

Periféricos y Dispositivos de Interfaz Humana



**UNIVERSIDAD
DE GRANADA**

CURSO 2024 - 2025

Práctica 5. Sonido

**GABRIEL VICO ARBOLEDAS
RAÚL RODRÍGUEZ RODRÍGUEZ**

Índice

1. Ejercicios obligatorios.....	3
1.1 Crear dos ficheros de sonido (WAV) para realizar los siguientes ejercicios. En el primero debe escucharse el nombre de la persona que realiza la práctica. En el segundo debe escucharse el apellido.....	3
1.2 Leer los dos ficheros de sonido creados y dibujar la forma de onda de ambos sonidos (por separado).....	4
1.3 Obtener la información de las cabeceras de ambos sonidos.....	5
1.4 Unir ambos sonidos en uno nuevo para escuchar el nombre y apellido correctamente.....	6
1.5 Dibujar la forma de onda de la señal y reproducir el sonido resultante (una vez unidos).....	6
1.6 Almacenar el sonido resultante en un archivo nuevo llamado “basico.wav”	7
2. Ejercicios opcionales.....	8
2.1 Pasarle un filtro de frecuencia para eliminar las frecuencias entre 10.000Hz y 20.000Hz. Almacenar la señal obtenida como un fichero WAV denominado “filtrado.wav”.	8
2.2 Tomar el sonido que se creó antes (lo tendremos en el archivo llamado “basico.wav”) para aplicarle el efecto de eco. Guardar ese sonido en un archivo nuevo llamado “eco.wav”. A continuación, se le debe dar la vuelta al sonido y almacenarlo como un fichero llamado “alreves.wav”	8

1. Ejercicios obligatorios

1.1 Crear dos ficheros de sonido (WAV) para realizar los siguientes ejercicios. En el primero debe escucharse el nombre de la persona que realiza la práctica. En el segundo debe escucharse el apellido.

Para crear los dos archivos, con el nombre y apellidos respectivamente, hemos accedido al servicio web [SpeechGen.io](https://speechgen.io), donde en el *prompt* de la IA escribimos en texto ambos datos y nos devuelve un archivo .wav que usaremos para la práctica.

Vista de la página del servicio web:

The screenshot shows the SpeechGen.io website interface. At the top, there's a navigation bar with the logo, 'SPEECHGEN.IO', and links for TTS, FAQ, Pricing, EN (language), and Log in (1489). The main heading is 'Realistic Text-to-Speech AI converter'. Below this, there are dropdown menus for 'English (US)', 'PRO Angel', 'pitch', and 'speed'. A toolbar contains icons for file operations and SSML. The main text area contains a prompt: 'Turn your text into speech using cutting-edge AI voices with an American English accent. Use it for work, videos, business, ads, social media, entertainment, and so much more. Just type or paste your text, generate the voice-over, and download the audio file.' Below the text area are dropdowns for 'mp3', 'Pause for paragraphs', 'Pause for sentences', and 'Sample Rate'. A large blue button labeled 'Generate speech' is prominent. On the right, a 'Balance' section shows '1489 Limits' and a 'Get more limits' button. Below this, a table shows character limits: '2979 characters' and '1489 characters'. At the bottom, there's a section titled 'AI voice examples' with five voice samples: 'PRO Avery', 'PRO Jenny EN', 'PRO Jane Smith', 'PRO Angel', and 'Matthew'.

English (US) PRO Angel pitch speed

Turn your text into speech using cutting-edge AI voices with an American English accent. Use it for work, videos, business, ads, social media, entertainment, and so much more. Just type or paste your text, generate the voice-over, and download the audio file.

Characters 259

mp3 Pause for paragraphs Pause for sentences Sample Rate

Generate speech

Balance
1489 Limits ?
Get more limits

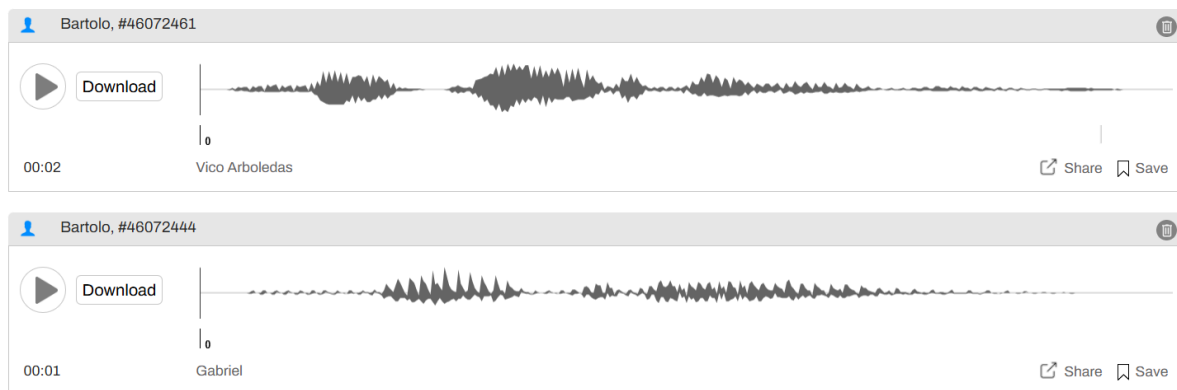
	2979 characters ?
PRO	1489 characters ?

Create realistic Voiceovers online! Insert any text to generate speech and download audio mp3 or wav for any purpose. Speak a text with AI-powered voices. You can convert text to voice for free for reference only. For all features, [purchase the paid plans](#)


AI voice examples


PRO Avery PRO Jenny EN PRO Jane Smith PRO Angel Matthew

Resultados del TTS del servicio web:



Nombres de ambos archivos:

 [apellidos.wav](#)

 [nombre.wav](#)

1.2 Leer los dos ficheros de sonido creados y dibujar la forma de onda de ambos sonidos (por separado).

La **lectura de ficheros** se hace mediante el siguiente código:

```
# Leer dos ficheros de sonido
nombre <- readWave('nombre.wav')
nombre
listen(nombre)

apellidos <- readWave('apellidos.wav')
apellidos
listen(apellidos)
```

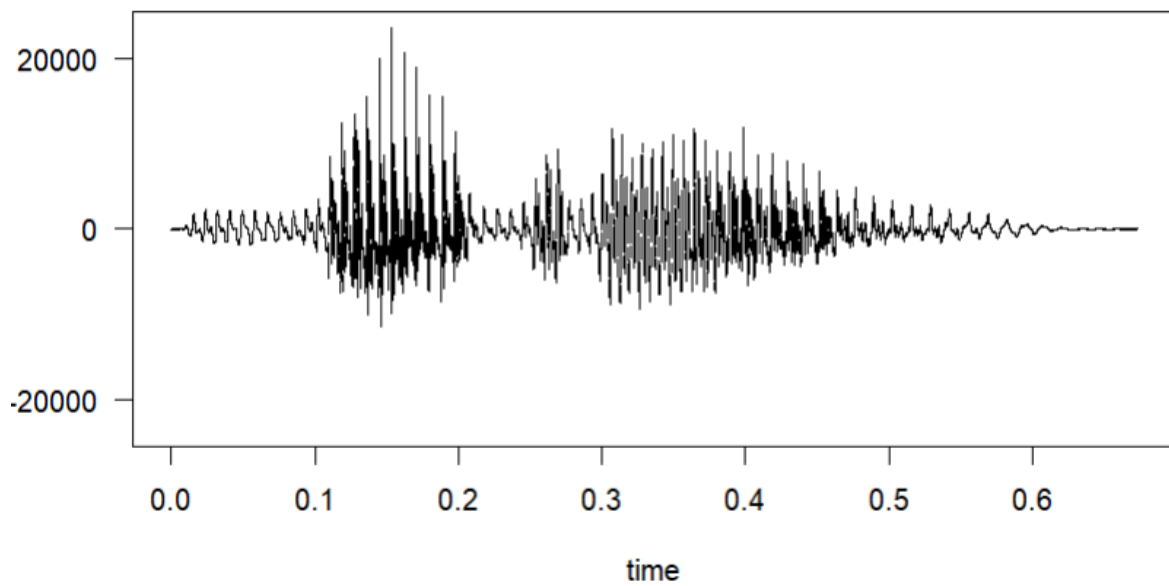
Y para **dibujar** la **forma** de **onda** de ambos ficheros, se hace con lo siguiente:

```
# Dibujar la forma de onda

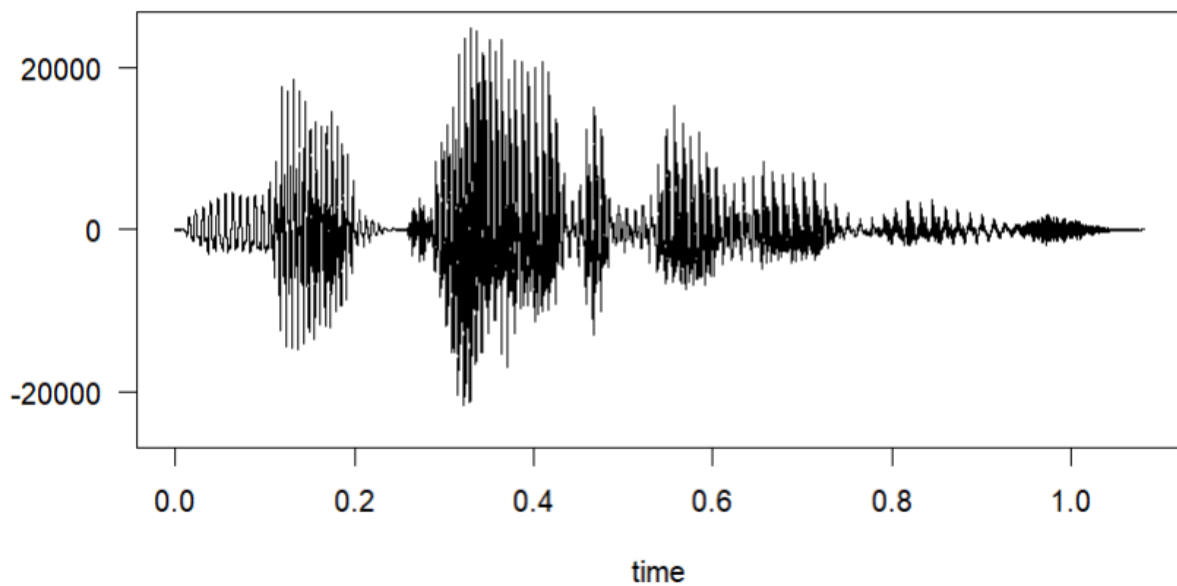
plot(extractWave(nombre, from = 1, to = length(nombre)), main="Nombre")
plot(extractWave(apellidos, from = 1, to = length(apellidos)),
main="apellidos")
```

Donde primero pintamos la onda de nombre y luego de apellidos:

Nombre



apellidos



1.3 Obtener la información de las cabeceras de ambos sonidos.

Para obtener la **información** de ambos ficheros, se debe ejecutar lo siguiente:

```
# Leer dos ficheros de sonido  
nombre <- readWave('nombre.wav')
```

```
nombre
```

```
apellidos <- readWave('apellidos.wav')  
apellidos
```

Donde realmente la orden que da la información es poner el **nombre** de la **variable** en sí, y no es necesario leer de nuevo los ficheros, pero da una mejor visualización del orden de ejecución. Tras esto, se puede ver por la terminal los datos cuando se ejecuta la orden dicha:

```
> nombre
```

```
Wave Object  
  Number of Samples:      32303  
  Duration (seconds):     0.67  
  Samplingrate (Hertz):   48000  
  Channels (Mono/Stereo): Mono  
  PCM (integer format):   TRUE  
  Bit (8/16/24/32/64):    16
```

```
> apellidos
```

```
Wave Object  
  Number of Samples:      51887  
  Duration (seconds):     1.08  
  Samplingrate (Hertz):   48000  
  Channels (Mono/Stereo): Mono  
  PCM (integer format):   TRUE  
  Bit (8/16/24/32/64):    16
```

1.4 Unir ambos sonidos en uno nuevo para escuchar el nombre y apellido correctamente.

Para **unir** los **sonidos** en una misma variable, se debe utilizar la función **pastew** (fichero2, fichero1, output); donde primero aparecerá el sonido de fichero1 y posteriormente el de fichero2:

```
# Unir ambos sonidos  
NombreApellidos <- pastew (apellidos, nombre, output="Wave")  
NombreApellidos
```

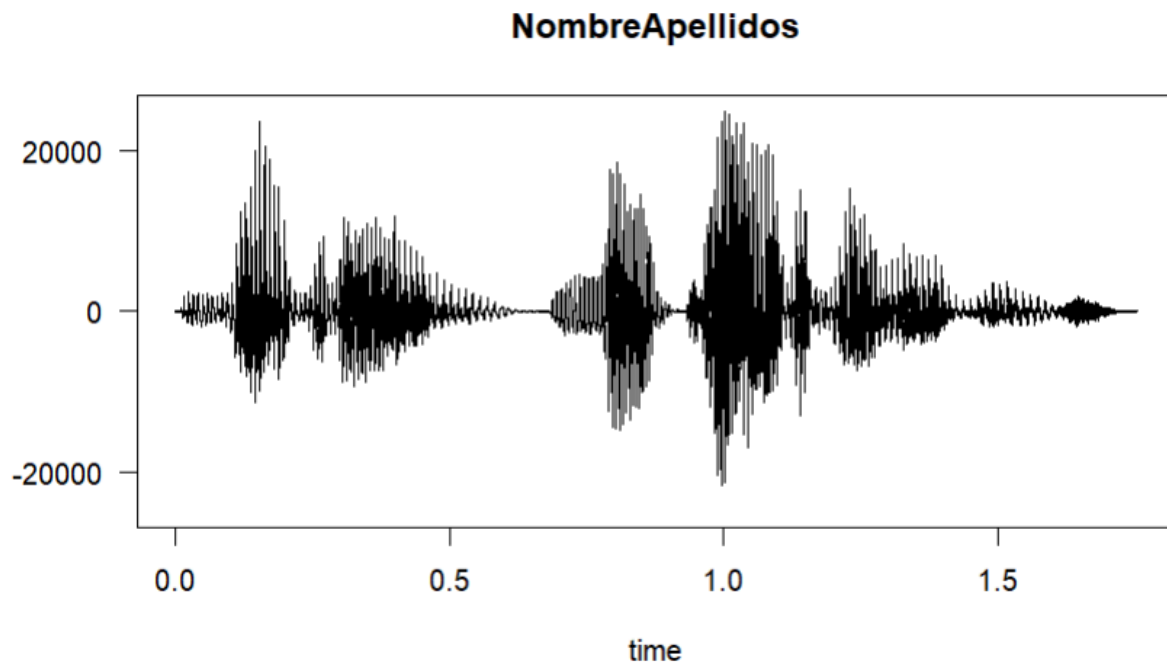
1.5 Dibujar la forma de onda de la señal y reproducir el sonido resultante (una vez unidos).

Para **escuchar** el **audio**, se debe ejecutar la siguiente orden:

```
listen(NombreApellidos)
```

Y para dibujar la forma de onda de la señal:

```
plot(extractWave(NombreApellidos, from = 1, to =  
length(NombreApellidos)), main="NombreApellidos")
```




También se podría poner el nº de muestras total que se puede visualizar en la información del fichero, en lugar de utilizar `length(nombre_variable)`.


1.6 Almacenar el sonido resultante en un archivo nuevo llamado “basico.wav”.

Para **almacenar** el nuevo **sonido** en un **fichero** se hace con la función **writeWave**(nombre_variable, file.path(“nombre_fichero.extension”):

```
# Guardar el audio  
writeWave (NombreApellidos, file.path("basico.wav"))
```

Y nos queda así:

 programa.R

 basico.wav

2. Ejercicios opcionales

2.1 Pasarle un filtro de frecuencia para eliminar las frecuencias entre 10.000Hz y 20.000Hz. Almacenar la señal obtenida como un fichero WAV denominado “filtrado.wav”

Para pasar un **filtro de frecuencia** se debe usar la función **bwfilter**:

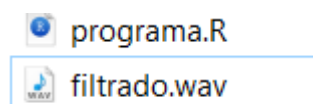
```
# Filtro de frecuencia para eliminar las frecuencias
filtro_NombreApellidos <- bwfilter(NombreApellidos, 48000, channel = 1,
n = 1, from = 10000, to = 20000, bandpass = TRUE, listen=FALSE, output =
"Wave")
listen(filtro_NombreApellidos)

filtro_NombreApellidos <- normalize(filtro_NombreApellidos, unit = "16")

listen(filtro_NombreApellidos)

writeWave (filtro_NombreApellidos, file.path("filtrado.wav"))
```

Se necesita **normalizar** el sonido para que cuando guarde el sonido en el nuevo fichero generado, lo haga al volumen que se espera, ya que sino R redondea los valores del sonido y el fichero reproduce el audio a un volumen muy bajo. Este es el resultado tras el guardado:



2.2 Tomar el sonido que se creó antes (lo tendremos en el archivo llamado “basico.wav”) para aplicarle el efecto de eco. Guardar ese sonido en un archivo nuevo llamado “eco.wav”. A continuación, se le debe dar la vuelta al sonido y almacenarlo como un fichero llamado “alreves.wav”

Para **añadir eco** al audio debe utilizarse la función **echo** tal y como se muestra a continuación:

```
# Añadimo eco:
NombreApellidos_Eco <- echo (NombreApellidos,
f=NombreApellidos@samp.rate, amp=c(0.8,0.4,0.2,0.1), delay=c(1,2,3,4),
```



```
output="Wave")

NombreApellidos_Eco <- normalize(NombreApellidos_Eco, unit = "16")
writeWave (NombreApellidos_Eco, file.path("eco.wav"))
```

Donde también es necesario normalizar el audio para un correcto guardado.

Para obtener el audio **al revés**, se debe utilizar la función **revw** :

```
# Al revés:
NombreApellidos_Eco_Alreves <- revw (NombreApellidos_Eco, output="Wave")

NombreApellidos_Eco_Alreves <- normalize(NombreApellidos_Eco_Alreves,
unit = "16")
writeWave (NombreApellidos_Eco_Alreves, file.path("alreves.wav"))
```

Finalmente, los dos archivos guardados con sus respectivas modificaciones:

