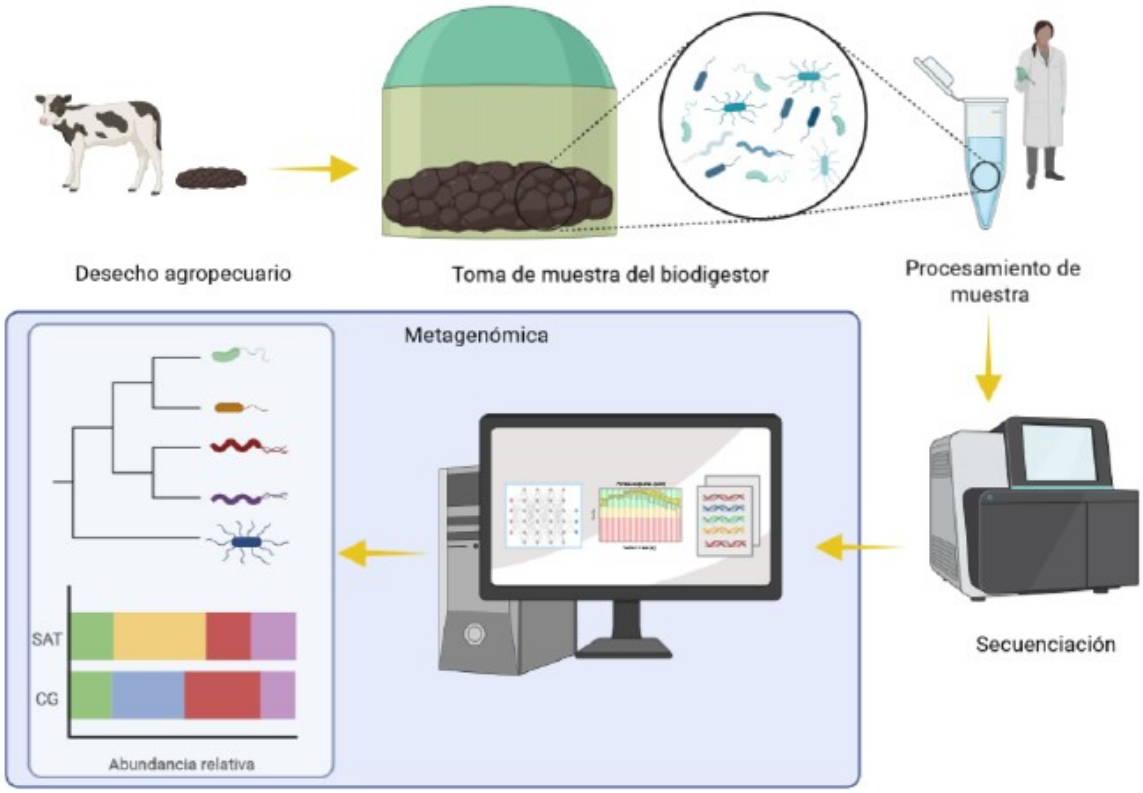
Manual de usuario

Pipeline metagenómica

Dulce I. Valdivia Martínez

28 de octubre de 2022



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/dulcirena/metagenomics

Índice

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.

