echogram (v0.1.0): Análisis de Ecogramas en R

Héctor Villalobos[1]

[1] *Instituto Politécnico Nacional - Centro Interdisciplinario de Ciencias Marinas. (IPN-CICIMAR). Depto. de Pesquerías y Biología Marina.*
email: hvillalo@ipn.mx

8 de febrero de 2017

Resumen

Un ecograma digital es la representación visual de una matriz numérica generada por ecosondas científicas a partir de la emisión de pulsos de sonido de alta frecuencia y la recepción de los ecos producidos por organismos acuáticos y el fondo presentes en la trayectoria de los pulsos. El escrutinio y el análisis de los ecogramas digitales multi-frecuencia son la base de las estimaciones de biomasa mediante métodos de la acústica pesquera. Existen varios formatos de datos para almacenar ecogramas, entre los que destaca el formato hac, definido y adoptado por científicos del Consejo Internacional para la Exploración del Mar (ICES). Aunque hay disponibles diferentes opciones de *software* dedicado para procesar y analizar ecogramas, algunos de ellos tienen un precio alto. Además, muchos equipos acústicos de todo el mundo, en algún momento de su cadena de análisis, recurren a R por su conveniencia, ya que proporciona numerosos procedimientos estadísticos, facilidades de mapeo y procesamiento de datos en general. El paquete `echogram` permite importar fácilmente datos acústicos de múltiples frecuencias almacenados en archivos hac y producir visualizaciones de ecogramas con paletas de colores predefinidas o personalizadas. También es posible combinar ecogramas consecutivos; enmascarar o borrar áreas no deseadas del ecograma; modelar y restar ruido de fondo; y más importante, desarrollar, probar e interpretar diferentes combinaciones de frecuencias para realizar el filtrado acústico de los datos del ecograma. Al permitir el acceso directo a las matrices de datos subyacentes de un ecograma digital en el entorno de programación flexible y poderoso de R, `echogram` constituye una herramienta didáctica, de autoaprendizaje y de investigación muy útil.

Índice

1. Obtener datos de archivos *.hac

El formato HAC es el estándar adoptado por el *International Council for the Exploration of the Sea* para el intercambio de datos acústicos pesqueros (ICES, 2005).

En `echogram` se han desarrollado tres funciones que pueden interactuar directamente con este formato, a saber `position.hac()`, `bottom.hac()` y `read.echogram()`. Éstas son funciones "*wrapper*" de las funciones del paquete `readHAC`, desarrollado por Kasper Kristensen (*Technical University of Denmark*), que pretenden simplificar la importación en R de datos contenidos en archivos .hac.

Estas funciones se han probado en datos obtenidos con una ecosonda científica Simrad EK60 y archivados como datos "crudos" (.raw) mediante el *software* ER60, también de Simrad. La conversión de los archivos .raw en .hac se realizó también con ER60, exportando como *Sample Data* los valores de fuerza de retrodispersión por volumen (S_v), la posición geográfica y las detecciones de blancos individuales (*Single Target*) (Figura 1).

