Trabajo final

Juan Cantero Jimenez

3/13/2022

Introducción

En este trabajo se nos ha proporcionado una muestra de audio contaminada con ruido blanco. En este trabajo se muestra una metodología para la reducción de ruido en el audio original.

Modelo: Singular Spectrum Analysis + Lineal model.

Haciendo uso de la aproximación propuesta en las instrucciones del trabajo, la cual utiliza una matriz con funciones periodicas de distinto periodo, no se ha podido obtener un modelo satisfactorio, bajo mi criterio. Es necesario destacar que haciendo uso de esta aproximación los mejores resultados fuero obtenidos haciendo uso de un modelo de regresión lineal multiple de tipo lasso, no se muestran los resultados.

De forma alternativa se propone hacer uso de una metodología conocida como Singular Spectrum Analysis, SSA de aqui en adelante, que permite descomponer una serie temporal en distintas componentes. Esta metodología será usada para la generación de las variables explicativas que se usarán para hacer regresión sobre el audio proporcionado. Las distintas componentes obtenidas por la metodología de SSA se recompondrán haciendo uso de un modelo de regresión lineal simple. Tras esto para ajustar aun más los resultados, se eliminarán las partes en las que por inspección auditiva solo deján escuchar ruido.

A continuación se muestra la metodología en detalle.

Primero se realiza el SSA sobre la muestra de audio original. El parámetro más relevante para la realización de un analisis de tipo SSA es L. Este parámetro debe de interpretarse como la ventana en la que se buscarán las distintas componentes de la serie temporal, debe de ser lo suficientemente grande como para ser capaz de capturar los patrones que se encuentren en la serie. Así se ha decidido escoger un L de 500 pues debería de ser más que suficiente para captar el patrón de una onda períodica de audio.

```
library(Rssa)
```

```
## Loading required package: svd
```

```
## Loading required package: forecast
```

```
## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo
```

```
##
## WARNING: Rssa was compiled without FFTW support.
```

```
## The speed of the routines will be slower as well.
```

```
##
## Attaching package: 'Rssa'
```

```
## The following object is masked from 'package:stats':
##
##     decompose
```

```
library(seewave)
library(tuneR)
data(sheep) ## Audio original, estaba poco escondido, mas si se tiene en
## cuenta que seewave ofrece muchas funcionalidades para el trabajo con muestras
## de audio
load("cabritus.Rdata")
s1 <- ssa(cabritus@left,  L = 500)
recon <- reconstruct(s1, groups = as.list(1:500))
```

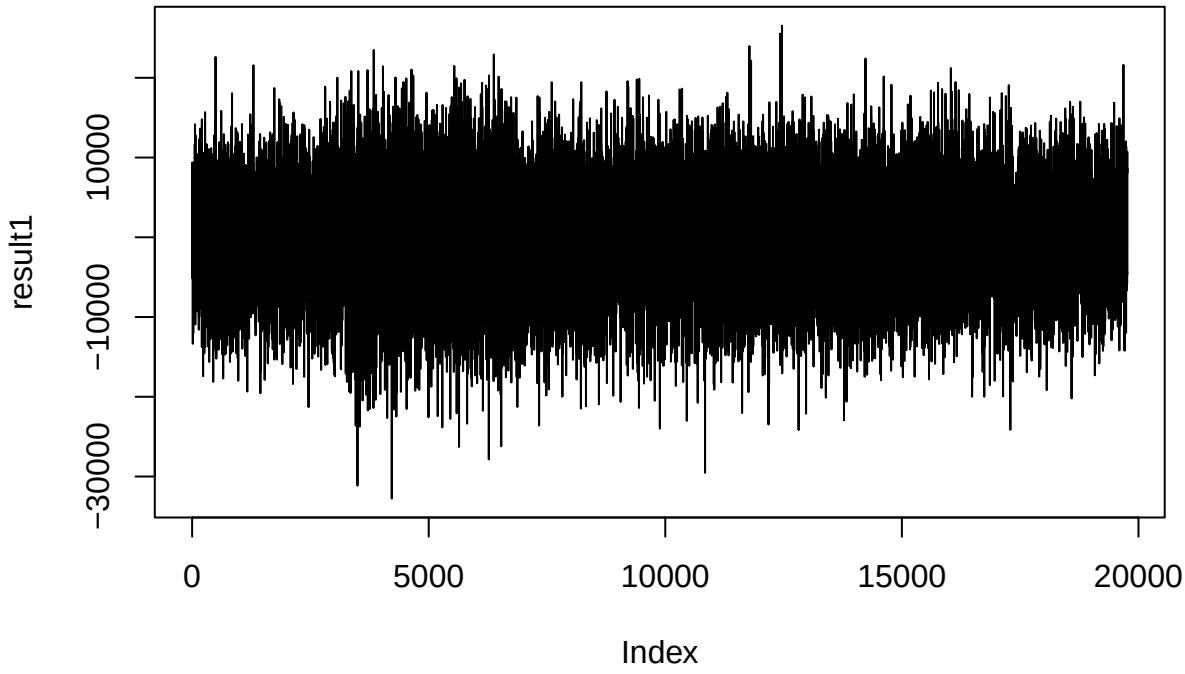
```
## Warning in trlan.svd(.get.or.create.trajmat(x), neig = neig, ..., lambda
## = .sigma(x), : TRLAN: info->ned (500) is large relative to the matrix dimension
## (500)
```

```
## Warning in trlan.svd(.get.or.create.trajmat(x), neig = neig, ..., lambda
## = .sigma(x), : ** It is more appropriate to use LAPACK dsyev/ssyev.
```

```
## Warning in trlan.svd(.get.or.create.trajmat(x), neig = neig, ..., lambda
## = .sigma(x), : TRLAN: ** reset maxlan to 500! **
```

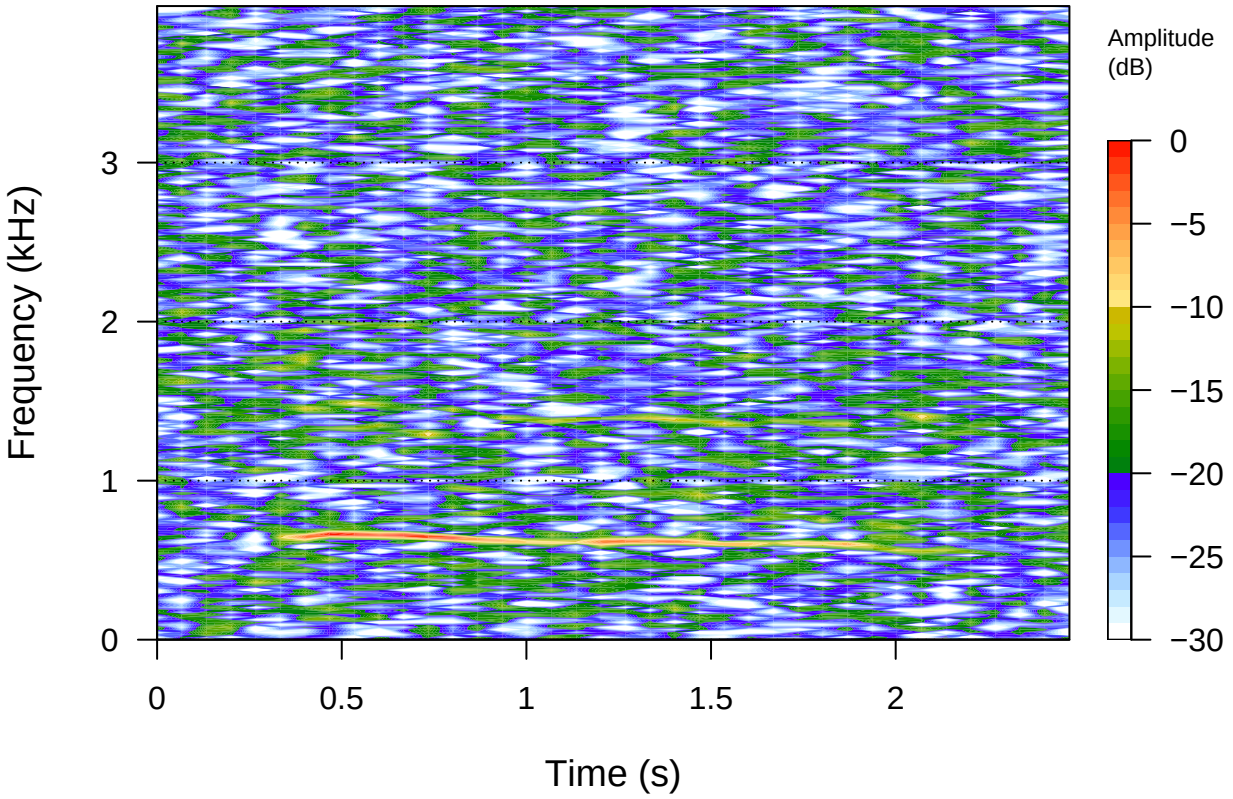
Tras esto se aplica un modelo de regresión lineal simple para la reconstrucción del audio original.

```
freq.amp_modified <- data.frame(lapply(recon, as.vector))
freq.amp_modified$y <- cabritus@left

model1 <- lm(y ~ ., data=freq.amp_modified)

result1 <- cabritus@left-model1$residuals
plot(result1, type = "l")
```




```
spectro(Wave(result1, samp.rate=8000))
```

```
## Warning in .local(left, ...): 'bit' not specified, assuming 16bit
```




```
sqrt(mean((result1-sheep@left)^2))
```

```
## [1] 6345.539
```

Como se puede observar, obtenemos un RMSE de 6345.539. Puesto que las distintas variables explicativas generadas por la función reconstruct() se encuentran ordenadas de mayor a menor relevancia para la reconstrucción de la serie original, se ira reduciendo el número de variables para ver si mejora el RMSE obtenido.

```
freq.amp_modified <- data.frame(lapply(recon[1:250], as.vector))
freq.amp_modified$y <- cabritus@left

model2 <- lm(y ~ ., data=freq.amp_modified)

result2 <- cabritus@left-model2$residuals

sqrt(mean((result2-sheep@left)^2))
```

```
## [1] 5543.593
```

```
freq.amp_modified <- data.frame(lapply(recon[1:125], as.vector))
freq.amp_modified$y <- cabritus@left

model3 <- lm(y ~ ., data=freq.amp_modified)

result3 <- cabritus@left-model3$residuals

sqrt(mean((result3-sheep@left)^2))
```

```
## [1] 4526.45
```

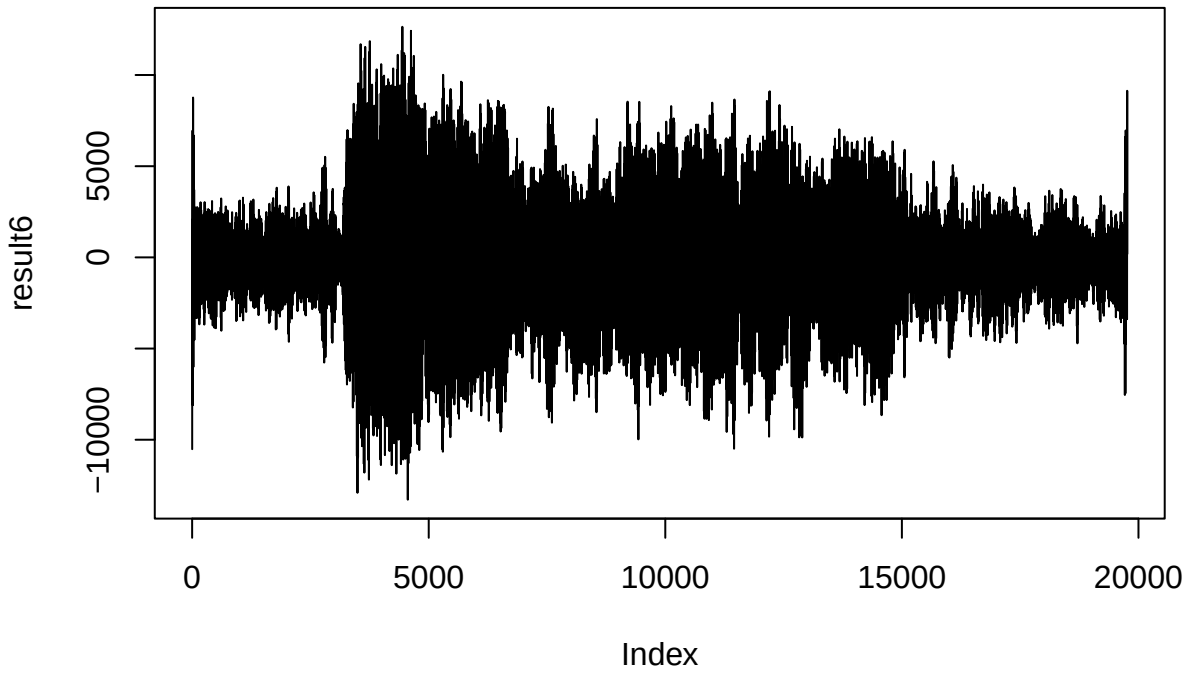
```
freq.amp_modified <- data.frame(lapply(recon[1:62], as.vector))
freq.amp_modified$y <- cabritus@left

model4 <- lm(y ~ ., data=freq.amp_modified)

result4 <- cabritus@left-model4$residuals

sqrt(mean((result4-sheep@left)^2))
```

```
## [1] 3763.311
```

```
freq.amp_modified <- data.frame(lapply(recon[1:32], as.vector))
freq.amp_modified$y <- cabritus@left

model5 <- lm(y ~ ., data=freq.amp_modified)

result5 <- cabritus@left-model5$residuals


sqrt(mean((result5-sheep@left)^2))
```

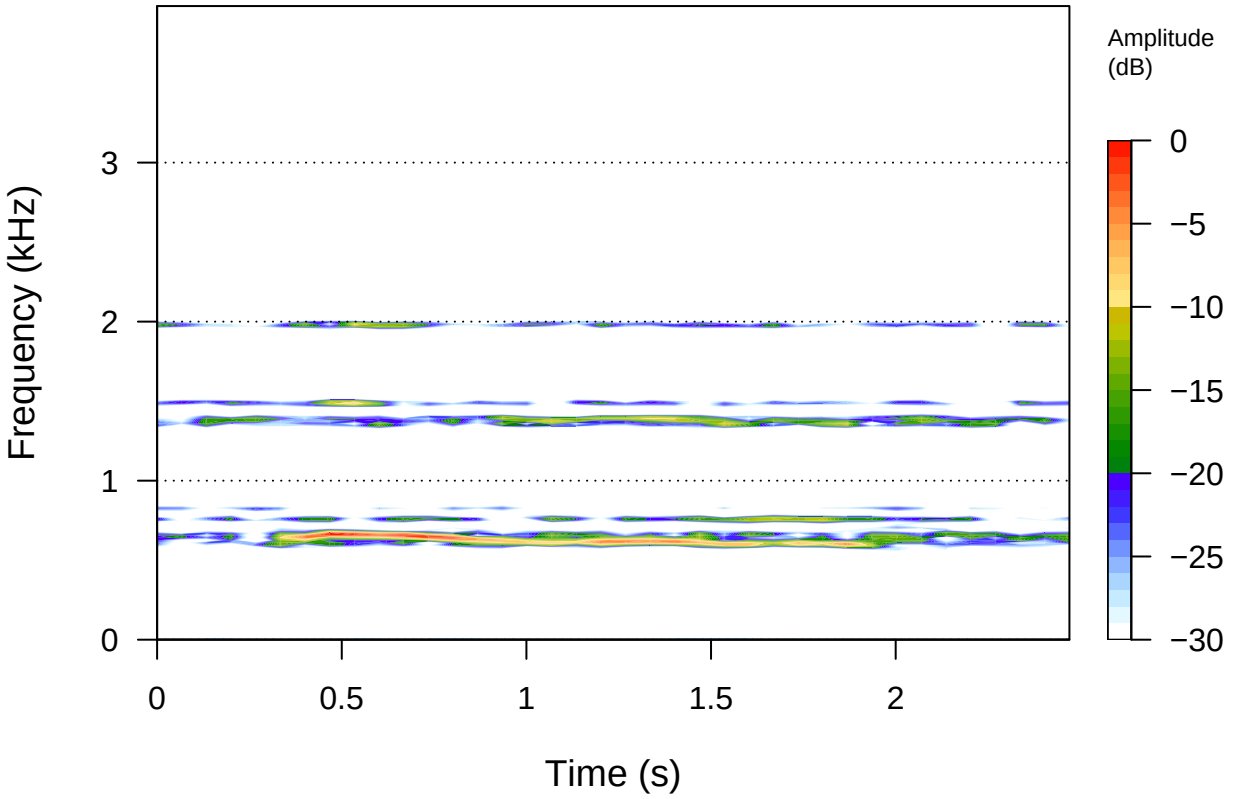
```
## [1] 3481.653
```

```
freq.amp_modified <- data.frame(lapply(recon[1:22], as.vector))
freq.amp_modified$y <- cabritus@left

model6 <- lm(y ~ ., data=freq.amp_modified)

result6 <- cabritus@left-model6$residuals
plot(result6, type = "l")
```




```
spectro(Wave(result6, samp.rate=8000))
```

```
## Warning in .local(left, ...): 'bit' not specified, assuming 16bit
```



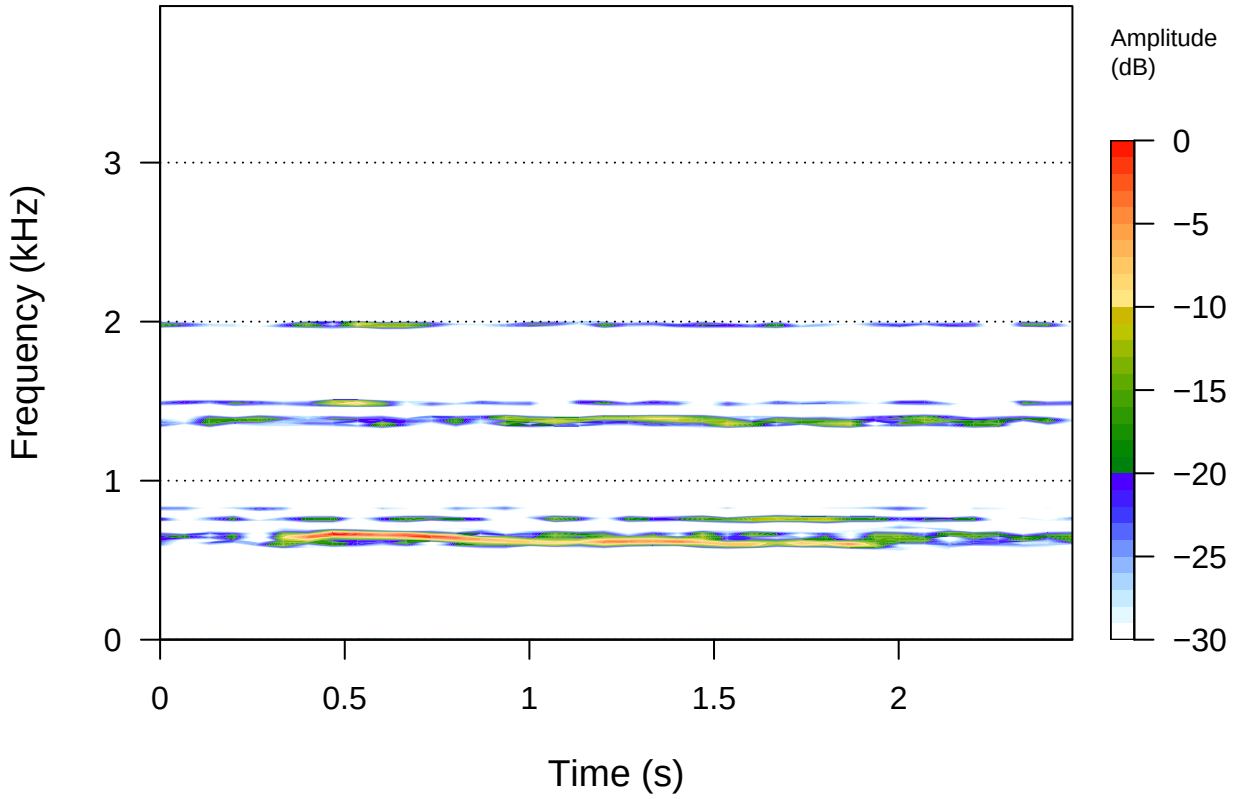

```
sqrt(mean((result6-sheep@left)^2))
```

```
## [1] 3449.701
```

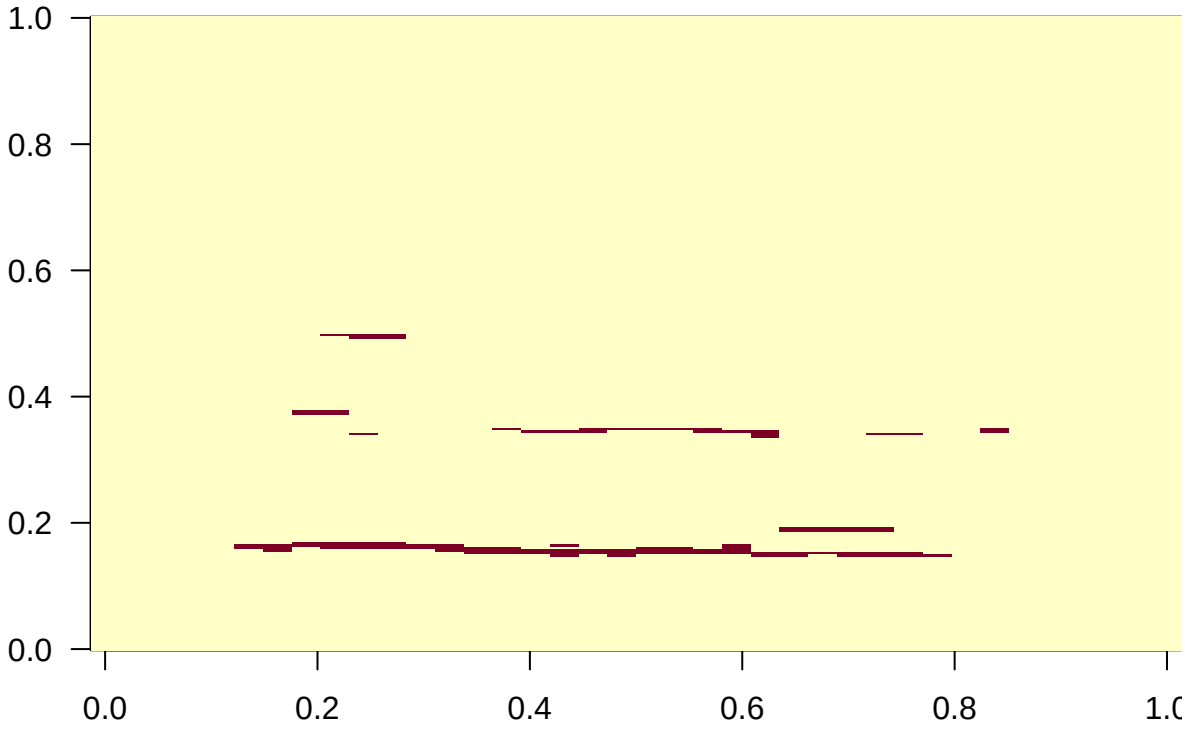
El mejor modelo obtenido es el model6 con el que se obtiene un RMSE de 3449.701. Se puede intentar mejorar este resultado eliminando la información de las regiones en las que solo se escucha ruido. Si obsevamos el espectrograma del resultado numero 6 podemos ver como la principal frecuencia, cercana a 1 kHz, no se extiende desde el principio hasta el final. Así se eliminará la información en la que esta frecuencia principal no esta presente. Para esto se realiza la transformada de Fourier, de esta se escogen los "pixeles" que sean superiores al percentil 99 de los datos. Tras esto se suma por columnas la transformada de Fourier, obteniendo las regiones de tiempo en las que existe o no existe sonido. La transformada de Fourier posee una limitación que radica en la reducción del tamaño de la muestra de audio. Así será necesario estirar esta representación manteniendo las proporciones. Esto se realizará con la fución strech. El resultado de la función strech se usará como "mascara" para eliminar las regiones que solo poseen ruido.

```
stf <- spectro(Wave(result6, samp.rate=8000))$amp
```

```
## Warning in .local(left, ...): 'bit' not specified, assuming 16bit
```

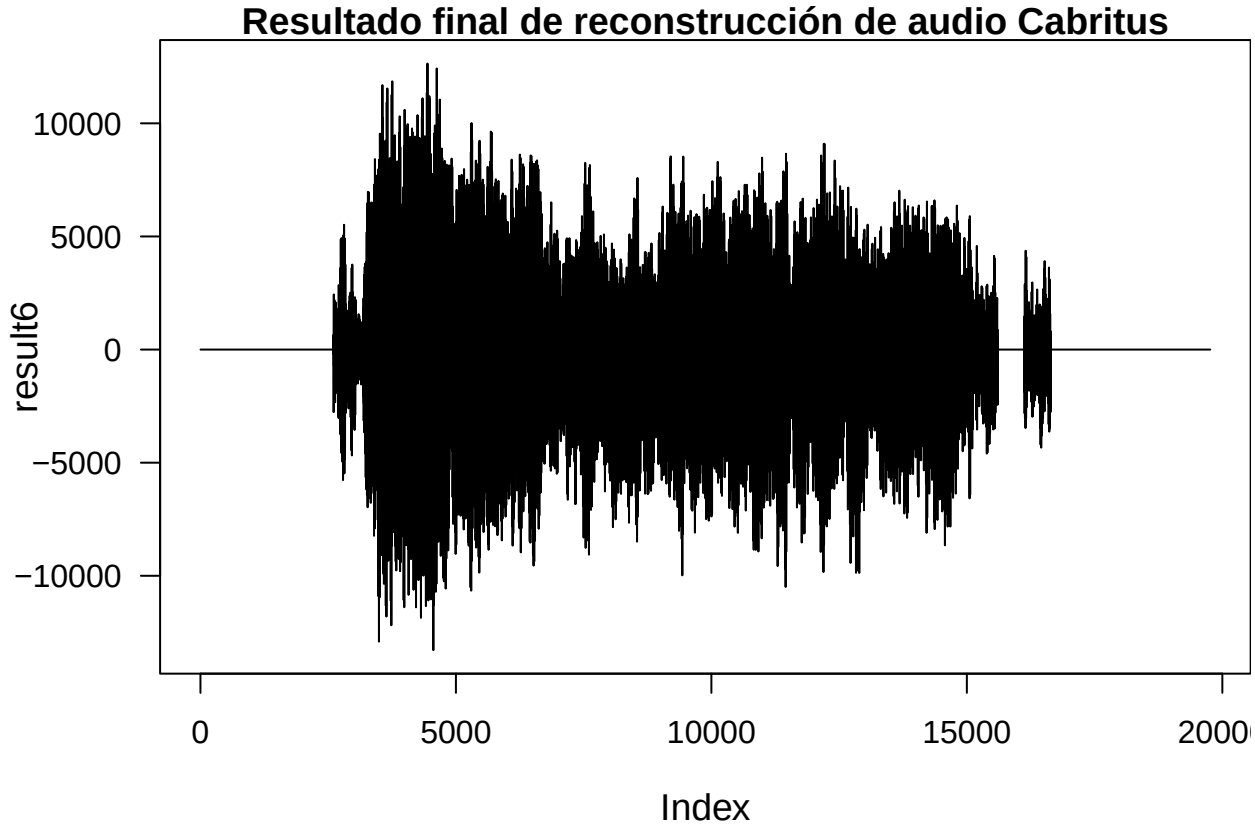



```
stf.discrete <-t(matrix(stf > quantile(as.vector(stf), (0.99)), nrow= nrow(stf), ncol=ncol(stf)))
par(mfrow=c(1,1))
image(stf.discrete)
```



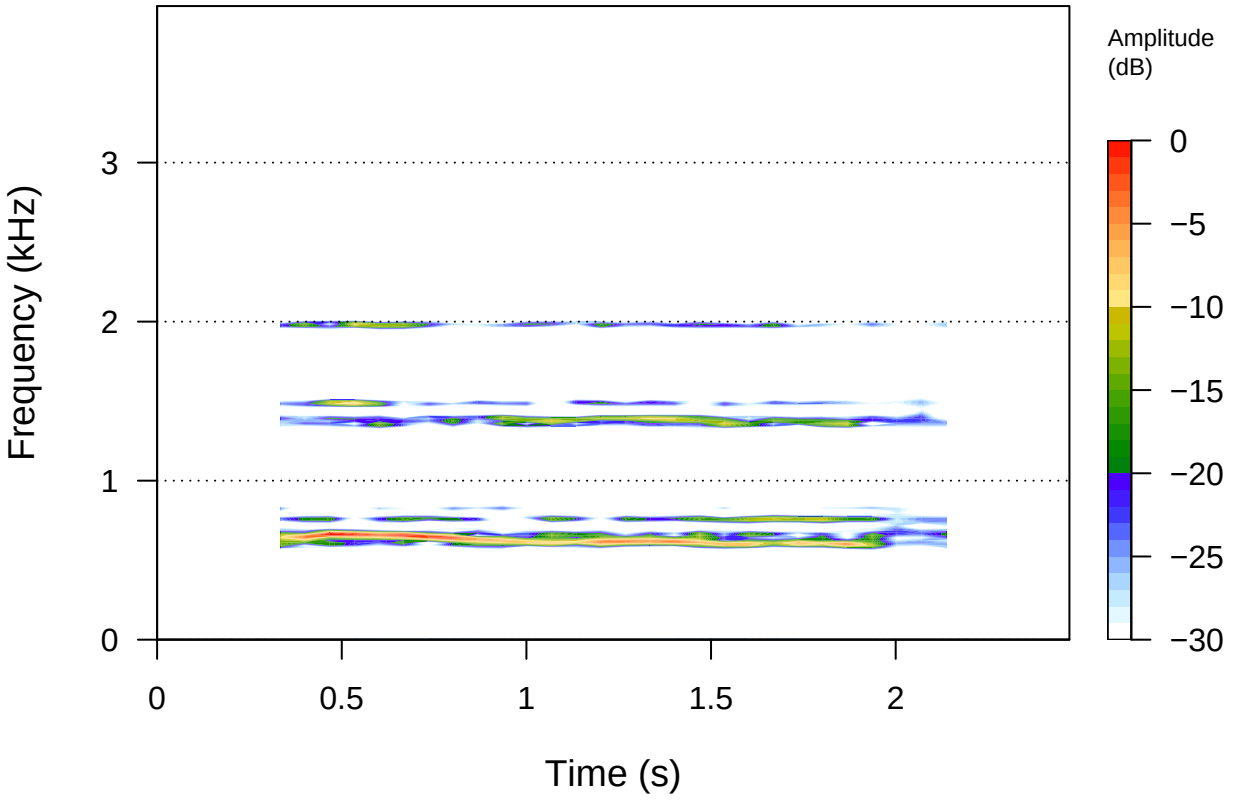

```
franctions <- apply(stf.discrete, 1, sum)
```

```
strech <- function(x, flength = 19764){
  result <- c()
  for (i in 1:flength){
    result <- c(result, x[round(qunif(i/flength, 0, length(x)))])
  }
  return(result)
}
contour<-strech(franctions)

result6[ contour == 0] <- 0
plot(result6, type = "l", main="Resultado final de reconstrucción de audio Cabritus")
```




```
spectro(Wave(result6, samp.rate=8000))
```

```
## Warning in .local(left, ...): 'bit' not specified, assuming 16bit
```




```
sqrt(mean((result6-sheep@left)^2))
```

```
## [1] 3376.546
```

```
save( result6, file ="result6.RData")
```

El resultado final de la metodología aplicada al problema propuesto ofrece un RMSE de 3375.658 con el audio original.