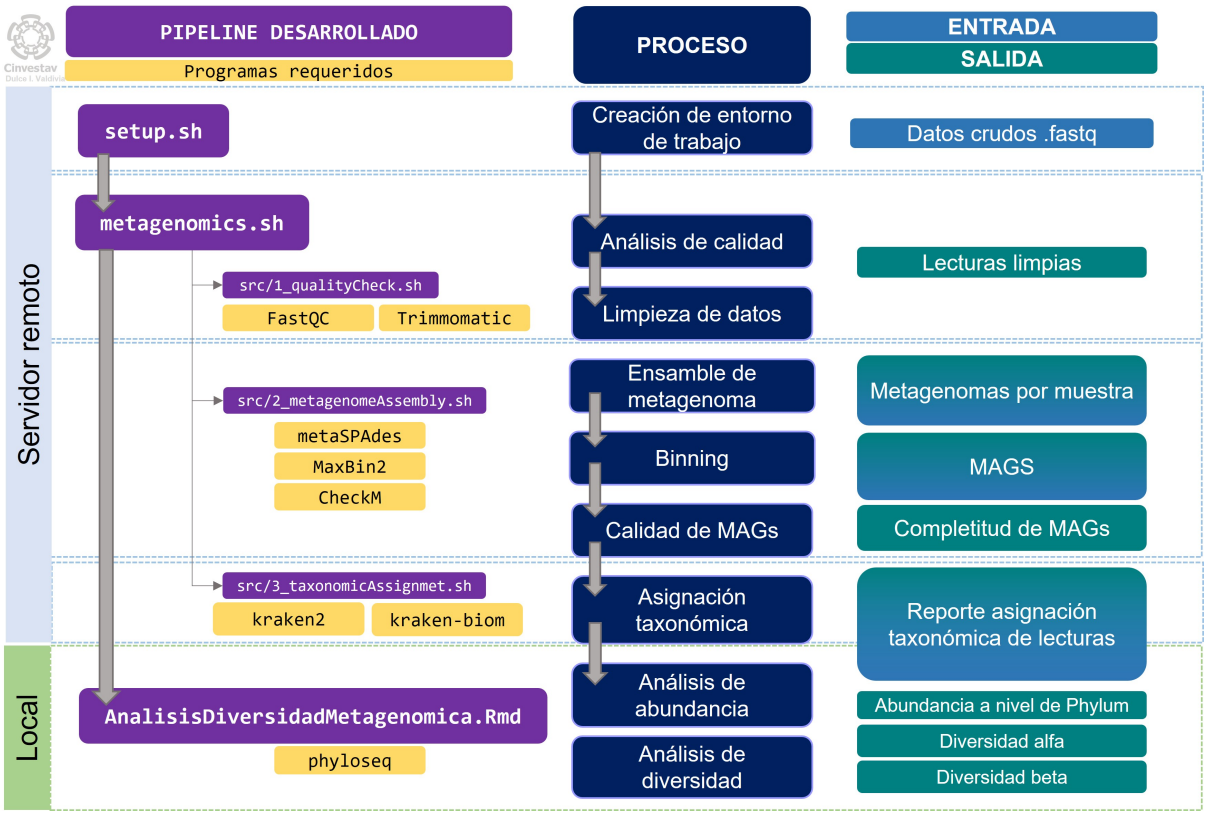


Figura 1: Diagrama de flujo del pipeline de metagenómica

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 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`

```
$ conda create -n meta-omics
```

2.1.2. SRA toolkit

Buscar en la siguiente página la versión adecuada para el sistema operativo en el que se está trabajando y descargarlo. Desde la terminal se descarga así:

```
$ wget <direccionDelArchivoAdecuado>
```

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
```

2.1.3. FastQC

```
$ conda install -c bioconda fastqc python=2.7
```

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`. Posteriormente debe ser descomprimida (`tar -vxzf`) en el directorio:

`/home/dvaldivia/data/krakenDB`

Importante:
§ El script `metagenomics.sh` asume este directorio como el lugar donde se encuentra la base de datos de kraken.

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
```

2.1.7. metaSpades

Descargar con `wget` el archivo `SPAdes-3.15.5-Linux.tar.gz` desde

https://cab.spbu.ru/software/spades/

Después:

```
$ 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 especial de la instalación.

```
$ cd ~/.checkm/
$ tar-xvzf checkm_data_2015_01_16.tar.gz
```

2.1.10. Paquetes de R

La instalación de los paquetes de R se especifica en la Sección 4 de este manual.

3. Procesamiento de muestras en el servidor

3.1. Creación de 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/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:

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

Después para correrlo solo es necesario hacer:

```
./getData.sh
```

3.2. 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
```

3.3. 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 `AnalisisDiversdiadMetagenomica.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/dulcirena/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

Contents

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")
```

1 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")
```

2 Exploración inicial y limpieza de datos

Checamos el contenido del objeto taxonomy:

```
# Información general:
  taxonomy
```

```
## phyloseq-class experiment-level object
## otu_table()   OTU Table:         [ 535 taxa and 3 samples ]
## tax_table()   Taxonomy Table:    [ 535 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
## 9604                649748             278860             182365
## 91347                11641                  1                757
## 1903409             204197                  0               1082
## 543                  15161               4761              10700
## 1903414               1405                  0               1094
## 1903411                885                182                597
```

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

```
## Taxonomy Table:     [6 taxa by 7 taxonomic ranks]:
##         Rank1          Rank2               Rank3
## 9604    "k__Eukaryota" "p__Chordata"       "c__Mammalia"
## 91347   "k__Bacteria"  "p__Proteobacteria" "c__Gammaproteobacteria"
## 1903409 "k__Bacteria"  "p__Proteobacteria" "c__Gammaproteobacteria"
## 543     "k__Bacteria"  "p__Proteobacteria" "c__Gammaproteobacteria"
## 1903414 "k__Bacteria"  "p__Proteobacteria" "c__Gammaproteobacteria"
## 1903411 "k__Bacteria"  "p__Proteobacteria" "c__Gammaproteobacteria"
##         Rank4                Rank5                    Rank6 Rank7
## 9604    "o__Primates"        "f__Hominidae"           "g__" "s__"
## 91347   "o__Enterobacterales" "f__"                   "g__" "s__"
## 1903409 "o__Enterobacterales" "f__Erwiniaceae"       "g__" "s__"
## 543     "o__Enterobacterales" "f__Enterobacteriaceae" "g__" "s__"
## 1903414 "o__Enterobacterales" "f__Morganellaceae"    "g__" "s__"
## 1903411 "o__Enterobacterales" "f__Yersiniaceae"      "g__" "s__"
```

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

```
##         Rank1          Rank2               Rank3
## 9604    "k__Eukaryota" "p__Chordata"       "c__Mammalia"
## 91347   "k__Bacteria"  "p__Proteobacteria" "c__Gammaproteobacteria"
## 1903409 "k__Bacteria"  "p__Proteobacteria" "c__Gammaproteobacteria"
## 543     "k__Bacteria"  "p__Proteobacteria" "c__Gammaproteobacteria"
## 1903414 "k__Bacteria"  "p__Proteobacteria" "c__Gammaproteobacteria"
## 1903411 "k__Bacteria"  "p__Proteobacteria" "c__Gammaproteobacteria"
##         Rank4                Rank5                    Rank6 Rank7
```

```
## 9604    "o__Primates"         "f__Hominidae"          "g__" "s__"
## 91347   "o__Enterobacterales" "f__"                   "g__" "s__"
## 1903409 "o__Enterobacterales" "f__Erwiniaceae"        "g__" "s__"
## 543     "o__Enterobacterales" "f__Enterobacteriaceae" "g__" "s__"
## 1903414 "o__Enterobacterales" "f__Morganellaceae"     "g__" "s__"
## 1903411 "o__Enterobacterales" "f__Yersiniaceae"       "g__" "s__"
```

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:

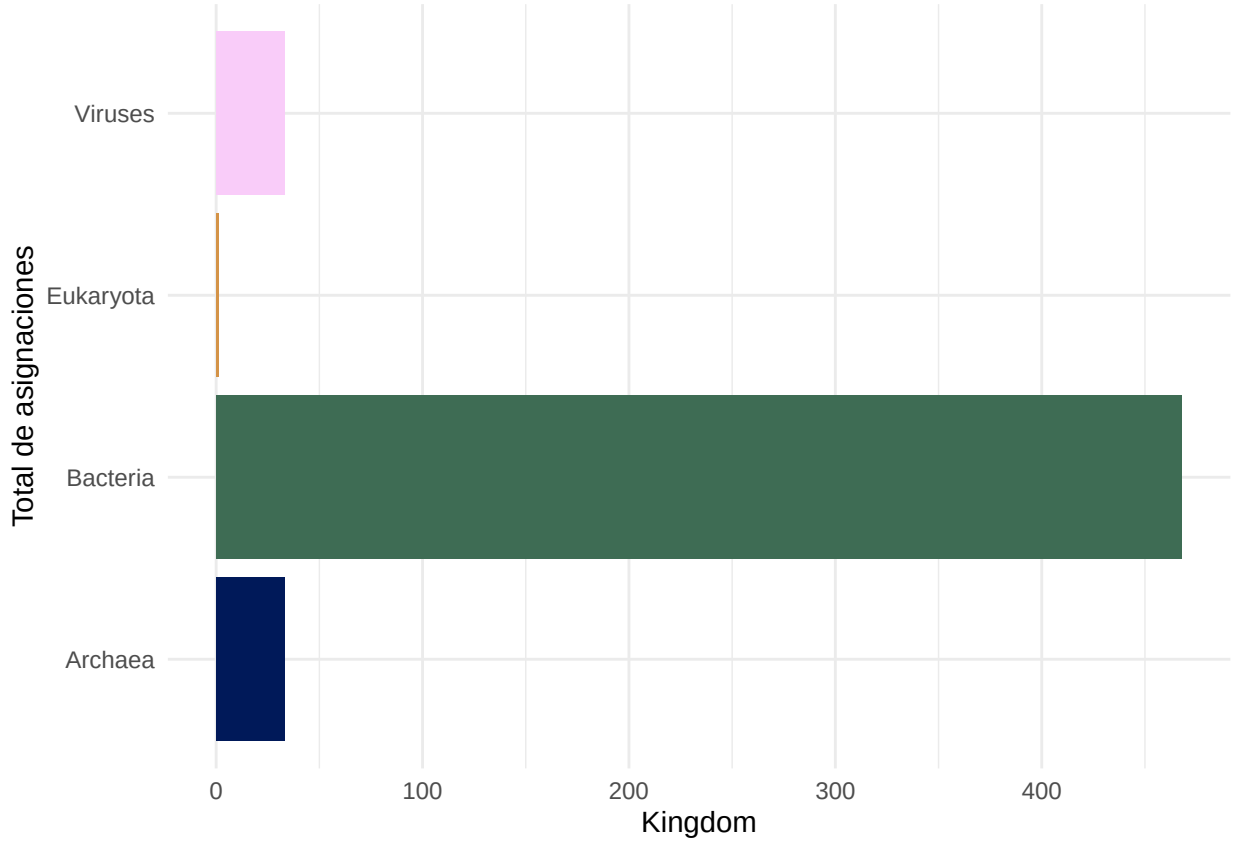
```
# Limpiar etiquetas:
taxonomy@tax_table@.Data <- substring(taxonomy@tax_table@.Data, 4)

# 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
## 9604    "Eukaryota" "Chordata"       "Mammalia"             "Primates"
## 91347   "Bacteria"  "Proteobacteria" "Gammaproteobacteria" "Enterobacterales"
## 1903409 "Bacteria"  "Proteobacteria" "Gammaproteobacteria" "Enterobacterales"
## 543     "Bacteria"  "Proteobacteria" "Gammaproteobacteria" "Enterobacterales"
## 1903414 "Bacteria"  "Proteobacteria" "Gammaproteobacteria" "Enterobacterales"
## 1903411 "Bacteria"  "Proteobacteria" "Gammaproteobacteria" "Enterobacterales"
##         Family               Genus Species
## 9604    "Hominidae"          ""    ""
## 91347   ""                   ""    ""
## 1903409 "Erwiniaceae"        ""    ""
## 543     "Enterobacteriaceae" ""    ""
## 1903414 "Morganellaceae"     ""    ""
## 1903411 "Yersiniaceae"       ""    ""
```

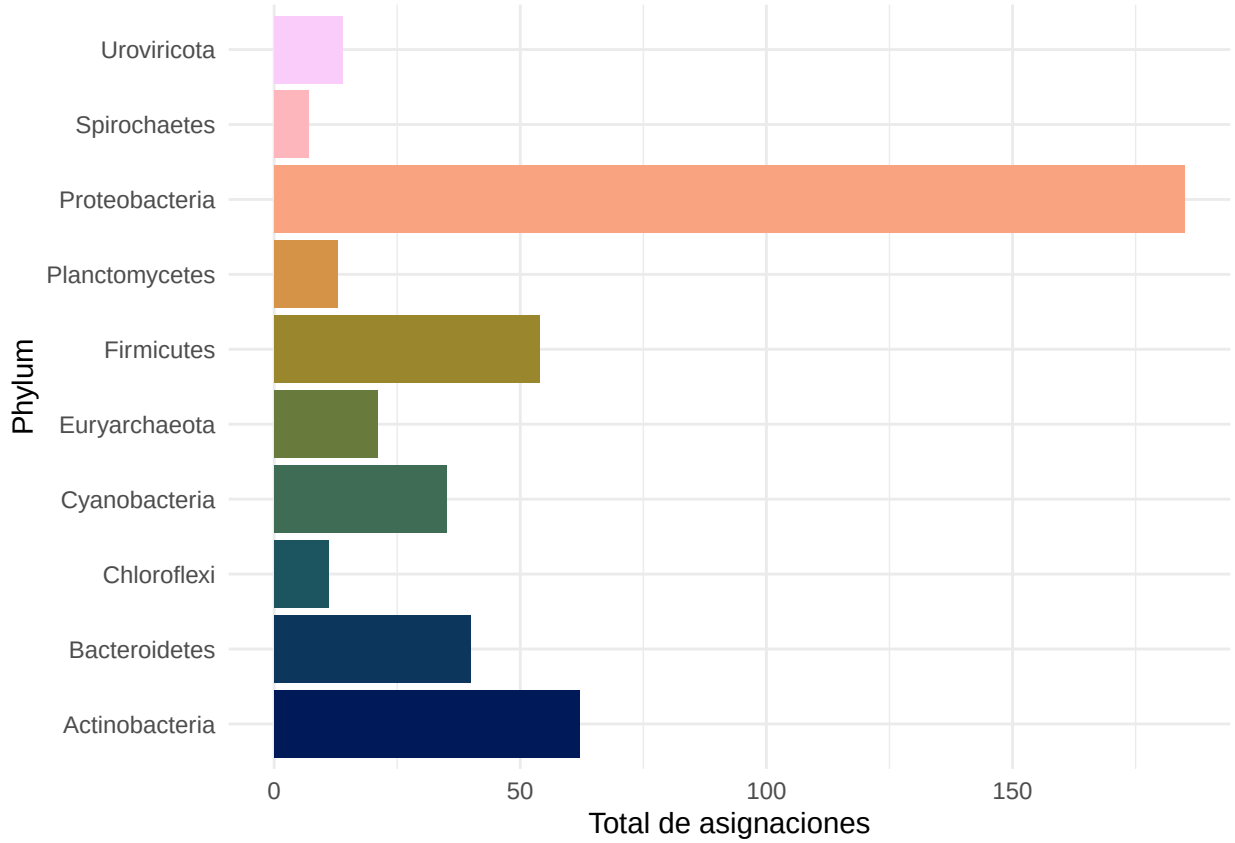
Con esta primer limpieza podemos explorar cuántos linajes distintos se asignaron a los distintos niveles taxonómicos. Checamos los tres más altos:

```
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() +
```

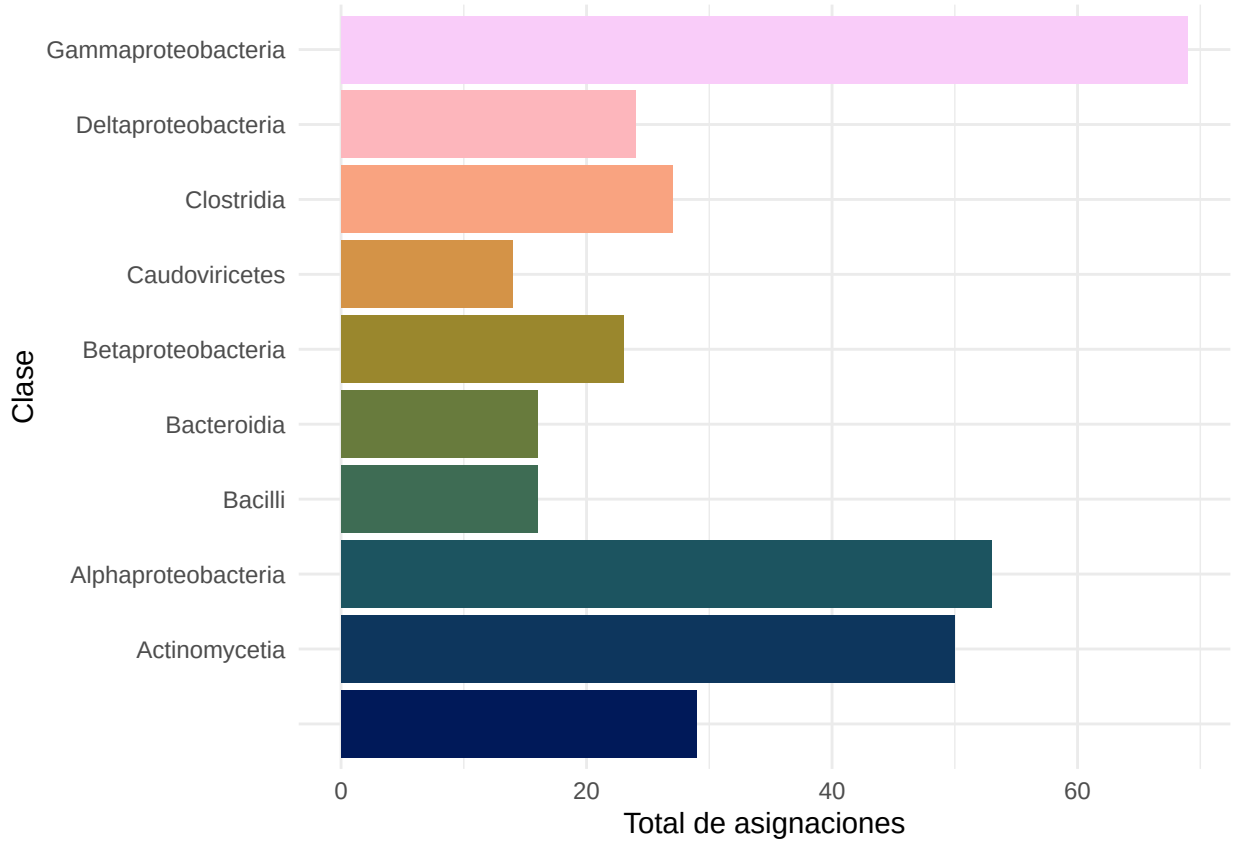
```
      theme(legend.position = "none")
```




```
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.

3 Abundancia taxonómica

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

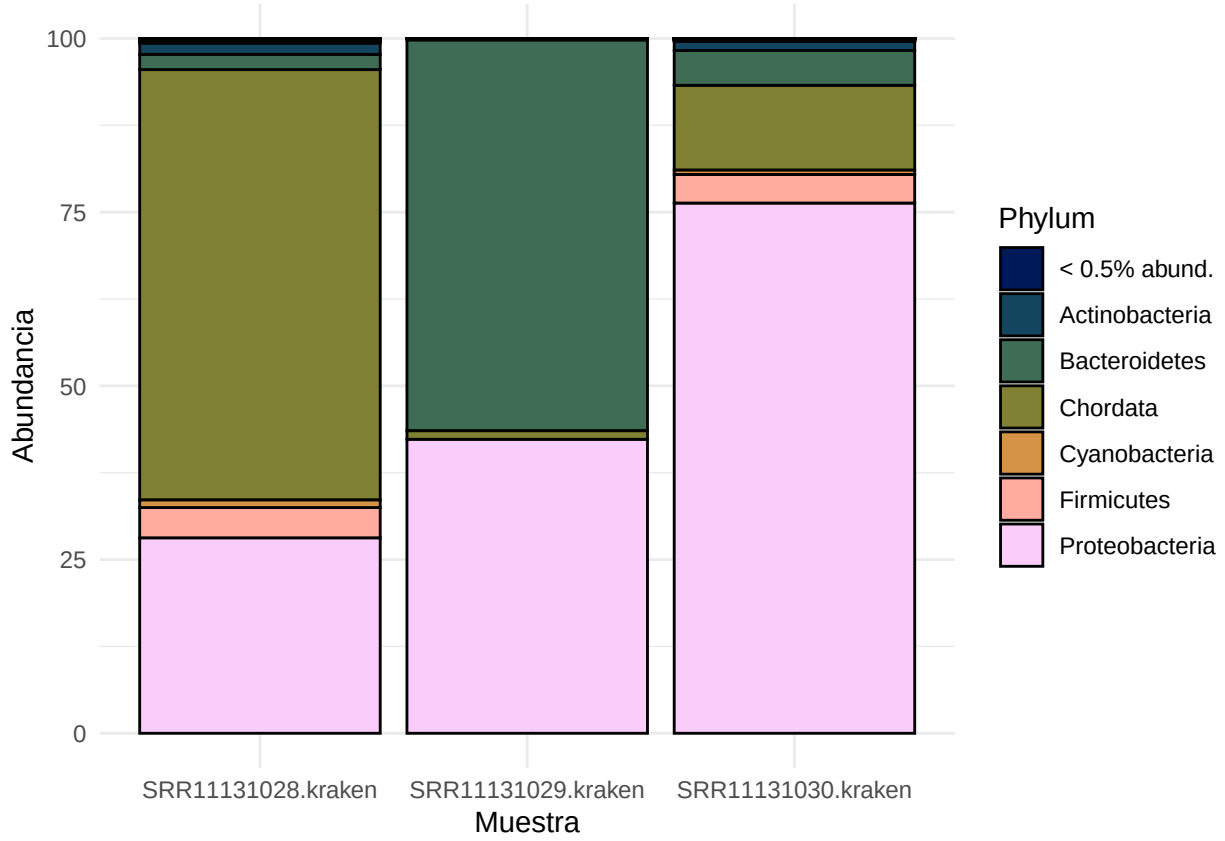
Primero generaremos el porcentaje de **abundancia** de cada taxa en cada muestra y limitaremos el estudio a nivel de **phylum**.

```
  # 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
## 9604           61.94483056       1.221219e+00        12.20927119
## 91347           1.10981453       4.379326e-06         0.05068088
## 1903409        19.46746826       0.000000e+00         0.07243951
## 543             1.44539972       2.084997e-02         0.71636115
```

```
## 1903414         0.13394806       0.000000e+00         0.07324291
## 1903411         0.08437298       7.970373e-04         0.03996894
```

```r
# 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()
```




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

```
      unique()
```

```
## # A tibble: 46 x 1
##    Phylum
##    <chr>
##  1 Spirochaetes
##  2 Firmicutes
##  3 Planctomycetes
##  4 Tenericutes
##  5 Chloroflexi
##  6 Euryarchaeota
##  7 Fusobacteria
##  8 Verrucomicrobia
##  9 Candidatus Saccharibacteria
## 10 Deinococcus-Thermus
## # ... with 36 more rows
```

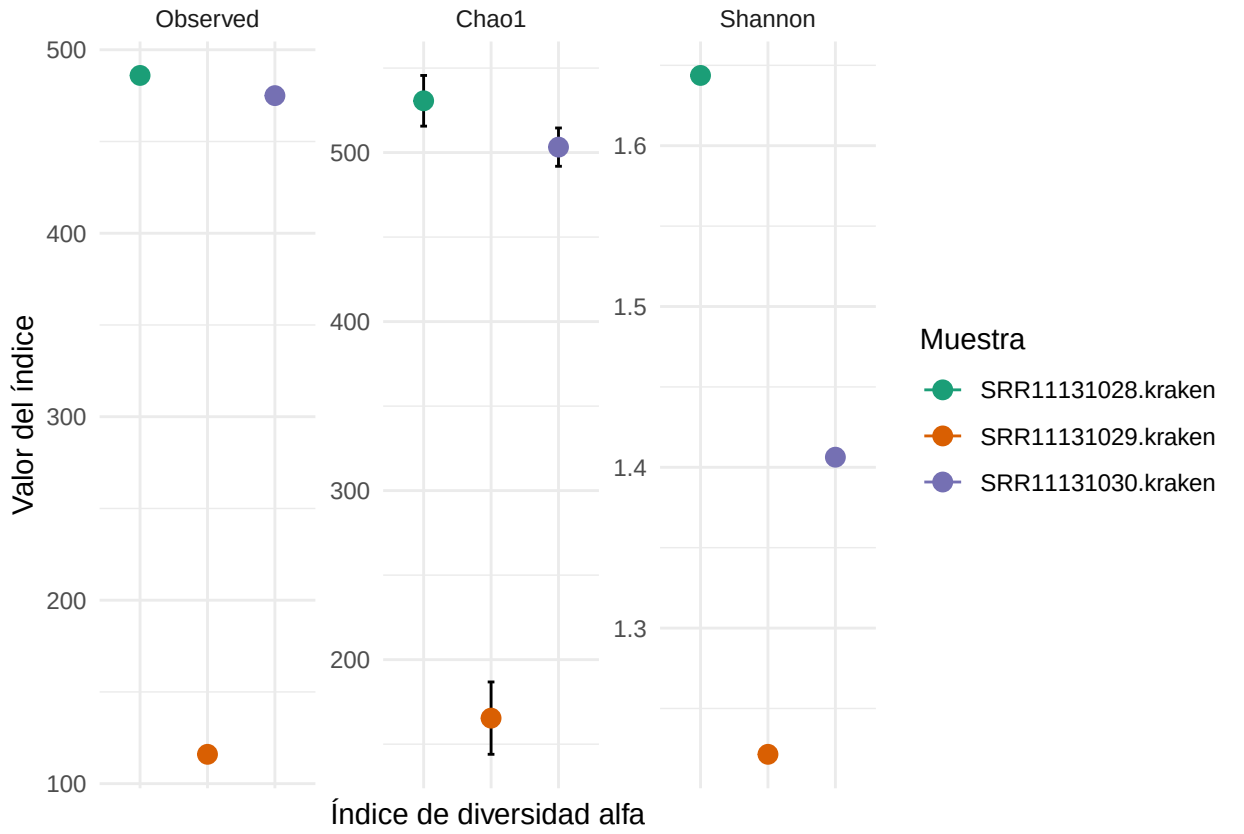
4 Análisis de diversidad

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*.

4.1 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
```

```
##              samples variable      value       se
## 1 SRR11131028.kraken Observed 486.000000       NA
## 2 SRR11131029.kraken Observed 116.000000       NA
## 3 SRR11131030.kraken Observed 475.000000       NA
## 4 SRR11131028.kraken    Chao1 530.634146 14.99520
## 5 SRR11131029.kraken    Chao1 165.400000 21.40188
## 6 SRR11131030.kraken    Chao1 503.218750 11.32118
## 7 SRR11131028.kraken  Shannon   1.643707       NA
## 8 SRR11131029.kraken  Shannon   1.221503       NA
## 9 SRR11131030.kraken  Shannon   1.406291       NA
```

