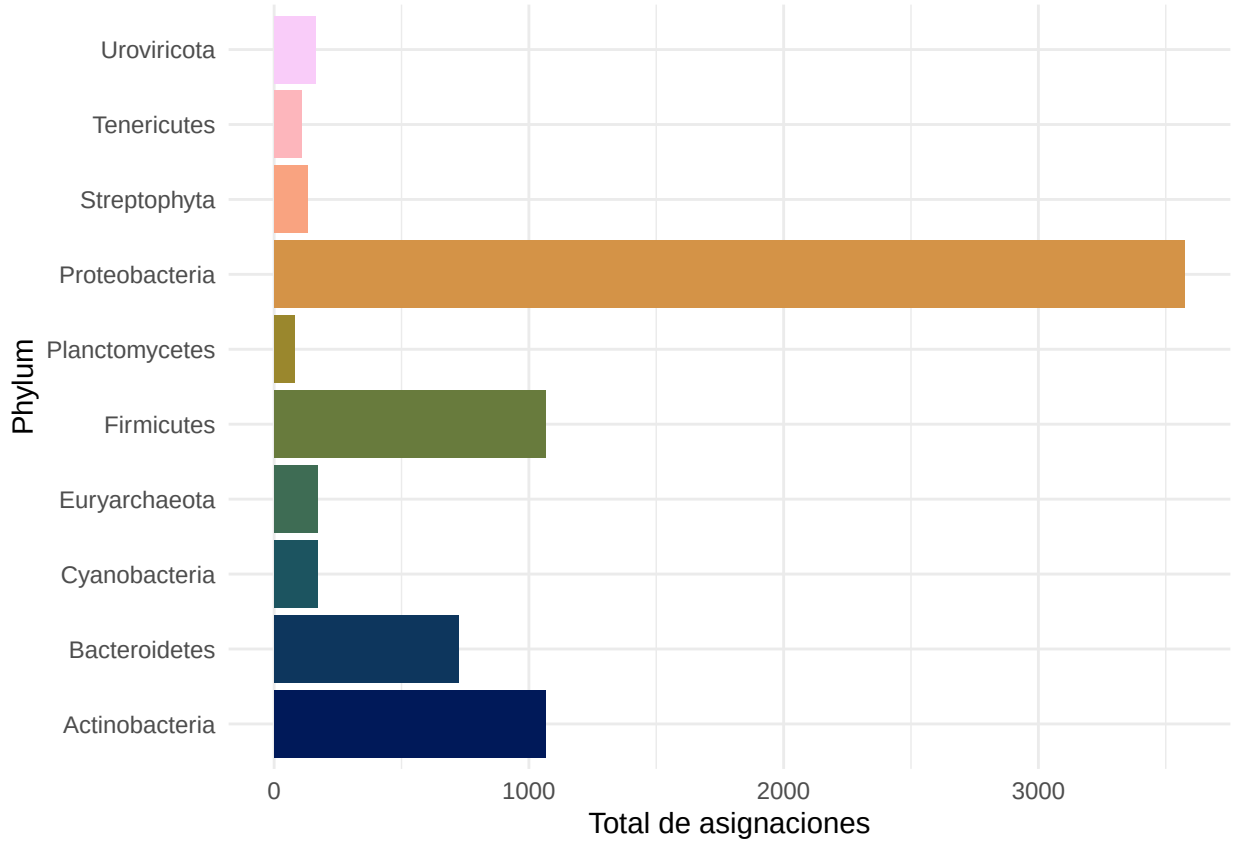
```
##       Kingdom     Phylum         Class           Order    Family    Genus
## 38820 "Eukaryota" "Streptophyta" "Magnoliopsida" "Poales" ""        ""
## 4479  "Eukaryota" "Streptophyta" "Magnoliopsida" "Poales" "Poaceae" ""
## 4577  "Eukaryota" "Streptophyta" "Magnoliopsida" "Poales" "Poaceae" "Zea"
## 4558  "Eukaryota" "Streptophyta" "Magnoliopsida" "Poales" "Poaceae" "Sorghum"
## 4539  "Eukaryota" "Streptophyta" "Magnoliopsida" "Poales" "Poaceae" "Panicum"
## 38727 "Eukaryota" "Streptophyta" "Magnoliopsida" "Poales" "Poaceae" "Panicum"
##       Species
## 38820 ""
## 4479  ""
## 4577  "mays"
## 4558  "bicolor"
## 4539  ""
## 38727 "virgatum"
```

