UNIVERSIDAD DE GRANADA

Práctica 5 de Periféricos y Dispositivos de Interfaz Humana

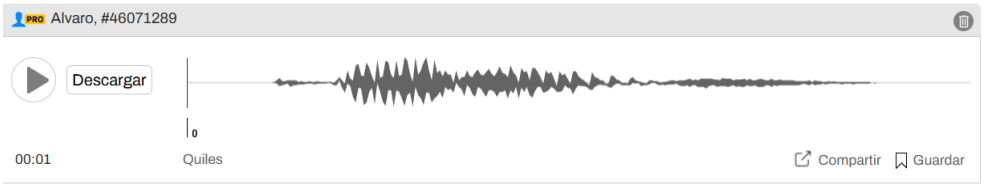
Curso 2024-2025

Francisco Quiles Ramírez

ÍNDICE:

1- Crear dos ficheros de sonido

Para poder usar dos ficheros de sonido con mi nombre se ha usado la página https://speechgen.io/es/ donde se ha aplicado una voz en español y luego se descarga el archivo en wav.



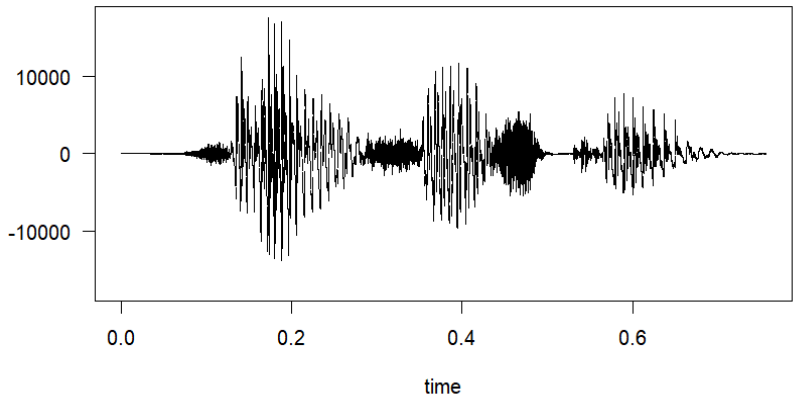
2- Leer los sonidos y dibujar la forma de onda

Para este apartado lo primero que se hace es cargar las librerías y con setwd indicar las rura donde se encuentran los archivos wav grabados anteriormente, ahora usamos readWave para leer nuestros archivos y asignarlos a una variable llamada nombre y otra llamada apellido, con esto ya tendremos los sonidos leídos, para dibujar la onda se usa plot, donde pondremos el archivo wave que queremos y la longitud, en este caso desde 1 hasta la longitud del archivo.

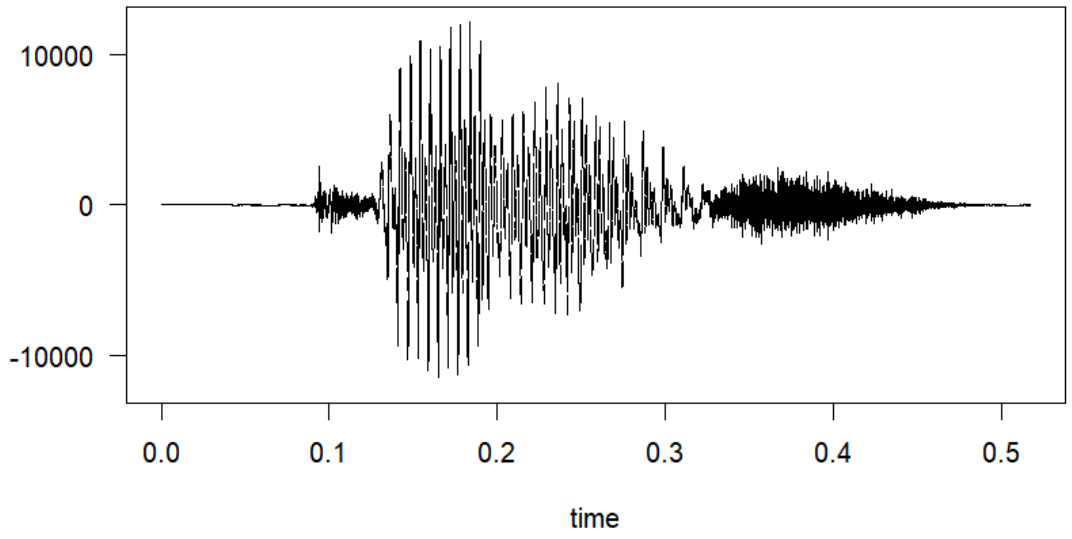
```
library(tuneR)
library(seewave)
library(audio)


setwd("/Users/franc/Desktop/Cuarto - 2ºCuatri/Perifericos/PDIH/p5")

nombre <- readWave("nombreFrancisco.wav")
apellido <- readWave("apellidoQuiles.wav")

plot(extractWave(nombre, from = 1, to = length(nombre@left)))
plot(extractWave(apellido, from = 1, to = length(apellido@left)))
```

```
> setwd("/Users/franc/Desktop/Cuarto - 2ºCuatri/Perifericos/PDIH/p5")
> nombre <- readWave("nombreFrancisco.wav")
> apellido <- readWave("apellidoQuiles.wav")
> plot(extractWave(nombre, from = 1, to = length(nombre@left)))
> plot(extractWave(apellido, from = 1, to = length(apellido@left)))
> View(apellido)
>
```