4.2 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
```

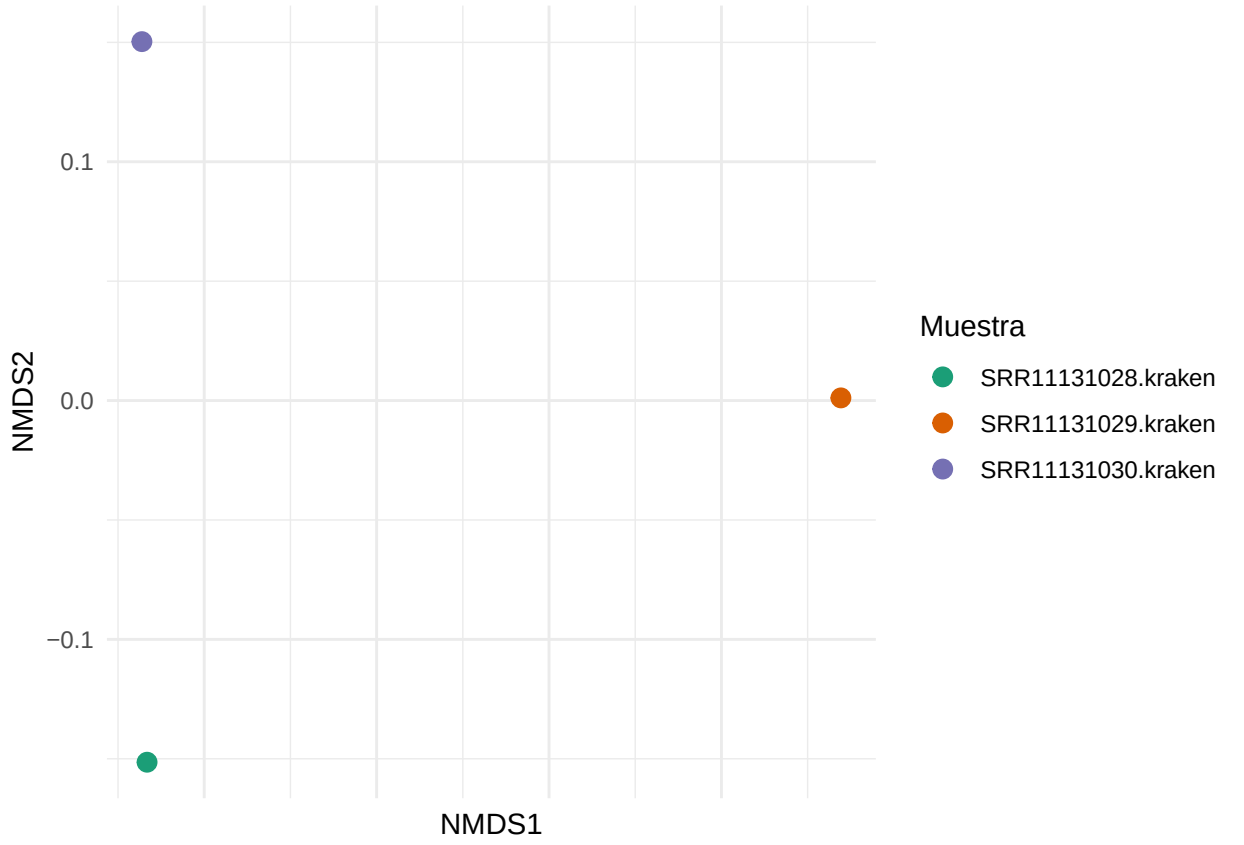
```
## Run 1 stress 0
## ... Procrustes: rmse 0.1790876  max resid 0.1949865
## Run 2 stress 0
## ... Procrustes: rmse 0.1126528  max resid 0.1271548
## Run 3 stress 0
## ... Procrustes: rmse 0.2019356  max resid 0.2414435
## Run 4 stress 0
## ... Procrustes: rmse 0.09543299  max resid 0.123563
## Run 5 stress 0
## ... Procrustes: rmse 0.09105407  max resid 0.1118982
## Run 6 stress 0
## ... Procrustes: rmse 0.1773901  max resid 0.1926107
## Run 7 stress 0
## ... Procrustes: rmse 0.263194  max resid 0.3685927
## Run 8 stress 0
## ... Procrustes: rmse 0.1885389  max resid 0.2448888
## Run 9 stress 0
## ... Procrustes: rmse 0.1839234  max resid 0.2297054
## Run 10 stress 0
## ... Procrustes: rmse 0.2050434  max resid 0.2353887
## Run 11 stress 0
## ... Procrustes: rmse 0.2403072  max resid 0.3344435
## Run 12 stress 0
## ... Procrustes: rmse 0.180131  max resid 0.2462415
## Run 13 stress 0
## ... Procrustes: rmse 0.1422673  max resid 0.1750746
## Run 14 stress 0
## ... Procrustes: rmse 0.1342506  max resid 0.1554051
## Run 15 stress 0
## ... Procrustes: rmse 0.2590333  max resid 0.3628763
## Run 16 stress 0
## ... Procrustes: rmse 0.07745322  max resid 0.08998643
## Run 17 stress 0
## ... Procrustes: rmse 0.05761986  max resid 0.0724357
## Run 18 stress 0
## ... Procrustes: rmse 0.1842236  max resid 0.2006978
## Run 19 stress 0
## ... Procrustes: rmse 0.223043  max resid 0.3094798
## Run 20 stress 0
## ... Procrustes: rmse 0.1361869  max resid 0.1716836
## *** No convergence -- 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`.