Con esta primer limpieza podemos explorar cuántos linajes distintos se asignaron a los distintos niveles taxonómicos. Checamos los tres más altos:

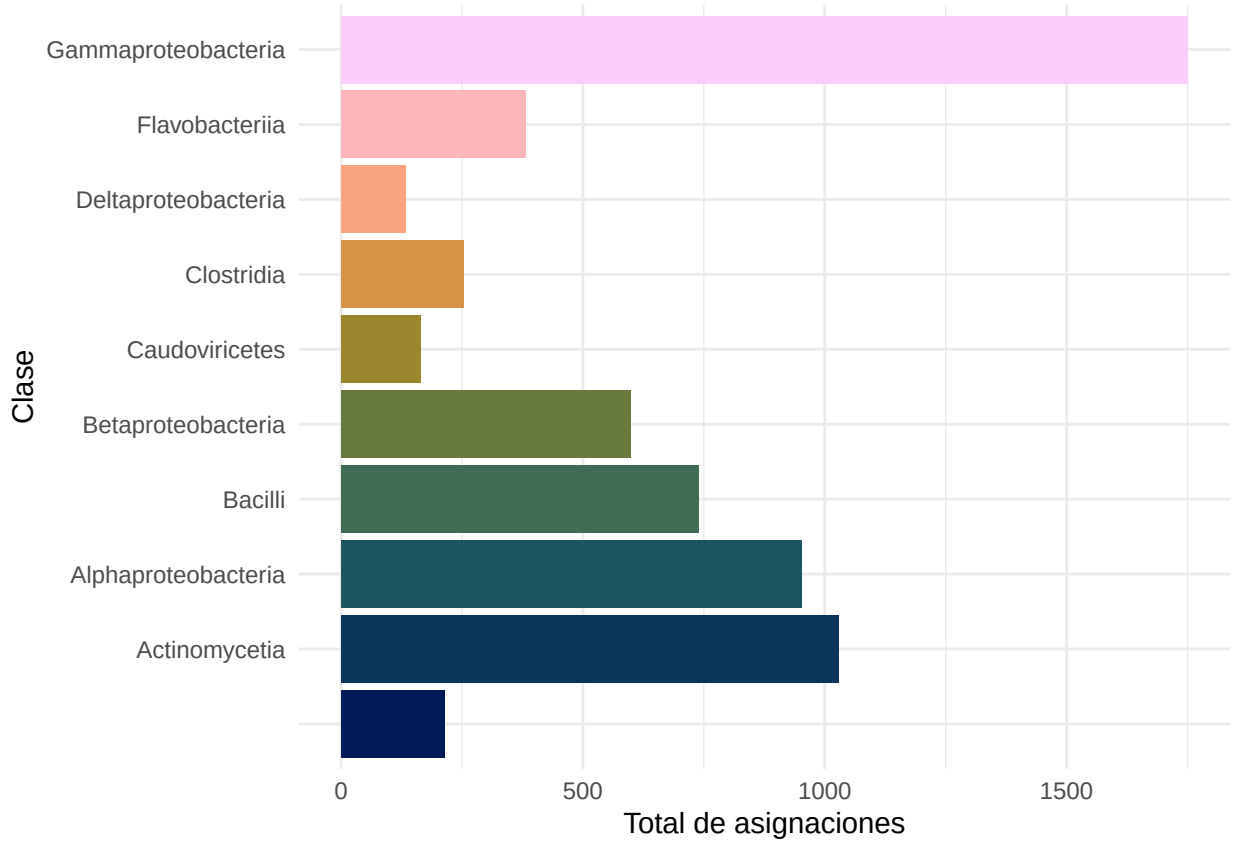
```
taxonomy@tax_table@.Data %>%
  as_tibble() %>%
  group_by(Kingdom) %>%
  count() %>%
  ggplot(aes(x = Kingdom, y = n, fill = Kingdom)) +
      geom_bar(stat = "identity") +
      coord_flip() +
      scale_fill_scico_d(palette = "batlow") +
      labs(x = "Total de asignaciones",
           y = "Kingdom") +
      theme_minimal() +
      theme(legend.position = "none")
```




```
taxonomy@tax_table@.Data %>%
  as_tibble() %>%
  group_by(Phylum) %>%
  count() %>%
  arrange(desc(n)) %>%
  head(10) %>%
  ggplot(aes(x = Phylum, y = n, fill = Phylum)) +
      geom_bar(stat = "identity") +
      scale_fill_scico_d(palette = "batlow") +
      labs(y = "Total de asignaciones",
           x = "Phylum") +
      theme_minimal() +
      coord_flip() +
      theme(legend.position = "none")
```




```
taxonomy@tax_table@.Data %>%
  as_tibble() %>%
  group_by(Class) %>%
  count() %>%
  arrange(desc(n)) %>%
  head(10) %>%
  ggplot(aes(x = Class, y = n, fill = Class)) +
      geom_bar(stat = "identity") +
      scale_fill_scico_d(palette = "batlow") +
      labs(y = "Total de asignaciones",
           x = "Clase") +
      coord_flip() +
      theme_minimal() +
      theme(legend.position = "none")
```



