Proyecto 2 Grafico, DADA2 y Secuenciación

Code

Hide

```
library(tidyverse)
```

```
── Attaching core tidyverse packages ──────────────────── tidyverse 2.0.0 ──
✓ dplyr     1.1.3     ✓ readr     2.1.4
✓ forcats   1.0.0     ✓ stringr   1.5.0
✓ ggplot2   3.4.4     ✓ tibble    3.2.1
✓ lubridate 1.9.3     ✓ tidyr     1.3.0
✓ purrr     1.0.2     ── Conflicts ────────────────────────────── tidyverse_conflicts() ──
✖ dplyr::filter() masks stats::filter()
✖ dplyr::lag()    masks stats::lag()
ℹ Use the ]8;;http://conflicted.r-lib.org/conflicted package]8;; to force all conflicts to b
ecome errors
```

Hide

```
library(ggplot2)
library(patchwork)
library(RColorBrewer)
library(ggbreak)
```

```
ggbreak v0.1.2

If you use ggbreak in published research, please cite the
following paper:

S Xu, M Chen, T Feng, L Zhan, L Zhou, G Yu. Use ggbreak to
effectively utilize plotting space to deal with large datasets
and outliers. Frontiers in Genetics. 2021, 12:774846. doi:
10.3389/fgene.2021.774846
```

Hide

```
library(plotrix)
library(ggsignif)
```

```
Warning: package 'ggsignif' was built under R version 4.3.2
```

Hide

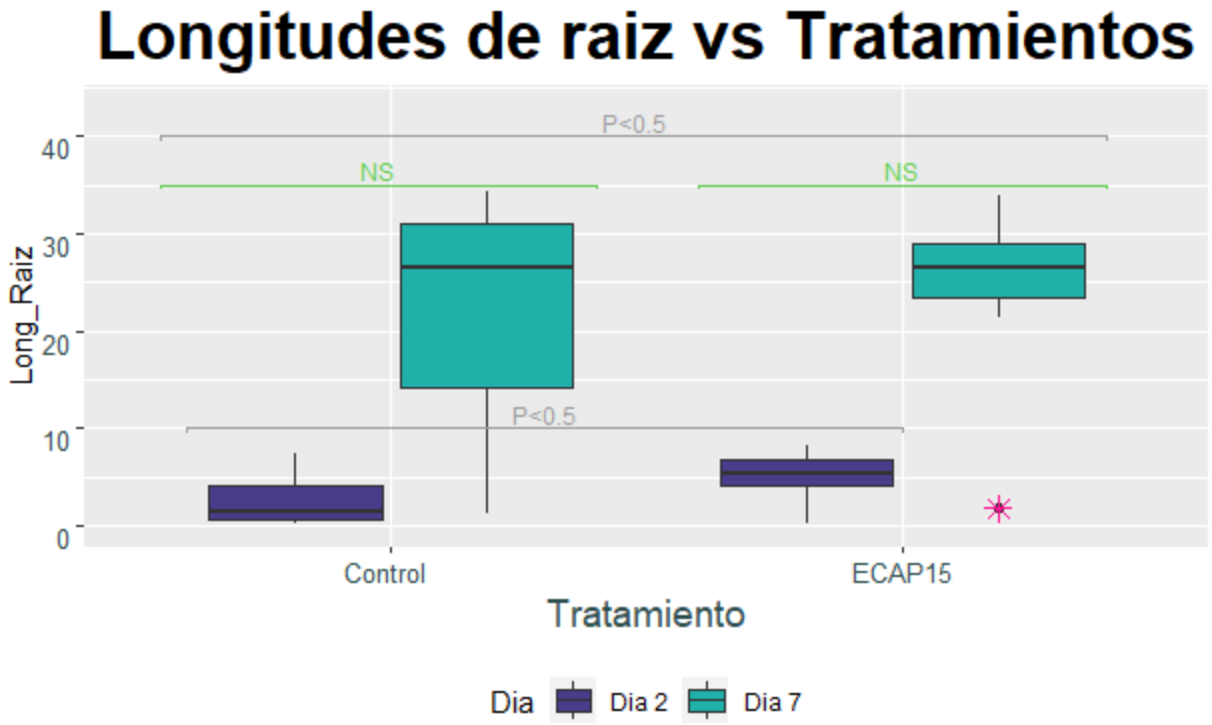
```
library(dada2)
```

```
Loading required package: Rcpp
```

Hide

```
library(dplyr)
```

Grafico agrupado por líneas con variabe del primer proyecto



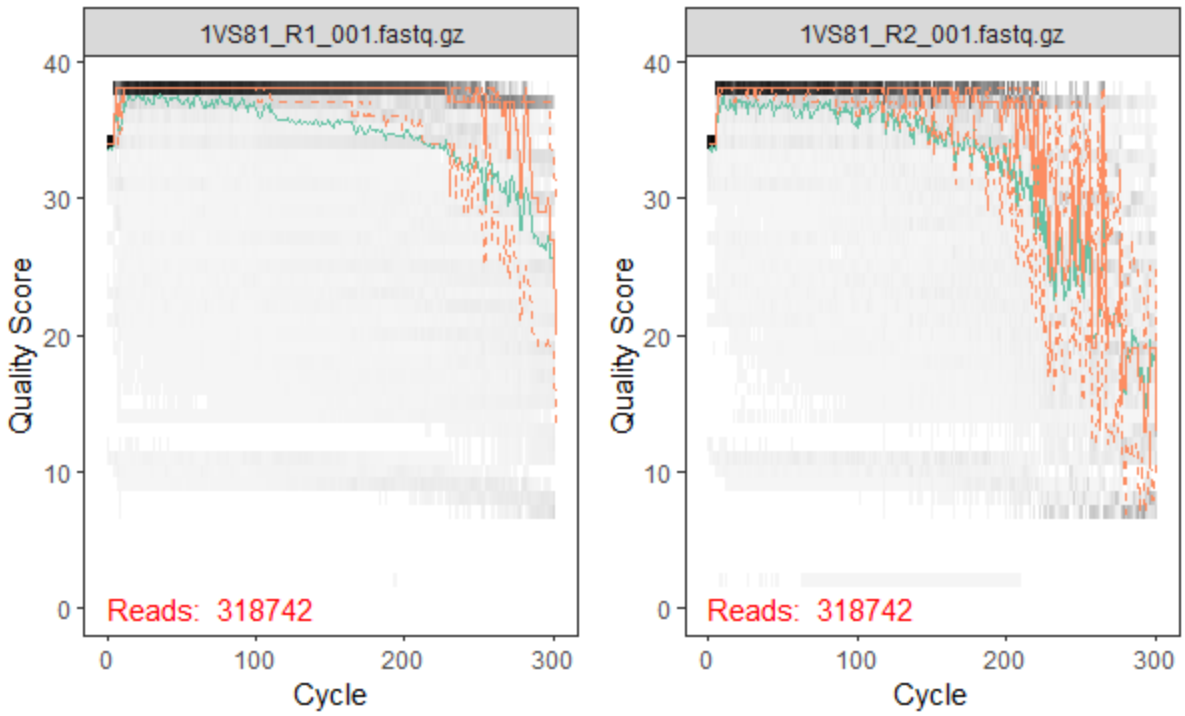
Secuenciación de muestra de VID (S81)

Hide

```
list.files(path)
```

```
[1] "1VS81_R1_001.fastq.gz" "1VS81_R2_001.fastq.gz"
[3] "filtered"
```

Inspección perfiles de calidad



Filtrar

Hide

```
# Camino a los datos filtrados

filtFs <- file.path(path, "filtered", paste0(sample.names,  "_F_filt.fastq.gz")) # forward
filtRs <- file.path(path, "filtered", paste0(sample.names, "_R_filt.fastq.gz")) # reverse

# Asignacion de los nombres de las muestras a nuestros objetos nuevos
names(filtFs) <- sample.names
names(filtRs) <- sample.names
```

Corte

Hide

```
# Guardar nuestro progreso
write.csv(out, "~/RStudio/CursoInnovak/Materiales/Conteo_reads_proyecto2_1.csv") # para guardar
la tabla
```

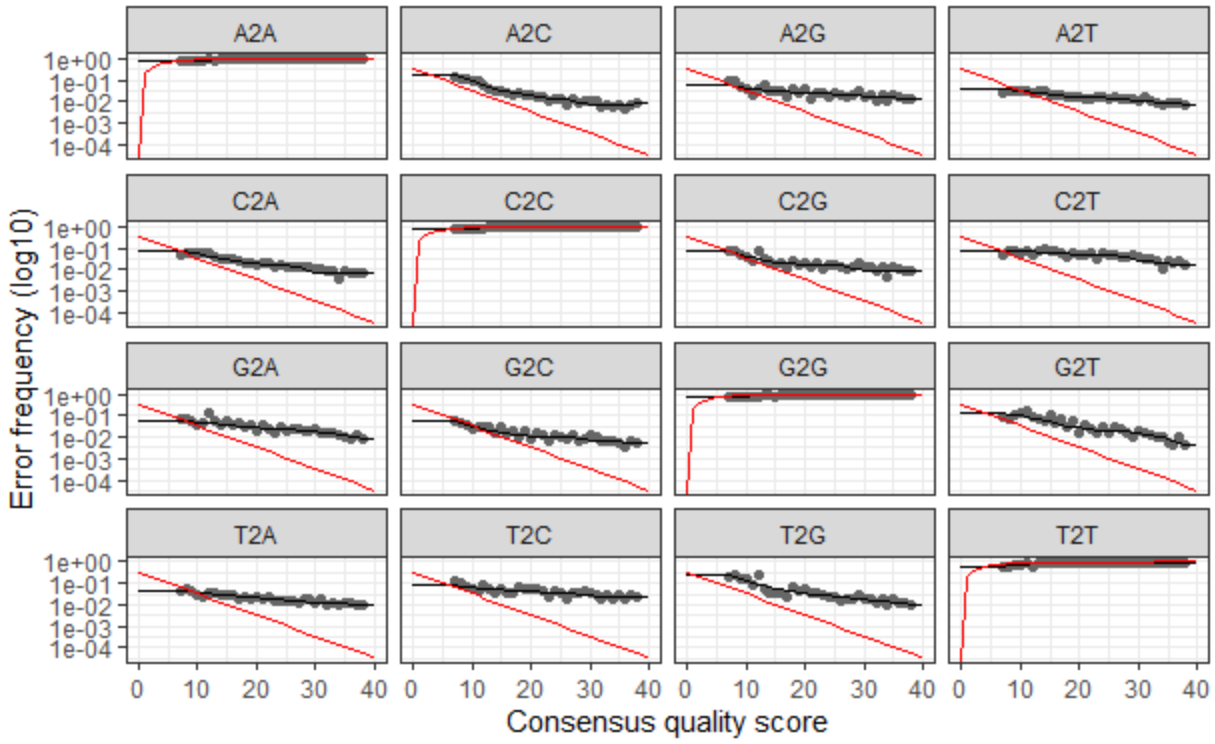
```
Error in eval(expr, p) : objeto 'out' no encontrado
```

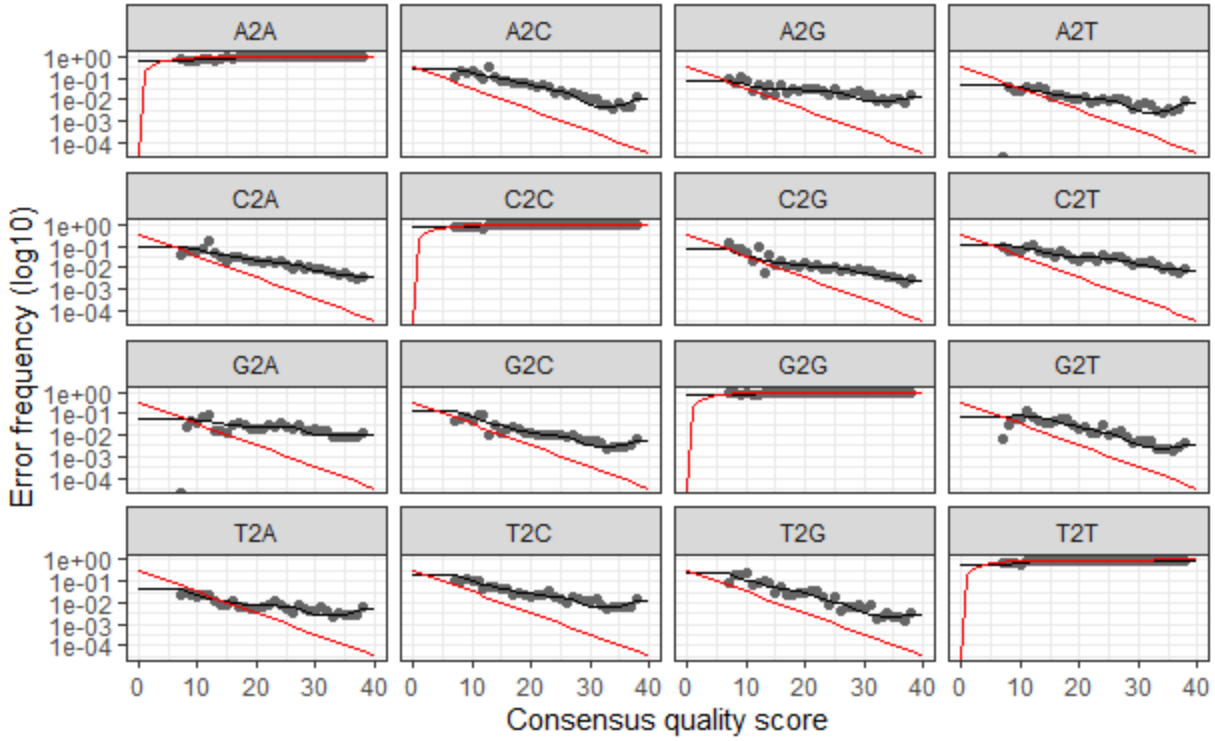
Interpretación del corte Como observamos, se decidió realizar el corte a 270,200 ya que se intentó cortar de otras maneras (250,240 - 260,240 - 270,200) y esta fué la que mejor resulto al realizar las combinaciones en cuanto a los ajustes de los errores que en este caso quedo en 5,5 sin ser tan estrictos ya que se intento tambien

con 2,2 y 2,5 pero el corte era mas pronunciado. Se podría haber modificado más veces pero por cuestiones de tiempo se decidió optar por esta opción.

Intenté apegarme a lo recomendado para el quality socre de estar lo mas cerca posible, o bien, dentro del rango 30-40 en el eje de las “y” de nuestras graficas intentando evitar/disminuir los mas errores posibles.

Tasas de error





Interpretación errores Al graficar los errores se observan las posibles combinaciones que existen en la muestra, esto nos ayuda para poder saber si hay diferencia en las bases (en este caso seria entre una misma secuencia). * Las lineas negras representan las tasas de error estimadas que basicamente son los ajustes de los datos reales pero que ya cuentan con los errores. * Las lineas rojas representan las tasas de error esperadas que son los datos reales.

Así, tendríamos en cuenta que en cada secuencia/muestra(as) que porcesemos, la tasa de error debe ser menor cuando la calidad sea mayor en estas graficas. Que si observamos, nuestras graficas nos arrojan que tenemos mayor calidad.

Interferencia de las muestras

Hide

```
# Forward
dadaFs_nopool_proy2.1 <- dada(filtFs, err=errF, multithread = TRUE,
                      pool = FALSE)
save(dadaFs_nopool_proy2.1, file = "dadaFs_nopool_proy2_1.RData")

#Reverse
dadaRs_nopool_proy2.1 <- dada(filtRs, err=errR, multithread = TRUE,
                      pool = FALSE)
save(dadaRs_nopool_proy2.1, file = "dadaRs_nopool_proy2.RData")

load("dadaFs_nopool_proy2_1.RData")
load("dadaRs_nopool_proy2.RData")
```

Interpretación de las interferencias En relación a la interferencia de las muestras, este nos arrojó cuantos errores fueron descartados para dejar limpia nuestra secuecia ya filtrada.

Union de las lecturas forwards y reverse

Hide

```
mergers_proy2.1 <- mergePairs(dadaFs_nopool_proy2.1, filtFs, dadaRs_nopool_proy2.1, filtRs, verb
ose = TRUE)
save(mergers_proy2.1, file = "mergers_proy2.RData")

# por si cerraron su sesion
load("mergers_proy2.RData")
```

Interpretación de la unión En este paso, lo que realizamos fue la unión de nuestra secuencia (forward y reverse). En dicha unión, aunque estemos uniendo lo ya filtrado, nos apareceran pares que seran rechazados si no se superponieron lo suficiente o si cuentan con muchos mismatches.

Fue hasta este paso, que me di cuenta al momento de correr el código de cuantas secuencias eran las que te quedaban y cuantas te quitaba, por lo que, fue aquí donde tomé la decisión en varias ocaciones de modificar mi corte para evitar la mayor perdida de unión de mi secuencia.