Ondas del nombre:



Ondas del apellido:



3- Información de las cabeceras de sonido

Para poder ver información de los archivos de nombre y apellido se usa la orden str, que muestra que ambos sonidos están en formato WAV, con una sola pista (mono, ya que @stereo = FALSE), una frecuencia de muestreo de 48.000 Hz y 16 bits por muestra. El archivo nombre contiene 36.335 muestras y apellido 24.815. Estos datos confirman que los sonidos se han cargado correctamente y están listos para ser procesados.

```
14  str(nombre);
15  str(apellido);
16
16:1  (Top Level)
Console  Terminal  Background Jobs
R 4.5.0 · C:/Users/franc/Desktop/Cuarto - 2ºCuatri/Perifericos/PDIH/p5/
> str(apelido);
Error: objeto 'apelido' no encontrado

> str(nombre);
Formal class 'Wave' [package "tuneR"] with 6 slots
  ..@ left     : int [1:36335] 0 0 0 0 0 0 0 0 0 0 ...
  ..@ right    : num(0)
  ..@ stereo   : logi FALSE
  ..@ samp.rate: int 48000
  ..@ bit      : int 16
  ..@ pcm      : logi TRUE
> str(apellido);
Formal class 'Wave' [package "tuneR"] with 6 slots
  ..@ left     : int [1:24815] 0 0 0 0 0 0 0 0 0 0 ...
  ..@ right    : num(0)
  ..@ stereo   : logi FALSE
  ..@ samp.rate: int 48000
  ..@ bit      : int 16
  ..@ pcm      : logi TRUE
>
```

4- Unir ambos sonidos

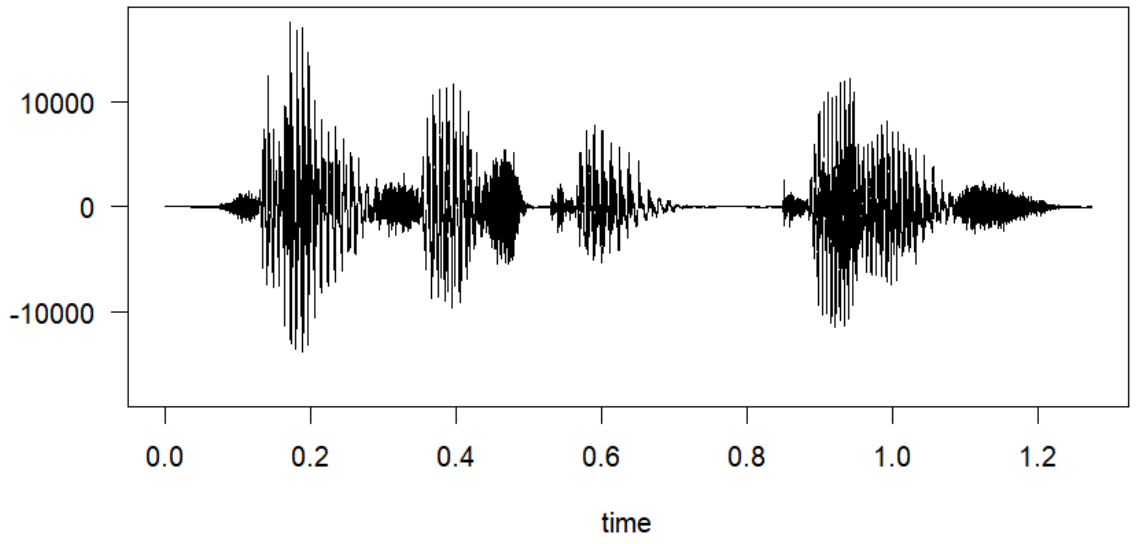
Se usa la función pastew que unirá el nombre y apellido en formato wave.

```
sonido_unido <- pastew(apellido, nombre, output = "Wave")
```

5- Dibujar y reproducir el sonido resultante

Al igual que antes se usa plot para representar la onda del sonido resultante y con listen podemos escuchar el audio de los dos juntos.

```
plot(extractWave(sonido_unido, from = 1, to = length(sonido_unido@left)))
listen(sonido_unido)
```



6- Almacenar el sonido resultante

Mediante writeWave guardamos el sonido con el nombre y apellidos en un archivo llamado unido.wav