Como vimos en la exploración por taxón, se encontraron algunos virus a nivel de reino. Ahora limpiaremos las asignaciones correspondientes a virus, mitocondrias o cloroplastos:

```
taxonomy <- subset_taxa(taxonomy,Kingdom != "Viruses" &
                                 Family != "mitochondria" &
                                 Class != "Chloroplast")
```

Una vez limpios nuestros datos, podemos volver a hacer las gráficas anteriores para visualizar los totales limpios.

Abundancia taxonómica

En esta sección exploraremos la composición taxonómica y de abundancia en las tres muestras.

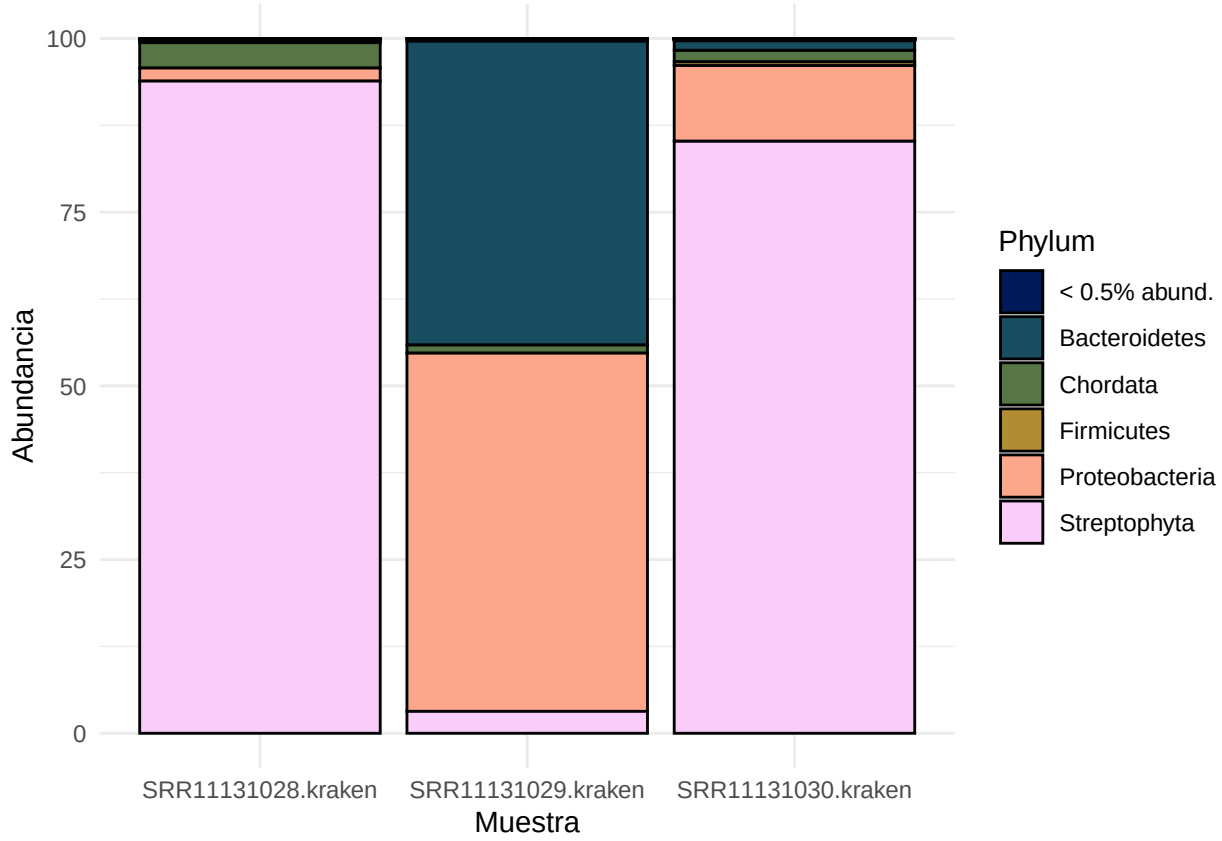
Primero generaremos el porcentaje de **abundancia** de cada taxa en cada muestra y limitaremos el estudio a nivel de **phylum**.

```
  # Calculamos los porcentajes de abundancia por muestra
  percentages <- transform_sample_counts(taxonomy,
                                         function(x) x * 100 / sum(x))
  # Verificamos:
  percentages@otu_table@.Data %>% head
```

```
##       SRR11131028.kraken SRR11131029.kraken SRR11131030.kraken
## 38820         0.03639054       4.436728e-05         0.03541256
## 4479          2.07887012       2.429962e-03         2.20331314
## 4577         85.74856944       2.235428e-03        76.44822643
## 4558          0.36459384       2.935066e-02         0.39025317
```

```
## 4539          0.04969143       4.095441e-05         0.05015882
## 38727         0.96433318       5.767746e-04         0.85869689
```

```
# Nos quedamos con los datos a nivel de Phylum y
# aquellos phyla que tengan una abundancia menor
# de <0.5 las colapsamos en una misma clase para la
# visualización.
phylaTax <- tax_glom(percentages, taxrank = "Phylum") %>%
                psmelt() %>%
                as_tibble() %>%
                mutate(Label = ifelse(Abundance < 0.5,
                          "< 0.5% abund.",
                          Phylum))
# Visualizamos:
phylaTax %>%
      ggplot(aes(x = Sample, y = Abundance, fill = Label)) +
          geom_bar(stat = "identity",
                   position = "stack",
                   color = "black") +
          labs(x = "Muestra", y = "Abundancia", fill = "Phylum") +
          scale_fill_scico_d(palette = "batlow") +
          theme_minimal()
```




```
# Para ver aquellos phyla con poca abundancia (<0.5):
phylaTax %>%
    filter(Label == "< 0.5% abund.") %>%
    dplyr::select(Phylum) %>%
```

```
      unique()
```

```
## # A tibble: 63 x 1
##    Phylum
##    <chr>
##  1 Firmicutes
##  2 Bacteroidetes
##  3 Actinobacteria
##  4 Phixviricota
##  5 Uroviricota
##  6 Deinococcus-Thermus
##  7 Cyanobacteria
##  8 Planctomycetes
##  9 Ascomycota
## 10 Candidatus Saccharibacteria
## # ... with 53 more rows
```

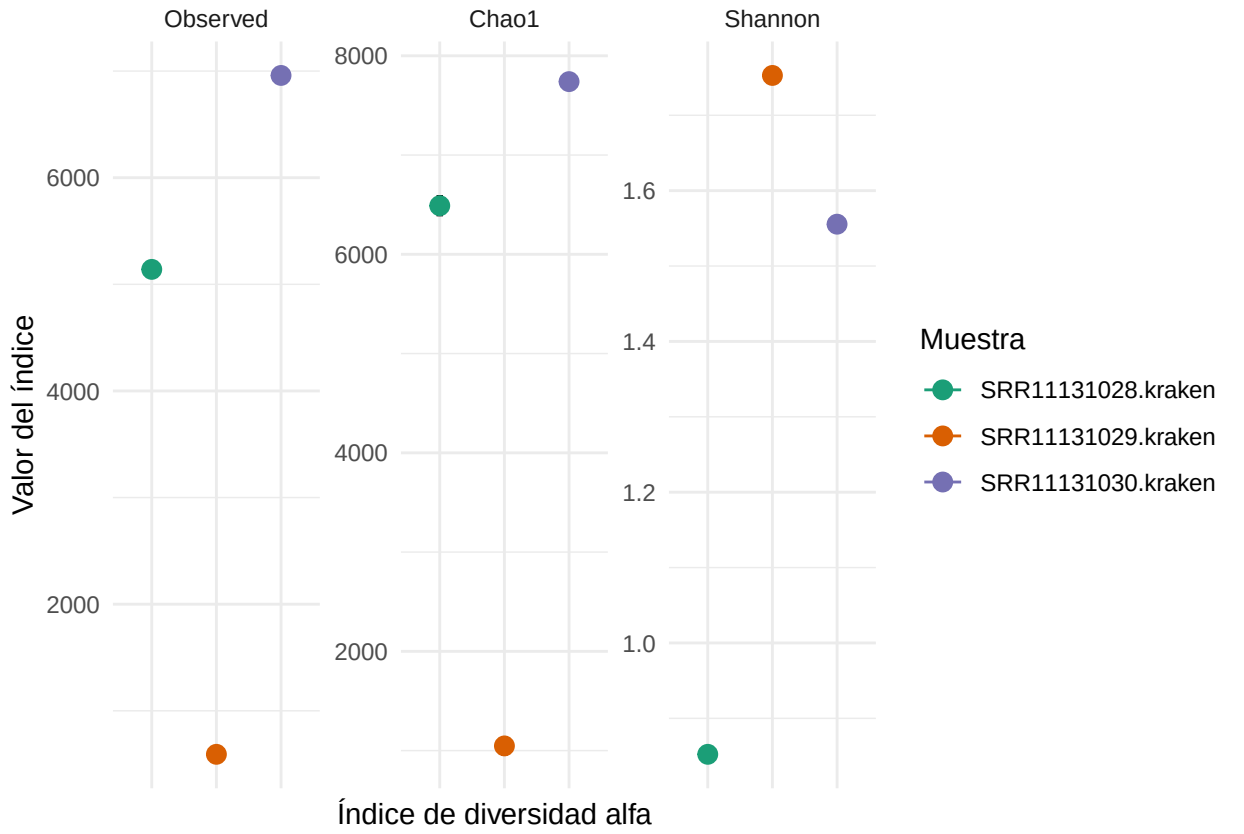
Análisis de diversidad

Una vez que analizamos la composición y abundancia taxonómica de las muestras, vamos a analizar cómo es su diversidad. La diversidad alfa, explica la diversidad *dentro de cada muestra*, mientras que la diversidad beta analiza la diversidad *entre las distintas muestras*.