Hacer tabla de secuencias

Hide

```
# Checar la longitud de todas las secuencias
table(nchar(getSequences(seqtab_proy2.1)))
```

```
 281  295  296  298  333  354  355  373  390  396  402  403  410  411
   1    1    4   29    3    1    1    2    1    1    1    1    1    1
 417  418  420  422  433  435  436  437  438  439  440  441  442  443
   1    1    6    1    2    2    2   13   49 1159 7934 1083  599  171
 444  445  446  447  448  449  451  452  453  454  455  456  457  458
 328 2911  161  184   24   14   19    9   55  431   52   37  119  311
```

Interpretación Después de realizar la unión, se procede a crear una tabla de las secuencias en donde podremos ver la variación de la secuancia del amplicón la cual nos ayudara a tener mejor resulución a la hora de aplicar nuestro código para la taxonomía, así como nuestras abundancias en cada una.

Quitar quimeras

Hide

```
seqtab.nochim_proy2.1 <- removeBimeraDenovo(seqtab_proy2.1, method = "consensus",
                                    multithread=TRUE, verbose = TRUE)
#Basado en esto (11765/15726*100) 74.81% de mis secuencias son quimeras

save(seqtab.nochim_proy2.1, file = "seq_conteos_proy2.RData")
load("seq_conteos_proy2.RData")

# Comparar esta tabla con la original que incluye quimeras
dim(seqtab.nochim_proy2.1)
# Incluyendo abundancias
sum(seqtab.nochim_proy2.1)/sum(seqtab_proy2.1) # porcentaje de secuencias no quimericas que se m
antuvieron

# Tomando en cuenta abundancias en realidad mantuvimos 48% de nuestras lecturas
```

Interpretación Las quimeras son secencias de ADN que se unen cuando estas no deberían hacerlo y las cuales deben ser eliminadas. Por suerte tenemos el código que nos identifica cuales de las uniones dadas son quimeras y nos ayuda a desasernos de ellas.

En nuestro código tenemos como resultado el porcentaje de secuecias que son y no son quimeras que se obtuvieron gracias a las abundancias.

Seguimiento del proceso

Hide

```
# Primero crearemos una funcion
getN <- function(x) sum(getUniques(x)) #esta funcion va a sumar el nuemro de valores unicos dent
ro de x "es importante mencionar que las funciones no se deben de modificar"

#Tabla
track_proy2.1 <- cbind(out, #Paso 1: filtrado y corte, Paso 2:quitar errores
               getN(dadaFs_nopool_proy2.1),
               getN(dadaRs_nopool_proy2.1), #Paso 3: denoising
               getN(mergers_proy2.1), #Paso 4: unir muestras
               rowSums(seqtab.nochim_proy2.1)) #Paso 5: quitar quimeras

# Nombramos nuestras filas y columnas
colnames(track_proy2.1) <- c("input", "filtered", "denoisedF", "denoisedR", "merged", "nonchim")

# Guardamos esta tabla
write.csv(track_proy2.1, "~/RStudio/CursoInnovak/Proyecto_2_Grafico-DADA2-Secuenciacion/Seguimie
nto_dada_proy2_1.csv")
```

Interpretación Creamos una nueva tabla llamada track_proy2.1, en este caso cambiamos la funcion suppply por la getN ya que; supply es para cuando tenemos más de una muestra y como en nuestro poryecto solo estamos trabajando con una sola muestra, pues hacemos el cambio de nuestro código con getN y quitamos del parentesis (“, getN”…“). Con esto podemos proseguir para asignar taxonomia

Asignar Taxonomia

Hide

```
taxa_proy2 <- assignTaxonomy(seqtab.nochim_proy2,
                        "~/RStudio/CursoInnovak/Secuenciación/Taxa/silva_nr99_v138.1_train_set.f
a.gz", multithread = TRUE)


# Tabla con especies
taxa_proy2_esp_2 <- addSpecies(taxa_proy2, "~/RStudio/CursoInnovak/Secuenciación/Taxa/silva_spec
ies_assignment_v138.1.fa.gz")

save(taxa_proy2_esp_2, file = "taxa_ch_proy2_1_esp.RData")
```

Interpretación En este caso, utilizamos la base de datos de SILVA, gracias a esta base ahora sabemos cuales microorganisos se encuentran en nuestra muestra. Para obtener la asignación, se toma en cuenta la tabla de conteo de las secuencias y la base SILVA utilizando un intervalo de confianza ya establecido.

En este caso, decidí agregar la especie solo para tener mi proyecto mas completo. Aunque por lo visto, no tuve exito con la asiganación de la especie en mis microorganismos encontrados en mi muestra.