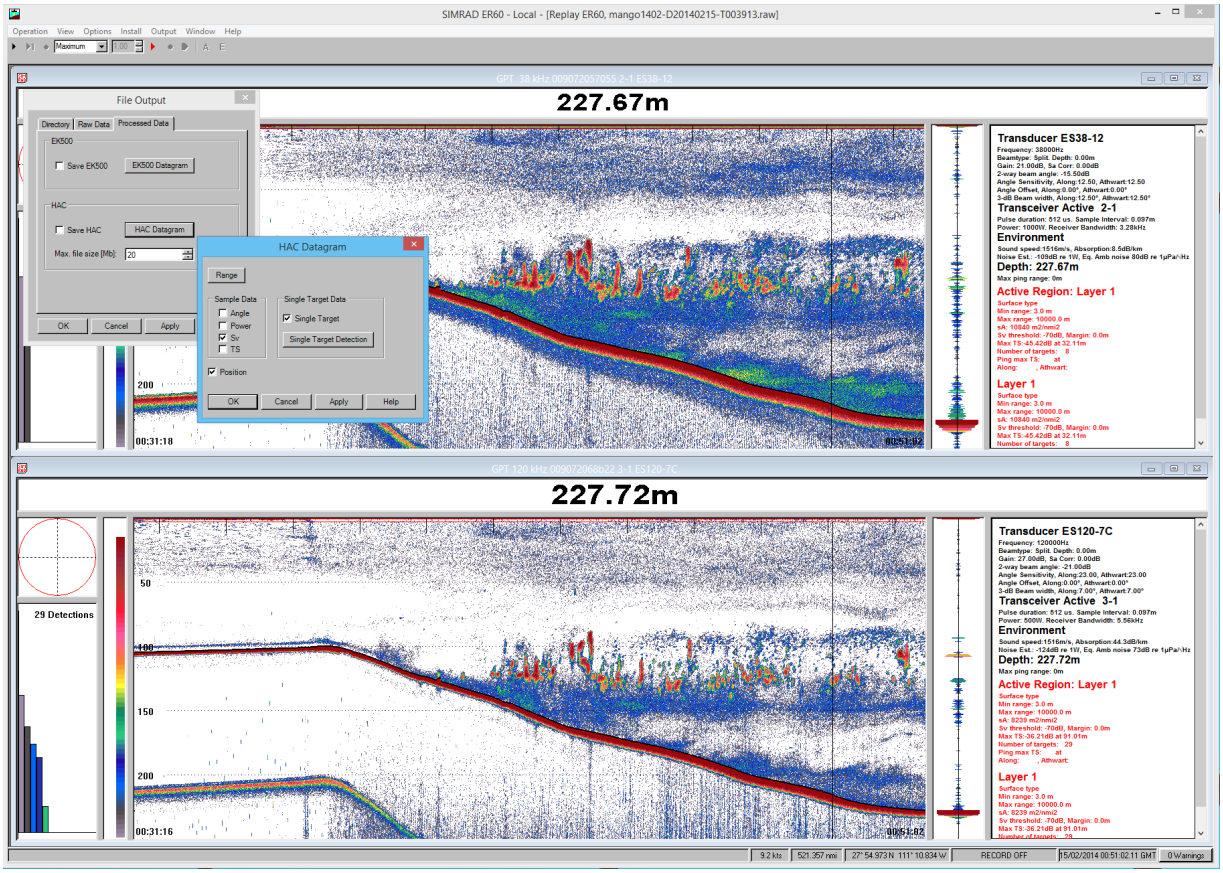

Figura 1: Captura de pantalla del software ER60 (Simrad, v2.4.3) mostrando un ecograma registrado en dos frecuencias (38 y 120 kHz). Se ilustran las opciones seleccionadas para generar los archivos .hac usados en `echogram`.

1.1. Posición geográfica (GPS)

La función `position.hac()` requiere como único argumento el nombre del archivo .hac. Esta función regresa un *data frame* con la fecha y hora del GPS (`time.gps`) y del CPU de la PC (`time.cpu`) al momento de la adquisión de los datos, así como la longitud (`lon`) y latitud (`lat`) correspondiente. Cada fila del *data frame* representa una lectura obtenida por el GPS del barco. Si la PC fue configurada con zona horaria "UTC" durante la adquisición de datos, entonces `time.gps` y `time.cpu` serán aproximadamente iguales, puesto que durante la importación a `time.cpu` se le agrega una fracción de segundo para lograr una precisión de 0.0001 s.

```
>  library(echogram) # cargar paquete echogram
>  pos <- position.hac("D20140215-T004330.hac") # importar posiciones
>  head(pos) # desplegar las primeras filas

             time.gps            time.cpu       lon      lat
1 2014-02-15 00:43:30 2014-02-15 00:43:30 -111.1590 27.91415
2 2014-02-15 00:43:32 2014-02-15 00:43:32 -111.1591 27.91416
3 2014-02-15 00:43:34 2014-02-15 00:43:34 -111.1592 27.91417
4 2014-02-15 00:43:36 2014-02-15 00:43:36 -111.1593 27.91418
5 2014-02-15 00:43:39 2014-02-15 00:43:39 -111.1594 27.91418
6 2014-02-15 00:43:40 2014-02-15 00:43:40 -111.1594 27.91420
```

A partir de la información anterior, la función `navigation.hac()` calcula el rumbo de navegación en grados (°, `bearing`), la distancia navegada en millas náuticas (nm, `navdist`), la diferencia de tiempo en horas (`time.dif`) y la velocidad de navegación en nudos (kn, `navspeed`) entre cada par de lecturas del GPS. El cálculo del rumbo y la distancia navegada se realiza mediante funciones del paquete `geosphere`. Más que la utilización directa de `navigation.hac()` por parte del usuario, ésta es invocada dentro de la función `read.echogram()` que se describe más adelante.

```
>  pos <- navigation.hac(pos)
>  head(pos) # desplegar las primeras filas (salida editada)
```

```
             time.cpu     lon    lat bearing   navdist   time.dif navspeed
1 2014-02-15 00:43:30 -111.16 27.914  276.52 0.0000000 0.00000000   0.0000
2 2014-02-15 00:43:32 -111.16 27.914  274.05 0.0052961 0.00055556   9.5329
3 2014-02-15 00:43:34 -111.16 27.914  275.99 0.0051153 0.00055556   9.2076
4 2014-02-15 00:43:36 -111.16 27.914  275.93 0.0051839 0.00053944   9.6097
5 2014-02-15 00:43:39 -111.16 27.914  277.97 0.0052368 0.00057611   9.0899
6 2014-02-15 00:43:40 -111.16 27.914  276.59 0.0052056 0.00055111   9.4457
```

Es posible, si así se desea, hacer un gráfico con esta información.

```
>  plot(pos$lon, pos$lat, type="l", xlab="longitud", ylab="latitud")
```

El código anterior es facilmente extensible para graficar la ruta de navegación contenida en un más de un archivo .hac o incluso para toda una campaña de prospección:

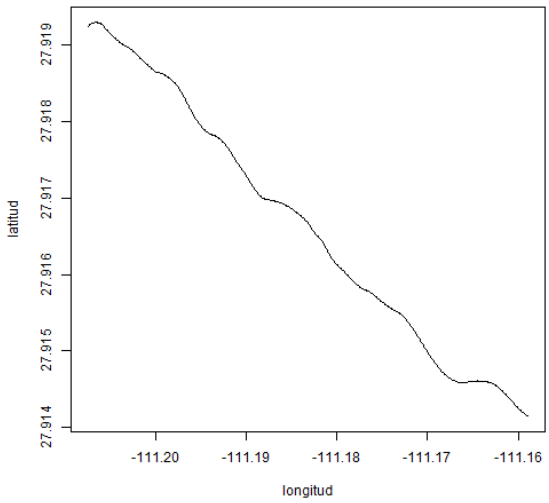

Figura 2: Posiciones GPS.

```
> # *** Código ilustrativo, no se incluyen resultados de su ejecución ***
> hacs <- list.files(pattern=glob2rx("*.hac")) # archivos a procesar
> nf <- length(hacs) # número de archivos
>
> # bucle para importar datos del GPS
> for ( k in 1:nf) {
+   x <- position.hac(hacs[k])
+   assign(paste("pos", k, sep="."), x)
+ }
>
> # integración en un solo data frame
> mpos <- do.call(rbind, mget(ls(pattern=glob2rx("pos.*")), .GlobalEnv))
> rm(list=ls(pattern=glob2rx("pos.*")))
>
> # calcular rumbo, distancia navegada y velocidad
> mpos <- mpos[order(mpos$time.gps), ] # antes, ordenar por tiempo
> mpos <- navigation.hac(mpos)
>
> # Gráficos
> # tratectoria
> plot(mpos$lon, mpos$lat, type="l", xlab="longitud", ylab="latitud")
> # principales direcciones de navegación
> library(plotrix); polar.plot(mpos$navspeed, mpos$bearing)
```

