

PDIH-Seminario Sonido-R

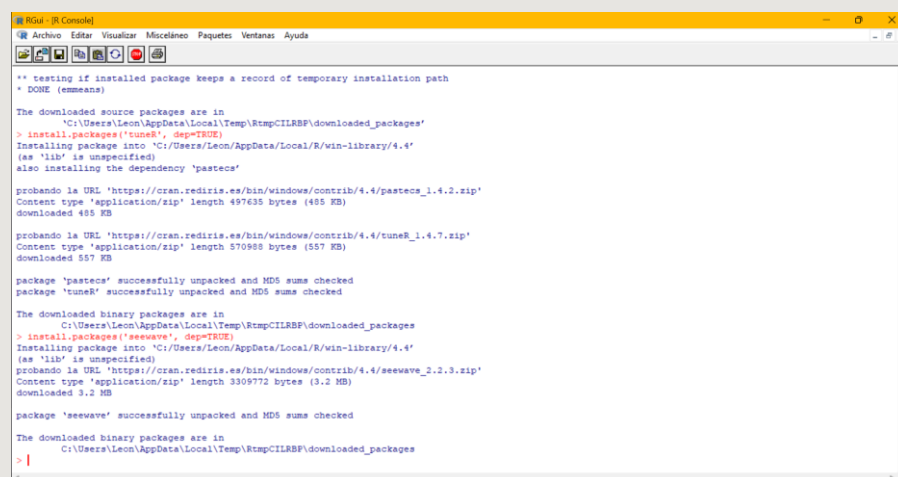
León Corbacho Rodríguez 4º 21/05/2024

1. Estudiar la estructura de un archivo de sonido WAV y aprender a manejar sonido desde R. A continuación realizará el script propuesto en la sección anterior de este guión y comprobará su correcto funcionamiento ejecutándolo en el entorno de RStudio.

Siguiendo la guía para la realización de la práctica antes de meternos en el sonido. Tras instalarnos correctamente R-Studio, debemos descargar las librerías:

TuneR: `install.packages('tuneR', dep=TRUE)`

Seewave: `install.packages('seewave', dep=TRUE)`



```
R Console

** testing if installed package keeps a record of temporary installation path
* DONE (emmeans)

The downloaded source packages are in
  'C:\Users\Leon\AppData\Local\Temp\RtmpCILRBP\downloaded_packages'
> install.packages('tuneR', dep=TRUE)
Installing package into 'C:\Users\Leon\AppData\Local\R\win-library\4.4'
(as 'lib' is unspecified)
also installing the dependency 'pastecs'

probando la URL 'https://cran.rediris.es/bin/windows/contrib/4.4/pastecs_1.4.2.zip'
Content type 'application/zip' length 497635 bytes (485 KB)
downloaded 485 KB

probando la URL 'https://cran.rediris.es/bin/windows/contrib/4.4/tuneR_1.4.7.zip'
Content type 'application/zip' length 570988 bytes (557 KB)
downloaded 557 KB

package 'pastecs' successfully unpacked and MD5 sums checked
package 'tuneR' successfully unpacked and MD5 sums checked

The downloaded binary packages are in
  C:\Users\Leon\AppData\Local\Temp\RtmpCILRBP\downloaded_packages
> install.packages('seewave', dep=TRUE)
Installing package into 'C:\Users\Leon\AppData\Local\R\win-library\4.4'
(as 'lib' is unspecified)
probando la URL 'https://cran.rediris.es/bin/windows/contrib/4.4/seewave_2.2.3.zip'
Content type 'application/zip' length 3309772 bytes (3.2 MB)
downloaded 3.2 MB

package 'seewave' successfully unpacked and MD5 sums checked

The downloaded binary packages are in
  C:\Users\Leon\AppData\Local\Temp\RtmpCILRBP\downloaded_packages
> |
```

Después en un nuevo script añadimos al principio de todas las librerías que se van a utilizar:

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

Tras esto realizamos las diferentes pruebas para las funciones de R que tienen esta librería:

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

En el ejemplo usamos los sonidos perro.wav y gato.mp3 que los cargamos usando la función:

```
-readWave('<archivo>.wav') o
- readMP3('<archivo>.mp3')
```

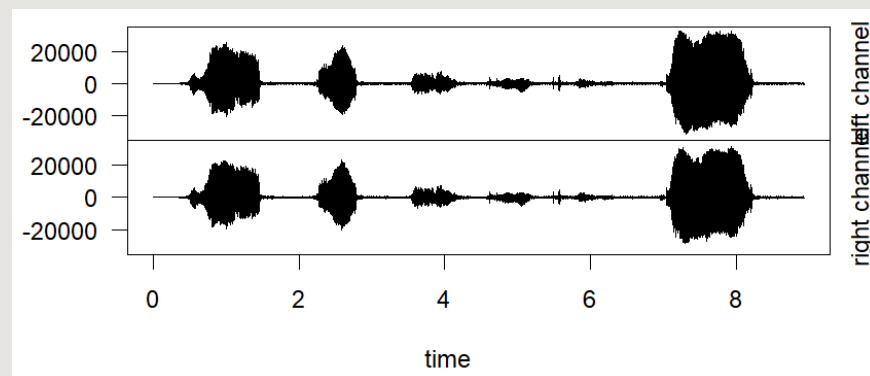
```
#Cargar, reproducir y visualizar sonidos
perro <- readWave('perro.wav')
perro
gato <- readMP3('gato.mp3')
gato
```

2. Dibujar la forma de onda de ambos sonidos.

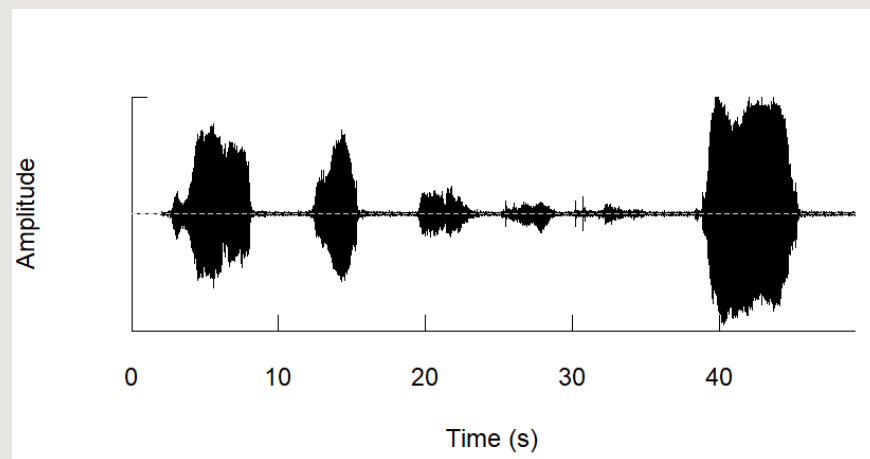
Con el uso del método plot podemos representarlo, son embargo se requiere saber la duración máxima del archivo. Para ahorrarnos la consulta existe otro método llamado oscillo(archivo, f=f) que nos da las ondas y no requiere de parámetros de duración. Para ver los ambos se usa:

```
20 plot( extractWave(gato, from = 1, to = 393984) )
21
22 oscillo(gato, f=f)
23 oscillo(perro, f=f)
```

PLOT:



OSCILLO:



3. Obtener la información de las cabeceras de ambos sonidos
Usando la función str(archivo) nos da la información de este impresa en la terminal:

```
str(perro)
str(gato)
```

```

> str(gato)
Formal class 'Wave' [package "tuner"] with 6 slots
..@ left      : int [1:393984] 0 0 0 0 0 0 0 0 0 0 ...
..@ right     : int [1:393984] 0 0 0 0 0 0 0 0 0 0 ...
..@ stereo    : logi TRUE
..@ samp.rate: num 44100
..@ bit       : num 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

```

4. Unir ambos sonidos en uno nuevo.

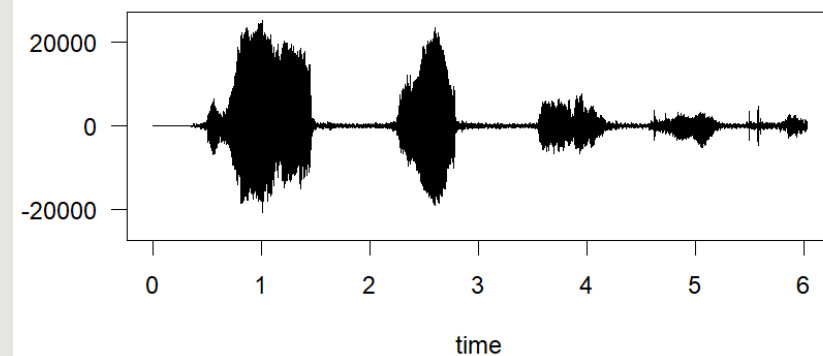
Para unir podemos recortar alguno de los audios o simplemente como hacemos aquí, pegar los dos audios juntos con `pastew(perro, gato, output="Wave")`

Dándonos la fusión de estas:

```

s3 <- pastew(perro, gato, output="Wave")
s3
plot( extractWave(s3, from = 1, to=265573) )
listen(s3)

```



5. Reproducir la señal obtenida y almacenarla como un nuevo fichero WAV, denominado "mezcla.wav".

Para crear un nuevo fichero de audio con el nuevo sonido mezclado, existe la función `write(combinado, file.path(nombre_archivo_nuevo.wav))`

```

writeWave(s3, file.path("mezcla.wav"))

```