Diversidad alpha

La función ‘`plot_richness` calcula los distintos índices de diversidad al mismo tiempo y provee el gráfico base de estos valores.

```
# Hacemos el grafico de los distintos indices de diversidad
# y lo guardamos en una variable para poder acceder posteriormente
# a los datos crudos.

alfaDiv <- plot_richness(taxonomy,
              measures = c("Observed",
                           "Chao1",
                           "Shannon")) +
              geom_point(aes(color = samples), size = 3) +
              scale_color_brewer(palette = "Dark2") +
              labs(x = "Índice de diversidad alfa",
                   y = "Valor del índice",
                   color = "Muestra") +
              theme_minimal() +
              theme(axis.text.x = element_blank())

# Visualizamos el grafico:
alfaDiv
```




```
  # Consultamos los datos crudos:
  alfaDiv$data
```

```
##                   Id            samples variable        value       se
## 1 SRR11131028.kraken SRR11131028.kraken Observed 5140.0000000       NA
## 2 SRR11131029.kraken SRR11131029.kraken Observed  592.0000000       NA
## 3 SRR11131030.kraken SRR11131030.kraken Observed 6959.0000000       NA
## 4 SRR11131028.kraken SRR11131028.kraken    Chao1 6489.7601523 93.04324
## 5 SRR11131029.kraken SRR11131029.kraken    Chao1 1047.2380952 84.62264
## 6 SRR11131030.kraken SRR11131030.kraken    Chao1 7739.6335150 62.35239
## 7 SRR11131028.kraken SRR11131028.kraken  Shannon    0.8522844       NA
## 8 SRR11131029.kraken SRR11131029.kraken  Shannon    1.7528400       NA
## 9 SRR11131030.kraken SRR11131030.kraken  Shannon    1.5555161       NA
```

Diversidad beta

Existen varias formas de calcular la diversidad beta. Aquí se muestra el método NMDS con distancia Bray-Curtis. El objetivo del método NMDS es hacer un análisis de reducción de dimensionalidad entre las muestras de manera que si fuesen similares generarían clusters en la representación 2D.

```
# Calculamos con la funcion ordinate
  betaDiv <- ordinate(percentages,
                      method = "NMDS",
                      distance = "bray")
```

```
## Square root transformation
## Wisconsin double standardization
## Run 0 stress 0
```

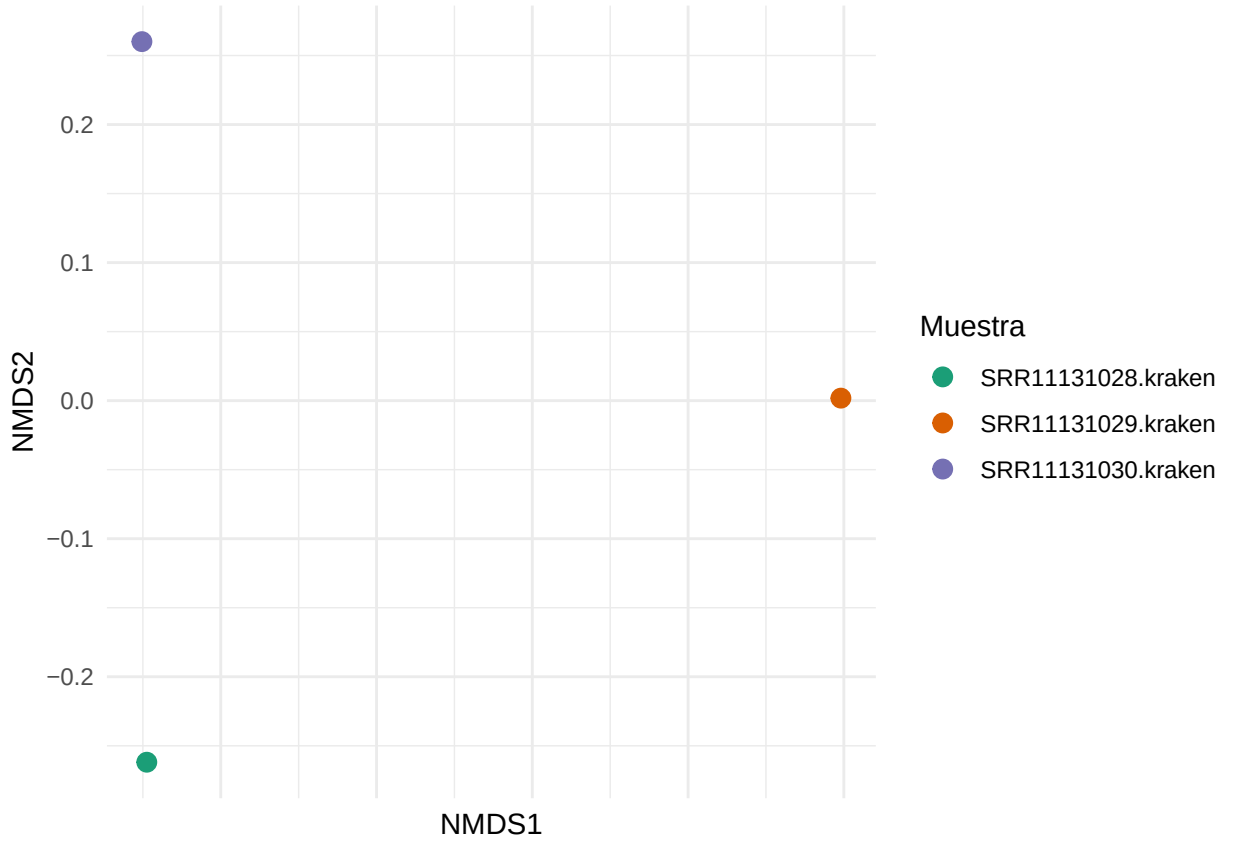
```
## Run 1 stress 0
## ... Procrustes: rmse 0.1390331  max resid 0.1702115
## Run 2 stress 0
## ... Procrustes: rmse 0.1148913  max resid 0.1330408
## Run 3 stress 0
## ... Procrustes: rmse 0.1975406  max resid 0.2675699
## Run 4 stress 0
## ... Procrustes: rmse 0.1409906  max resid 0.1749394
## Run 5 stress 0
## ... Procrustes: rmse 0.1313599  max resid 0.151565
## Run 6 stress 0
## ... Procrustes: rmse 0.1340694  max resid 0.1532895
## Run 7 stress 0
## ... Procrustes: rmse 0.1965149  max resid 0.2617583
## Run 8 stress 0
## ... Procrustes: rmse 0.165416  max resid 0.2112083
## Run 9 stress 0
## ... Procrustes: rmse 0.2892668  max resid 0.4058813
## Run 10 stress 0
## ... Procrustes: rmse 0.1506079  max resid 0.1547095
## Run 11 stress 0
## ... Procrustes: rmse 0.2930874  max resid 0.4112433
## Run 12 stress 0
## ... Procrustes: rmse 0.1739437  max resid 0.1766179
## Run 13 stress 0
## ... Procrustes: rmse 0.1790923  max resid 0.1792757
## Run 14 stress 0
## ... Procrustes: rmse 0.1372005  max resid 0.16693
## Run 15 stress 0
## ... Procrustes: rmse 0.1130325  max resid 0.1220603
## Run 16 stress 0
## ... Procrustes: rmse 0.1268069  max resid 0.1579495
## Run 17 stress 0
## ... Procrustes: rmse 0.04446322  max resid 0.04958306
## Run 18 stress 0
## ... Procrustes: rmse 0.1547667  max resid 0.1996715
## Run 19 stress 0
## ... Procrustes: rmse 0.0639068  max resid 0.07443458
## Run 20 stress 0
## ... Procrustes: rmse 0.2866335  max resid 0.4025722
## *** Best solution was not repeated -- monoMDS stopping criteria:
##     20: stress < smin
```

```
# Hacemos el grafico inicial
  x <- plot_ordination(percentages,
                  ordination = betaDiv)
```

```
## No available covariate data to map on the points for this plot `type`
```

```
# Agregamos la variable de la muestra para poder
# diferenciarlas en el grafico
  x$data <- x$data %>%
                mutate(muestra = rownames(x$data))

# Grafico final:
```

```
x +
  geom_point(aes(color = muestra), size = 3) +
  scale_color_brewer(palette = "Dark2") +
  labs(color = "Muestra") +
  theme_minimal() +
  theme(axis.text.x = element_blank())
```



Observamos que nuestras muestras se encuentran muy separadas entre sí y, por lo tanto, son disímiles entre ellas. Para utilizar otros métodos y distancias puede consultarse la función `distanceMethodList`.