1.2. Profundidad del fondo detectado

La función `bottom.hac()` puede usarse para importar la profundidad en metros del fondo detectado (m, `detBottom`) en cada pulso o emisión de sonido de la ecosonda (*ping*), así como el

tiempo de la emisión (`pingTime`).

```
>  bot <- bottom.hac("D20140215-T004330.hac") # importar fondo detectado
>  head(bot)

             pingTime detBottom
1 2014-02-15 00:43:30  -172.266
2 2014-02-15 00:43:31  -172.411
3 2014-02-15 00:43:32  -172.502
4 2014-02-15 00:43:33  -172.638
5 2014-02-15 00:43:34  -172.704
6 2014-02-15 00:43:35  -172.802
```

Esta función además incorpora otros argumentos que permiten seleccionar la frecuencia acústica (`channel`) y opcionalmente, si se desea, realiza un gráfico del perfil del fondo, con la flexibilidad de elegir el límite máximo de profundidad (`maxDepth`) desplegado en la imagen.

```
>  bot <- bottom.hac("D20140215-T004330.hac", channel=1, plot=TRUE,
+                   maxDepth=-260)
```

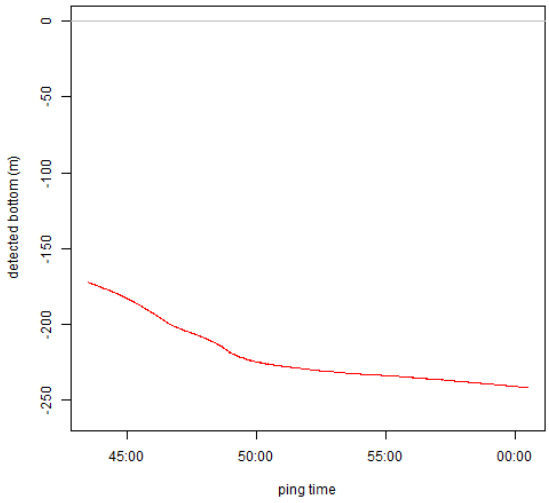

Figura 3: Perfil del fondo detectado (línea roja). La línea gris en la parte superior corresponde a la superficie.

Este gráfico puede servir para detectar rápidamente errores en el fondo detectado, mismos que podrían ser corregidos en el momento de convertir los archivos .raw en .hac mediante la verificación del correcto funcionamiento del detector de fondo en el ER60.

1.3. Importar ecogramas

La función `read.echogram()` hace uso de las tres funciones anteriores y regresa los datos necesarios para representar un ecograma dentro de R. Requiere especificar unicamente el nombre

del archivo .hac a importar y opcionalmente la frecuencia acústica (`channel`). Si ésta no se especifica, por defecto corresponde a la frecuencia más baja presente en el archivo .hac (`channel=1`). La función crea un objeto de clase "`echogram`" (una lista) con los siguientes componentes:

- `depth`, vector con la profundidad media de cada muestra en m y que corresponde a la dimensión vertical de la matriz `Sv`.
- `Sv`, matriz con los valores muestreados, en este caso la fuerza de retrodispersión por volumen (S_v, en decibeles, dB). Esta matriz tiene como atributo la frecuencia acústica de adquisición de los datos.
- `pings`, *data frame* con información de cada *ping* en el ecograma: tiempo de emisión (`pingTime`), profundidad del fondo detectado (m, `detBottom`), velocidad del barco (kn, `speed`), distancia navegada acumulada (nm, `cumdist`). El número de filas de este *data frame* representa la dimensión horizontal de la matriz `Sv`.