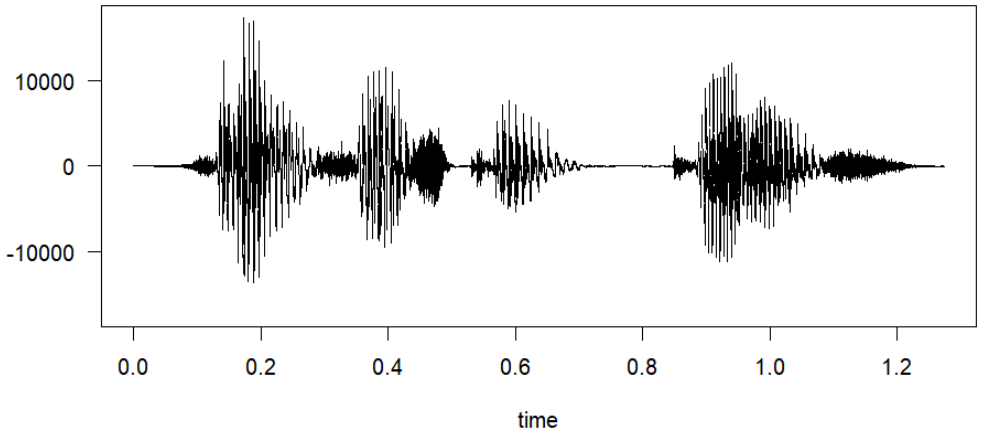
```
24  writeWave(sonido_unido, "unido.wav")
25
```

7- Aplicar filtro de frecuencia

Este fragmento de código aplica un filtro de frecuencia al sonido previamente unido. La función bwfilter() de la librería seewave se utiliza para eliminar todas las frecuencias comprendidas entre 10.000 Hz y 20.000 Hz del archivo sonido_unido. Para ello, se indica la

frecuencia de muestreo real del sonido con f = sonido_unido@samp.rate, y se establece bandpass = FALSE para que el filtro actúe como un filtro de rechazo (eliminando ese rango de frecuencias). El resultado se guarda en un nuevo objeto llamado filtrado, que posteriormente se exporta como archivo WAV bajo el nombre filtrado.wav con writeWave(). Finalmente, se reproduce el sonido filtrado con listen(filtrado) para comprobar auditivamente el efecto del filtro.

```
filtrado <- bwfilter(sonido_unido, f = sonido_unido@samp.rate,
                     from = 10000, to = 20000,
                     bandpass = FALSE, output = "Wave")

writeWave(filtrado, "filtrado.wav")
listen(filtrado)
```

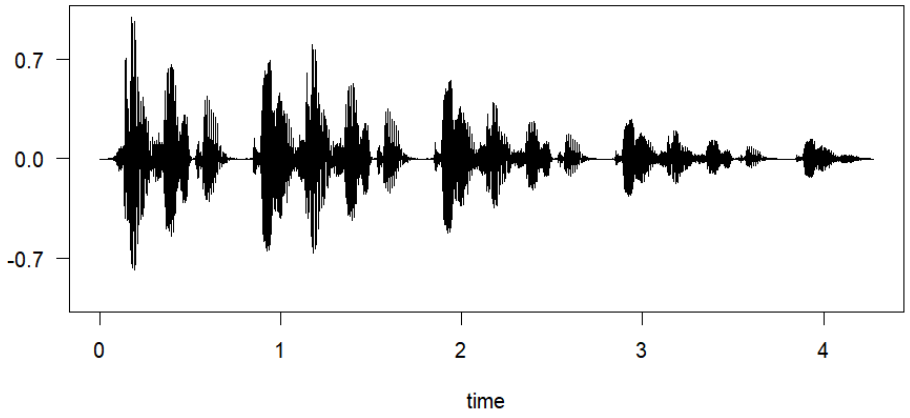


8- Aplicar eco

Este bloque de código comienza leyendo el archivo de sonido basico.wav, que contiene la unión del nombre y el apellido grabados previamente. A continuación, se le aplica un efecto de eco utilizando la función echo() de la librería seewave. Este efecto genera repeticiones del sonido original con retardos de 1, 2 y 3 segundos, y con amplitudes decrecientes (0.8, 0.4 y 0.2), simulando así un eco realista. El resultado se guarda como un nuevo archivo llamado eco.wav. Después, se utiliza la función revw() para invertir el sonido, es decir, hacer que se reproduzca desde el final hacia el principio, lo que da lugar a un efecto de "reverso temporal". Este nuevo sonido se guarda como alreves.wav. Finalmente, se reproducen ambos sonidos con listen() para comprobar auditivamente los efectos aplicados. Este proceso cumple con los requisitos del ejercicio 8 de la práctica, mostrando la aplicación de dos transformaciones sonoras consecutivas: eco e inversión temporal.

```r
# Ej8
basico <- readWave("basico.wav")
eco <- echo(basico,
            f = basico@samp.rate,
            amp = c(0.8, 0.4, 0.2),
            delay = c(1, 2, 3),
            output = "Wave")

writeWave(eco, "eco.wav")

alreves <- revw(eco, output = "Wave")

writeWave(alreves, "alreves.wav")


listen(eco)
listen(alreves)
```

Onda de eco:



Onda de alreves: