PRÁCTICA 4 PDIH
MANUEL FERNÁNDEZ LA-CHICA

EJERCICIO 1 - Leer dos ficheros de sonido (WAV o MP3) de unos pocos segundos de duración cada uno.

```
# establecemos el directorio de trabajo
setwd("/Users/manferlac/Desktop/PDIH/S6/S6-varios-sonidos")
# establecemos el reproductor para los sonidos
setWavPlayer('/usr/bin/afplay')

# Ejercicio 1 - Leemos los ficheros de sonido
oveja <- readWave('oveja.wav')
oveja
perro  <- readWave('perro.wav')
perro
```

La función *setwd()* se utiliza para establecer el directorio de trabajo donde se encuentren nuestros archivos de sonido.

La función *setWavPlayer()* se utiliza para establecer el reproductor de sonido.

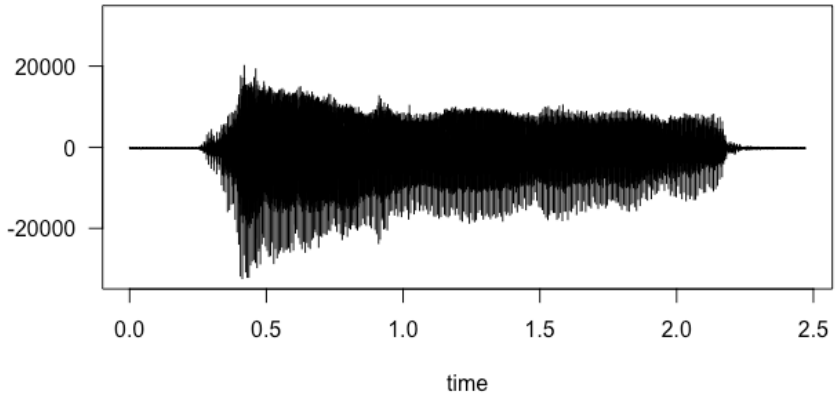
Leemos dos sonidos con la función *readWave()* en este caso al ser archivos de formato .wav

EJERCICIO 2 - Dibujar la forma de onda de ambos sonidos.

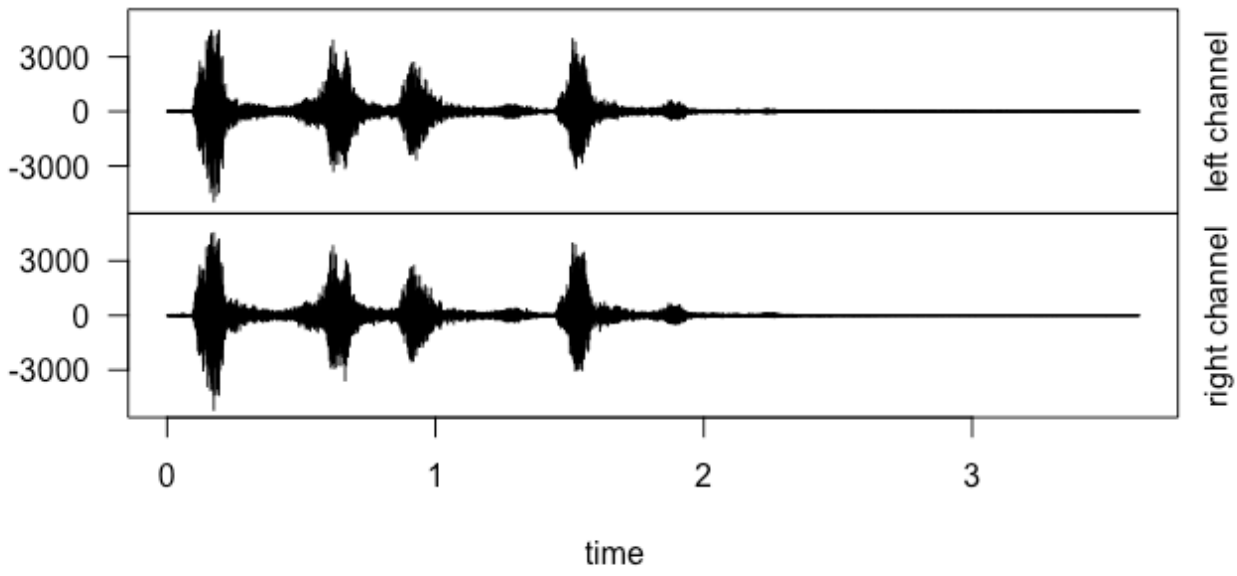
```
# Ejercicio 2 - Dibujamos la forma de onda de ambos sonidos
plot( extractWave(oveja, from = 1, to = 19764) )
plot( extractWave(perro, from = 1, to = 159732) )
```

Usamos la función plot() para obtener las ondas de los sonidos, *extractWave()* lo utilizamos para extraer el contenido de dichos sonidos, desde el comienzo hasta el final, el final lo obtenemos en el ejercicio anterior donde al mostrar la información de cabecera nos indica el total del archivo. También se podría usar la función length() para obtener el total del sonido.

Onda del sonido de la oveja:



Onda del sonido del perro:



EJERCICIO 3 - Obtener la información de las cabeceras de ambos sonidos.

```
# Ejercicio 3 - Mostramos la información de las cabeceras de ambos sonidos
str(oveja)
str(perro)
```

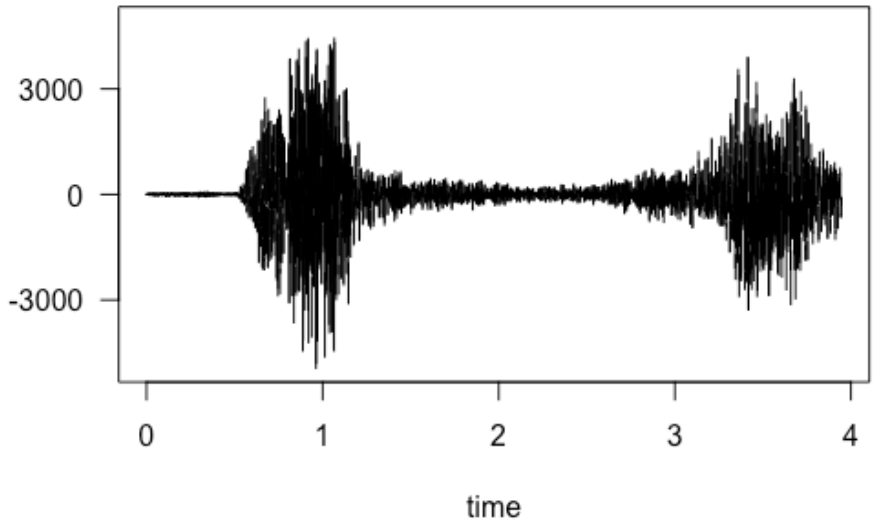
```
> # Ejercicio 3 - Mostramos la información de las cabeceras de ambos sonidos
> str(oveja)
Formal class 'Wave' [package "tuneR"] with 6 slots
  ..@ left     : int [1:19764] -7 -206 -250 -225 -237 -234 -231 -230 -231 -230 ...
  ..@ right    : num(0)
  ..@ stereo   : logi FALSE
  ..@ samp.rate: int 8000
  ..@ bit      : int 16
  ..@ pcm      : logi TRUE

> str(perro)
Formal class 'Wave' [package "tuneR"] with 6 slots
  ..@ left     : int [1:159732] 0 0 0 0 0 0 0 0 1 1 ...
  ..@ right    : int [1:159732] 0 0 0 0 0 0 0 0 1 1 ...
  ..@ stereo   : logi TRUE
  ..@ samp.rate: int 44100
  ..@ bit      : int 16
  ..@ pcm      : logi TRUE
```

EJERCICIO 4 - Unir ambos sonidos en uno nuevo

```
# Ejercicio 4 - Unimos ambos sonidos en uno nuevo
sonidoNuevo <- pastew(oveja, perro, output = "Wave")
# mostramos la info del sonido nuevo
sonidoNuevo
```

EJERCICIO 5 - Dibujar la forma de onda de la señal resultante

```
# Ejercicio 5 - dibujamos la onda de la señal nueva
plot( extractWave(sonidoNuevo, from = 1, to = 31573) )
```



EJERCICIO 6 - Pasarle un filtro de frecuencia para eliminar las frecuencias entre 10000Hz y 20000Hz

```
# Ejercicio 6 - Pasamos el filtro de frecuencia
f <- sonidoNuevo@samp.rate    #44100
filtro <- bwfilter(sonidoNuevo, f = f, channel = 1, n = 1, from = 1000, to = 4000, bandpass = TRUE, listen = FALSE, output = "Wave")
```

He utilizado estas frecuencias porque con las descritas en el problema me resultaba este fallo:

```
> filtro <- bwfilter(sonidoNuevo, f = f, channel = 1, n = 1, from = 0, to = 20000, bandpass = TRUE, listen = FALSE, output = "Wa
ve")
Error in butter.default(n = n, W = c(from, to)/(f/2), type = type) :
  butter: critical frequencies must be in (0 1)
>
```

EJERCICIO 7 - Almacenar la señal obtenida como un fichero WAV denominado “mezcla.wav”

```
# Ejercicio 7 - Almacenar la señal en un fichero wav
writeWave(filtro, file.path("mezcla.wav"))
```

EJERCICIO 8 - Cargar un nuevo archivo de sonido, aplicarle eco y a continuación darle la vuelta al sonido. Almacenar la señal obtenida como un fichero WAV denominado “alreves.wav”

```
# Ejercicio 8 - Cargar un nuevo archivo de sonido, aplicarle eco y a continuación darle la
gato <- readMP3('gato.mp3')
gato_con_eco <- echo(gato,f=22050,amp=c(0.8,0.4,0.2),delay=c(1,2,3),output = "Wave")
gato_con_eco@left <- 10000 * gato_con_eco@left
gato_alreves <- revw(gato_con_eco,output = "Wave")
writeWave(gato_alreves,file.path("alreves.wav"))
```

1ªlinea -> Leemos el archivo MP3 del gato

2ªlinea -> Aplicamos el eco al sonido del gato

3ªlinea -> Ajustamos el rango de amplitud

4ªlinea -> Establecemos el sonido al revés

5ªlinea -> Guardamos el nuevo sonido en un archivo llamado “alreves.wav”