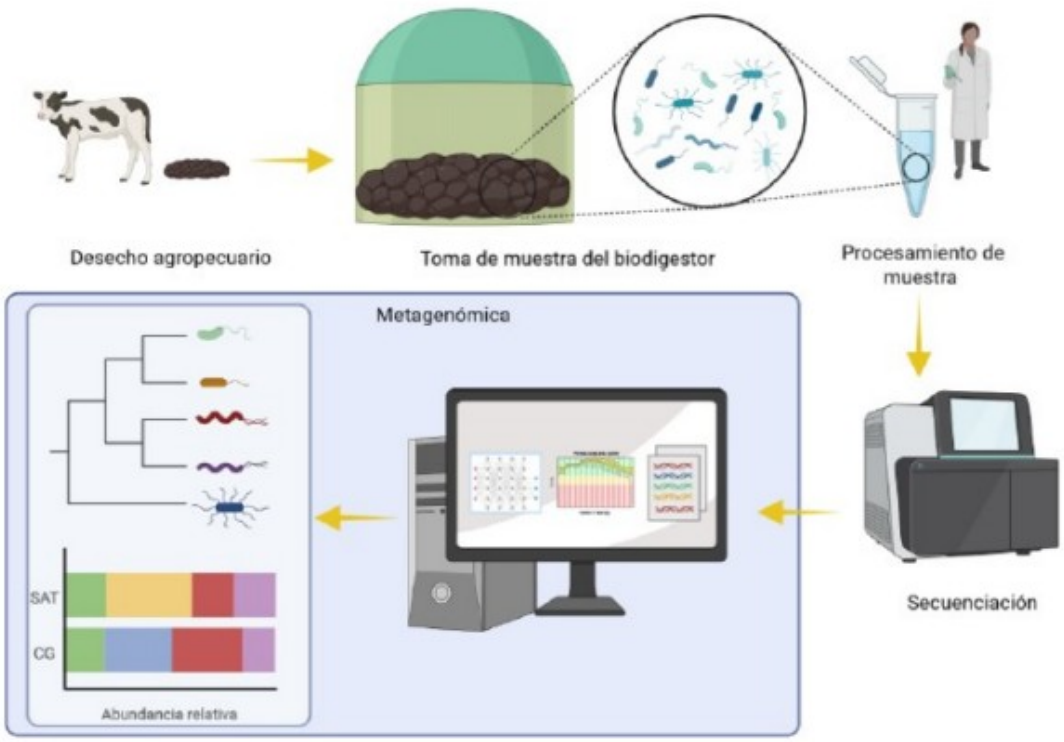
Manual de Usuario

Pipeline metagenómica

Dulce I. Valdivia Martínez
Erika V. Cruz Bonilla

Abril 2023



Resumen

Este pipeline de metagenómica se desarrolló para el proyecto Conacyt "*Producción de biocombustibles para uso rural a partir de desechos agropecuarios mediante la optimización de consorcios microbianos usando metagenómica*". En este documento se describen los pasos necesarios para analizar datos de secuenciación crudos en Unix y generar distintos análisis de diversidad en R.
El pipeline puede encontrarse en https://github.com/meta-genomics/Metagenomics

Índice

1. Descripción general del pipeline....2
2. Requerimentos....4
2.1. Software....4
2.1.1. Ambiente conda....4
2.1.2. SRA toolkit....4
2.1.3. FastQC....4
2.1.4. Trimmomatic....5
2.1.5. Kraken2....5
2.1.6. MaxBin2....5
2.1.6.1 Bowtie2....5
2.1.6.2 FragGeneScan....5
2.1.6.3 Hmmer3....5
2.1.6.4 IDBA-UD....5
2.1.7. metaSpades....5
2.1.8. kraken-biom....6
2.1.9. checkM....6
2.2 Paquetes de R....6
2.2.1. Phyloseq....6
2.2.2. ggplot2....6
2.2.3. scico....6
2.2.4. tidyverse....6
3. Procesamiento de muestras en el servidor....6
3.1 Especificación de directorios de bases de datos....6
3.1.1 metaSPAdes....7
3.1.2 MaxBin2....7
3.1.3 CheckM Database....7
3.1.4 Kraken Database....7
3.2. Creación del entorno de trabajo....7
3.3. Activación de ambiente conda....8
3.4. Procesamiento principal de las muestras....8
4. Análisis de diversidad en R....8

1. Descripción general del pipeline

El diagrama de flujo del pipeline desarrollado se muestra en la Figure 1. Consiste en tres scripts principales:

- El script `setup.sh` crea los directorios necesarios para llevar de forma automática el análisis.
- El script `metagenomics.sh` es el principal y procesa desde los datos crudos de secuenciación hasta la asignación taxonómica.
- Por último, el archivo de Rmarkdown `AnalisisDiversidadMetagenomica.Rmd` genera los un análisis de abundancia y diversidad.

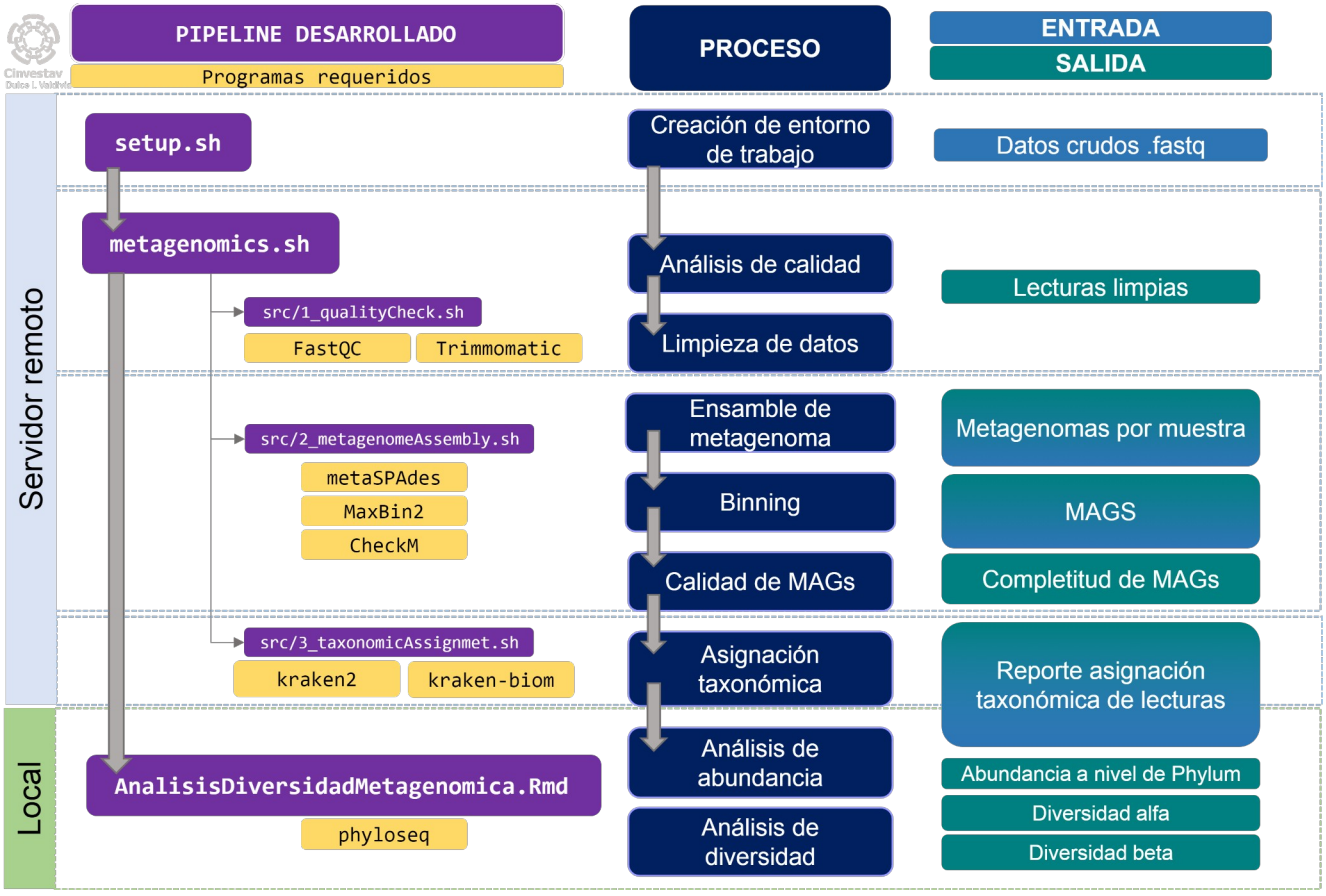


Figura 1: Diagrama de flujo del pipeline de metagenómica

Este pipeline se basa en el artículo Garfias-Gallegos et al., (2022) *Methods in Molecular Biology*, vol.2512 https://doi.org/10.1007/978-1-0716-2429-6_10 con varias modificaciones y correcciones.

2. Requerimentos

2.1. Software

Para poder utilizar los scripts principales, es necesario instalar varios programas y descargar bases de datos. Es necesario tener conda instalado para proceder con algunas instalaciones de manera sencilla.
A continuación se especifican estos programas y se explican los pasos de instalación

> Importante:
> § Los usuarios del servidor botanero no necesitan realizar la instalación de estos programas, ya se encuentran instalados en el servidor. Únicamente deben de instalar en su computadora personal los señalados en la sección de paquetes de R.
> § Los siguientes pasos de instalación requieren permisos de superusuario.
> § Este manual asumirá la creación de un ambiente conda donde corran estos programas.
> § Los siguientes pasos requieren modificar variables globales por lo que deben realizarse con la precaución necesaria.
> § Se recomienda generar un directorio lib/ donde se guarden los programas descargados. En el resto del manual se asume que se instalaron de esa forma los programas y que este directorio está en el mismo lugar que donde se correrá el pipeline.

2.1.1. Ambiente conda

Crearemos un ambiente llamado meta-omics. Es importante especificar que se necesita una version de python ≥ 3.4 para la compatibilidad con el resto de los programas.

```
$ conda create -n meta-omics python=3.5 anaconda
```

2.1.2. SRA toolkit

Buscar en la siguiente página https://github.com/ncbi/sra-tools/wiki/01.-Downloading-SRA-Toolkit la versión adecuada para el sistema operativo en el que se está trabajando y descargarlo.
Para seguir el proceso, digamos que el archivo descargado es:

sratoolkit.current-ubuntu64.tar.gz

Descomprimir el archivo y agregar el directorio fuente a la variable PATH:

```
$ tar -vxzf sratoolkit.tar.gz
$ cd sratoolkit.3.0.0-ubuntu64/lib
$ export PATH=$PATH:$PWD
```

Antes de usar SRA toolkit es necesario configurarlo según:
https://github.com/ncbi/sra-tools/wiki/03.-Quick-Toolkit-Configuration

2.1.3. FastQC

```
$ conda install -c bioconda fastqc
```

2.1.4. Trimmomatic

```
$ conda install -c bioconda trimmomatic
```

2.1.5. Kraken2

Instalar kraken:

```
$ conda install -c bioconda kraken2
```

Seleccionar una base de datos de esta página: https://benlangmead.github.io/aws-indexes/k2 y descargarla con wget. Descomprimir la base de datos (tar -vxzf filename.tar.gz) en un directorio de repositorio.

2.1.6. MaxBin2

Descargar con wget el archivo MaxBin-2.2.7.tar.gz desde
https://sourceforge.net/projects/maxbin2/files/

Después hacer:

```
$ tar -xvzf MaxBin-2.2.7.tar.gz
$ cd MaxBin-2.2.7
$ export PATH=$PATH:$PWD
$ cd src
$ make
$ cd ..
$ ./ autobuild_auxiliary
```

Si el autobuild no funciona para instalar los programas auxiliares, instalar manualmente los siguientes programas:

2.1.6.1 Bowtie2

```
$ conda install -c bioconda bowtie2
```

2.1.6.2 FragGeneScan

```
$ conda install -c bioconda fraggenescan
```

2.1.6.3 Hmmer3

```
$ conda install -c bioconda hmmer
```

2.1.6.4 IDBA-UD

```
$ conda install -c bioconda idba
```

2.1.7. metaSpades

Descargar con wget el archivo SPAdes-3.15.5-Linux.tar.gz desde https://cab.spbu.ru/software/spades/

```
$ wget http://cab.spbu.ru/files/release3.15.5/SPAdes-3.15.5-Linux.tar.gz
$ tar -xzf SPAdes-3.15.5.tar.gz
$ cd SPAdes-3.15.5-Linux/bin/
$ export PATH=$PATH:$PWD
```

2.1.8. kraken-biom

```
$ conda install -c bioconda kraken-biom
```

2.1.9. checkM

Instalamos el programa y las bibliotecas que necesita:

```
$ conda install numpy matplotlib
$ conda install -c bioconda pysam
$ conda install hmmer
$ conda install -c bioconda prodigal pplacer
$ pip3 install checkm–genome
```

Posteriormente descargamos con wget la base de datos checkm_data_2015_01_16.tar.gz desde https://data.ace.uq.edu.au/public/CheckM_databases/. Esta base de datos tiene que ser descomprimida en un directorio de reposición.

2.2 Paquetes de R

Estos paquetes se deben de instalar en la computadora personal del usuario.

2.2.1. Phyloseq

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

2.2.2. ggplot2

```
install.packages("ggplot2")
```

2.2.3. scico

```
install.packages("scico")
```

2.2.4. tidyverse

```
install.packages("tidyverse")
```

3. Procesamiento de muestras en el servidor

3.1 Especificación de directorios de bases de datos.

En el script metagenomics.sh es necesario especificar donde se encuentra la instalación de los programas MetaSPAdes y MaxBin así como los paths a los directorios de las bases de datos.
Para los usuarios botanero no es necesario hacer estas modificaciones al script, solo asegurarse de tener acceso a la carpeta /home/metagenomics/projects/biodigestores/

3.1.1 metaSPAdes

Modificar Línea 32:

```
export PATH="<direccionDeInstalación>/SPAdes-3.15.5-Linux/bin:$PATH"
```

3.1.2 MaxBin2

Modificar Línea 33:

```
export PATH="<direccionDeInstalación>/MaxBin-2.2.7/:$PATH"
```

3.1.3 CheckM Database

Modificar Línea 34:

```
export CHECKM_DATA_PATH="<DireccionDelaDB>"
```

3.1.4 Kraken Database

Modificar Línea 41:

```
# Config:
        dirKrakenDB="<DireccionDelaDB>"
```

Para usuarios del cluster botanero la base de datos disponible es PlusPF-8 update 08/9/2022, en el directorio /home/metagenomics/data/krakenDB

3.2. Creación del entorno de trabajo

El pipeline está hecho para correr de manera automática asumiendo un sistema de directorios específicos. Se crea de la siguiente manera:

```
$ ./setup.sh
```

Esto generará los siguientes directorios:

```
raw-reads/
results/
results/assemblies
results/fastqc
results/taxonomy
results/taxonomy/kraken
results/trimmed-reads
results/untrimmed–reads
```

Una vez generado el ambiente de trabajo, se deben poner en la carpeta raw-reads los archivos crudos de secuenciación .fastq. Pueden ser guardados ahí directamente o crear dentro de la carpeta ligas simbólicas a la localización de estos archivos para evitar duplicar datos en el servidor. Esta última opción podría realizarse de la siguiente forma:

```
$ cd raw-reads/
$ ln -s <pathAbsolutoArchivosFastq> .
```

Otra opción es descargar los datos existentes del Sequence Read Archive (SRA) de NCBI. Para descargar archivos de ahí puede utilizarse el script getData.sh. Debe de editarse por ejemplo con nano y especificarse los identificadores de las muestras a descargar. En la versión actual del script se descargan tres experimentos de prueba:

```
fasterq - dump -- split - files SRR11131028
fasterq - dump -- split - files SRR11131029
fasterq - dump -- split - files SRR11131030
```

Después para correrlo solo es necesario hacer:

```
./getData.sh
```

3.3. Activación de ambiente conda

Una vez que se realizó la instalación de todos los programas requeridos de la manera que especifica la Sección 1 de Requerimientos en este manual, procedemos a activar el ambiente conda que nos permitirá usarlos:

```
$ conda activate meta-omics
```

Para usuarios botanero, llamar al ambiente conda ya instalado en el servidor con la instrucción:

```
$ conda activate /home/metagenomics/.conda/envs/meta-omics
```

3.4. Procesamiento principal de las muestras

Para correr el análisis principal de las muestras solo es necesario especificar al programa principal metagenomics.sh el número de threads a utilizar en los demás análisis. En el siguiente ejemplo se piden 16 hilos:

```
$ ./ metagenomics . sh 16
```

La salida principal de este pipeline es el archivo:

results/taxonomy/kraken/taxonomy_kraken.json

y se utiliza como entrada en el análisis de diversidad en R.

4. Análisis de diversidad en R

El análisis de diversidad corresponde al script AnalisisDiversidadMetagenomica.Rmd el cual se ejecuta localmente en Rstudio. Este script así como su tutorial ya compilado en formato .pdf o .html pueden encontrarse en el repositorio del proyecto:

https://github.com/meta-genomics/Metagenomics/tree/main/analysis_R

A partir de la siguiente página de este documento se puede consultar el tutorial de esta última parte del análisis.

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"
```

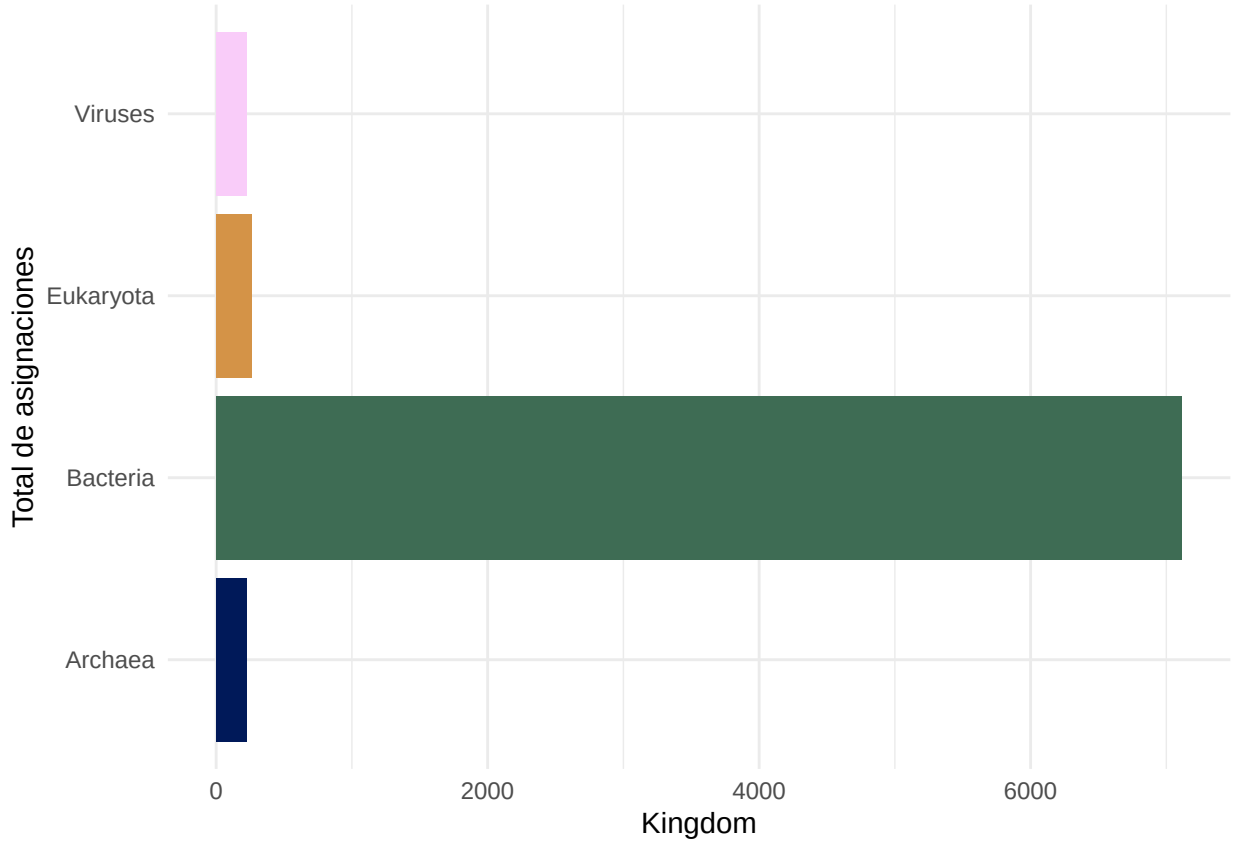
Vamos a limpiar los datos de dos maneras distintas. Primero, como observamos en la exploración del objeto tax_table, los taxones tienen al inicio una etiqueta de cuatro caracteres que corresponde al rango taxonómico al que corresponden. Eliminaremos estas etiquetas para tener una mejor visualización 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()
```

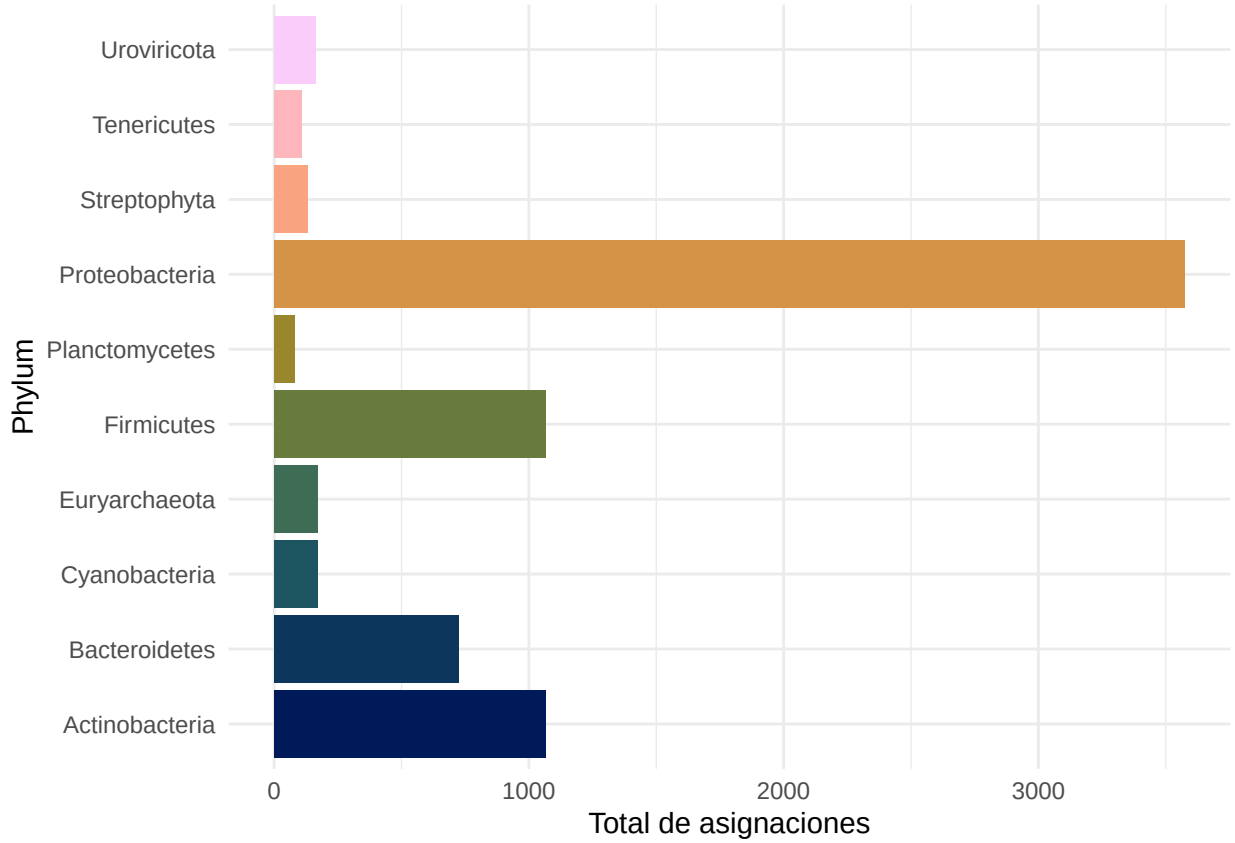
```
##       Kingdom     Phylum         Class           Order    Family    Genus
## 38820 "Eukaryota" "Streptophyta" "Magnoliopsida" "Poales" ""        ""
## 4479  "Eukaryota" "Streptophyta" "Magnoliopsida" "Poales" "Poaceae" ""
## 4577  "Eukaryota" "Streptophyta" "Magnoliopsida" "Poales" "Poaceae" "Zea"
## 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:

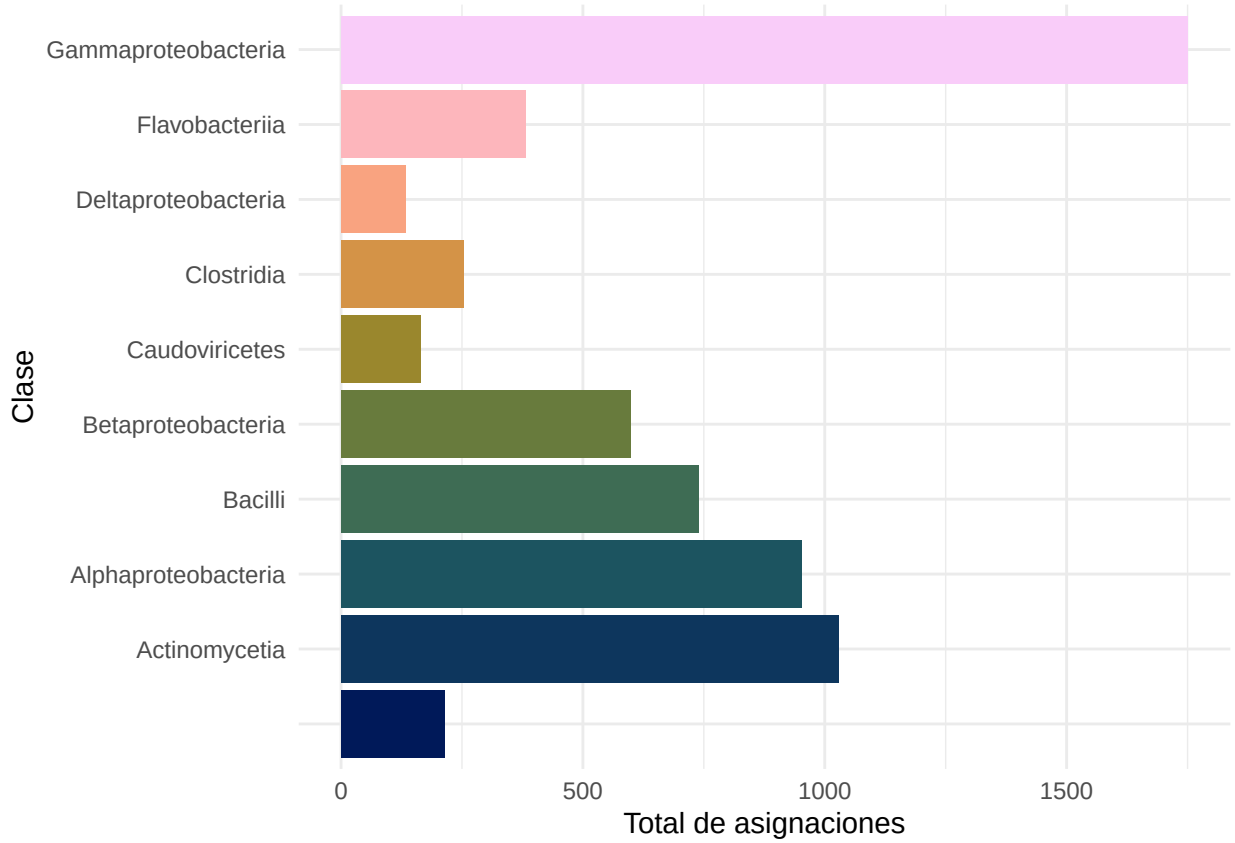
```
taxonomy@tax_table@.Data %>%
  as_tibble() %>%
  group_by(Kingdom) %>%
  count() %>%
  ggplot(aes(x = Kingdom, y = n, fill = Kingdom)) +
      geom_bar(stat = "identity") +
      coord_flip() +
      scale_fill_scico_d(palette = "batlow") +
      labs(x = "Total de asignaciones",
           y = "Kingdom") +
      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.

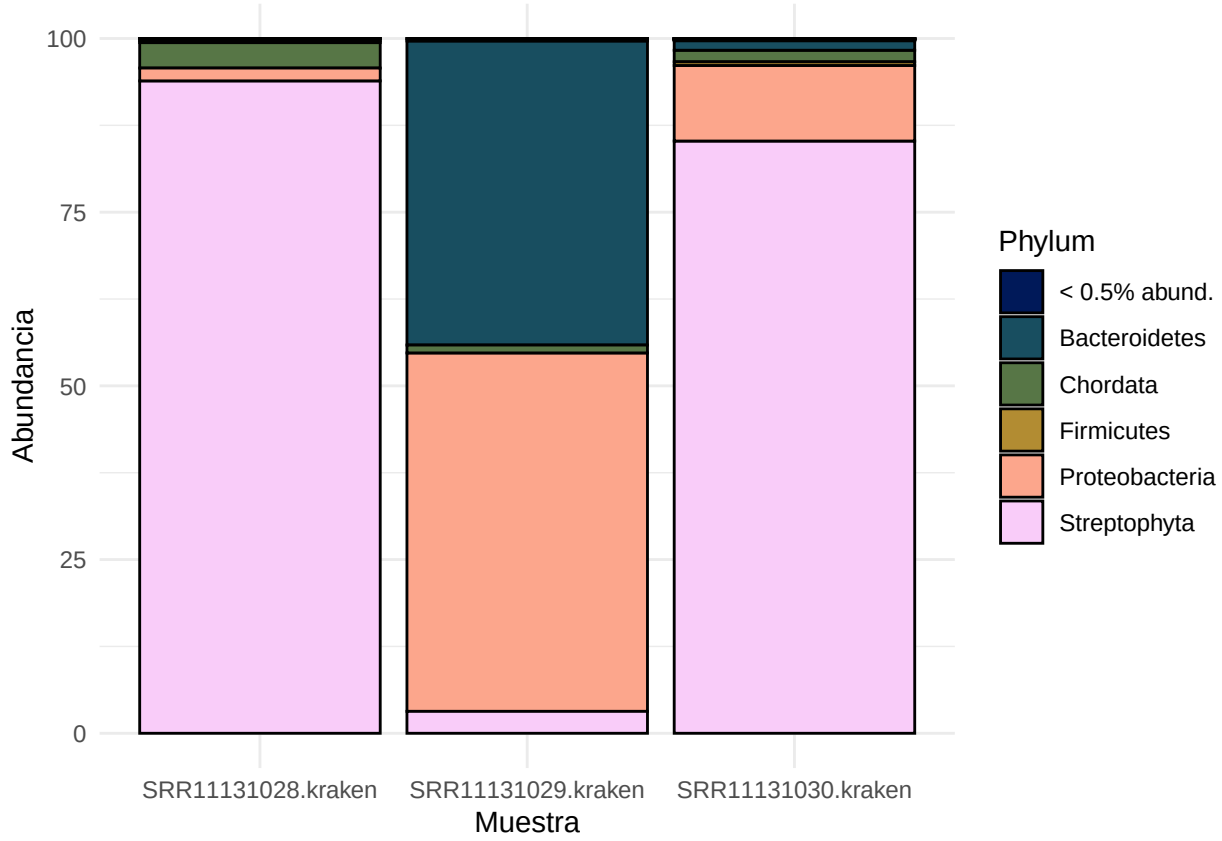
Primero generaremos el porcentaje de **abundancia** de cada taxa en cada muestra y limitaremos el estudio a nivel de **phylum**.

```
  # 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
```

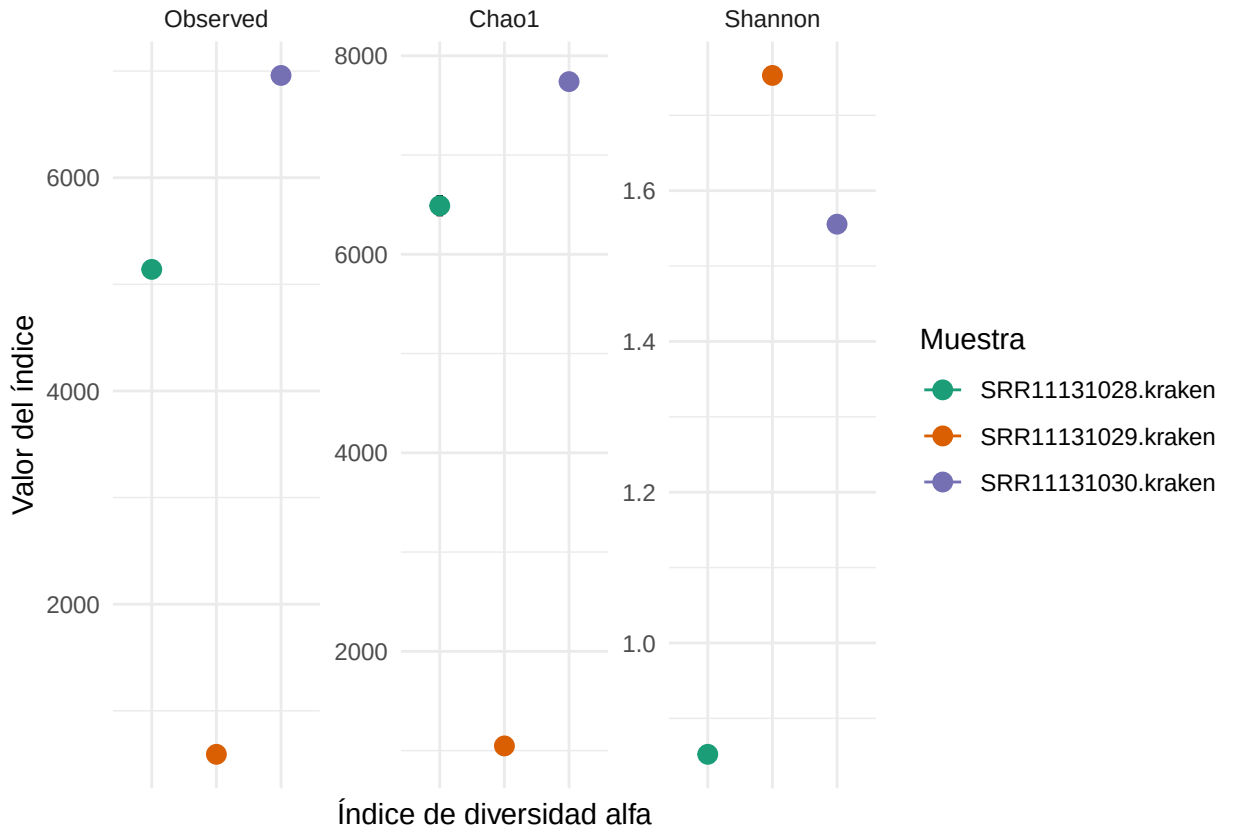
Análisis de diversidad

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
```

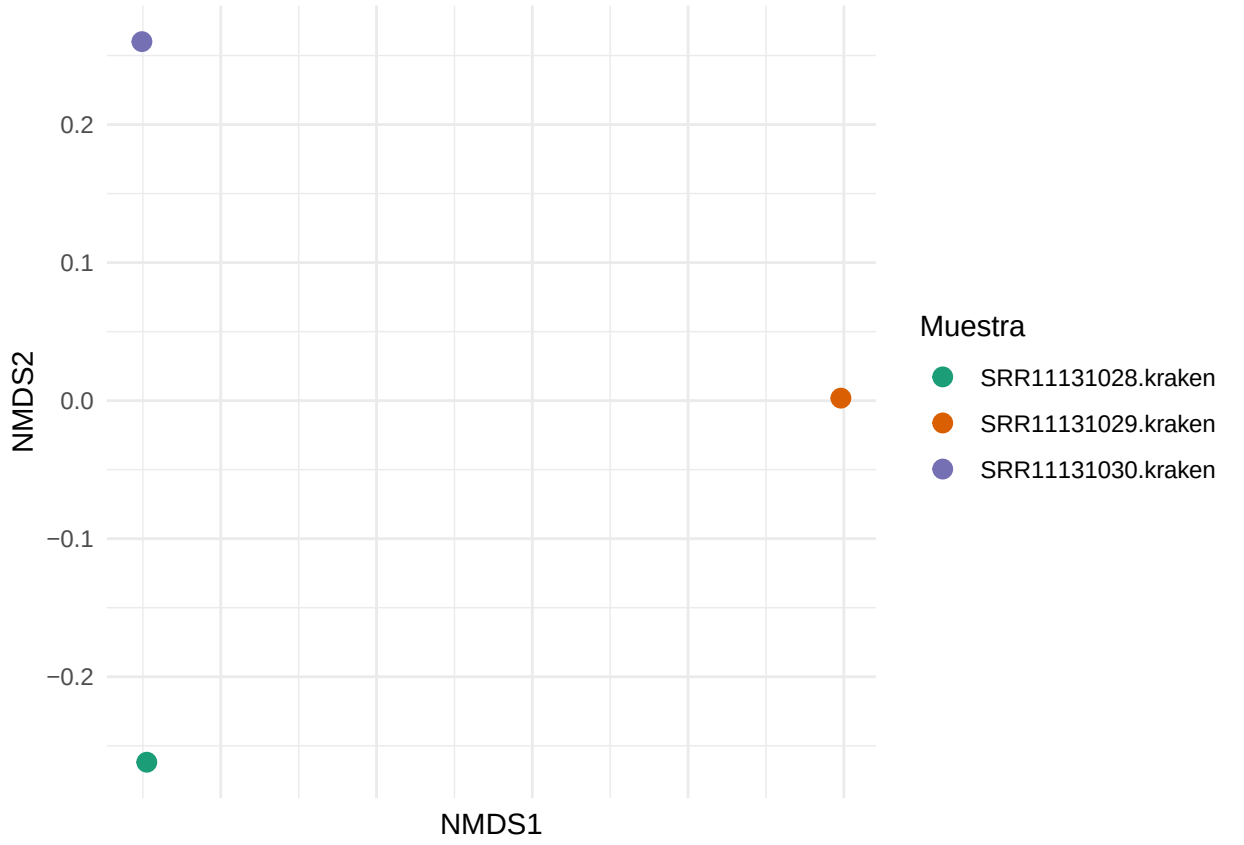
```
## Run 1 stress 0
## ... Procrustes: rmse 0.1390331  max resid 0.1702115
## Run 2 stress 0
## ... Procrustes: rmse 0.1148913  max resid 0.1330408
## Run 3 stress 0
## ... Procrustes: rmse 0.1975406  max resid 0.2675699
## Run 4 stress 0
## ... Procrustes: rmse 0.1409906  max resid 0.1749394
## Run 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`.