Sin necesidad de especificarlo, `read.echogram()` puede importar datos de archivos HAC donde la tupla de los pulsos sea de los tipos no comprimidos (U-16, 10030) generados por *software* como el ER60, o comprimidos y codificados (C-16, 10040) (ICES, 2005) por Movies+ y Movies3D (Berger et al., 2005). En el segundo caso sin embargo, el tiempo de procesamiento es considerablemente mayor debido a que no es posible el *parsing* vectorizado de las tuplas (Com. pers. Kasper Kristensen).

```
> eco.038 <- read.echogram("D20140215-T004330.hac", channel=1)
```

Una vez importado, se puede consultar la clase del objeto generado y su estructura:

```
> class(eco.038)

[1] "echogram"

> str(eco.038)

List of 3
 $ depth: num [1:2576] 0.0485 0.1455 0.2425 0.3395 0.4365 ...
 $ Sv   : num [1:2576, 1:1013] NA NA NA NA NA ...
  ..- attr(*, "frequency")= chr "38 kHz"
 $ pings:'data.frame': 1013 obs. of  4 variables:
  ..$ pingTime : POSIXct[1:1013], format: "2014-02-15 00:43:30" "2014-02-15 00:43:31" ...
  ..$ detBottom: num [1:1013] -172 -172 -173 -173 -173 ...
  ..$ speed    : num [1:1013] 0 0 9.53 9.53 9.21 ...
  ..$ cumdist  : num [1:1013] 0 0 0.00265 0.0053 0.00785 ...
 - attr(*, "class")= chr "echogram"
```

De este objeto se pueden extraer sus diferentes elementos, por ejemplo del vector `depth`, que define la profundidad de las muestras en un *ping*, se deduce que el largo de éstas es de 10 cm.

```
>  eco.038$depth[1:20] # desplegar primeros 20 valores
 [1] 0.0484992 0.1454976 0.2424960 0.3394944 0.4364928 0.5334912 0.6304896
 [8] 0.7274880 0.8244864 0.9214848 1.0184832 1.1154816 1.2124800 1.3094784
[15] 1.4064768 1.5034752 1.6004736 1.6974720 1.7944704 1.8914688
```

De la matriz `Sv`, que contiene los valores de S_v para cada muestra en el ecograma se puede consultar el rango de estos datos:

```
>  range(as.vector(eco.038$Sv), na.rm=TRUE)

[1] -136.55   -9.34
```

Por último, del *data frame* `pings` podemos visualizar las primeras filas:

```
>  head(eco.038$pings)

             pingTime detBottom    speed     cumdist
1 2014-02-15 00:43:30  -172.266 0.000000 0.000000000
2 2014-02-15 00:43:31  -172.411 0.000000 0.000000000
3 2014-02-15 00:43:32  -172.502 9.532939 0.002648039
4 2014-02-15 00:43:33  -172.638 9.532939 0.005296077
5 2014-02-15 00:43:34  -172.704 9.207577 0.007853737
6 2014-02-15 00:43:35  -172.802 9.207577 0.010452320
```

o mediante la función `summary()` ver algunos estadísticos de las variables de esta tabla:

```
>  summary(eco.038$pings, digits=3)

    pingTime                     detBottom        speed         cumdist
 Min.   :2014-02-15 00:43:30   Min.   :-242   Min.   :0.00   Min.   :0.00
 1st Qu.:2014-02-15 00:47:44   1st Qu.:-235   1st Qu.:9.10   1st Qu.:0.65
 Median :2014-02-15 00:51:59   Median :-230   Median :9.18   Median :1.30
 Mean   :2014-02-15 00:51:59   Mean   :-221   Mean   :9.17   Mean   :1.30
 3rd Qu.:2014-02-15 00:56:14   3rd Qu.:-207   3rd Qu.:9.29   3rd Qu.:1.95
 Max.   :2014-02-15 01:00:29   Max.   :-172   Max.   :9.61   Max.   :2.60
```

Cabe mencionar que dado que el primer *ping* en el ecograma representa la posición inicial para los cálculos de velocidad de navegación y distancia navegada acumulada, los valores de ambas variables en dicho *ping* son cero.

2. Visualizar ecogramas

Una vez importados los datos de un archivo .hac, el ecograma se puede visualizar con la función `echogram()`. Para ello basta con invocar la función con el nombre del objeto importado previamente como único argumento. No obstante, argumentos adicionales permiten definir el valor mínimo (umbral) de S_v a visualizar, la separación entre colores (en dB) y el esquema de colores deseado. Con el argumento `scheme` se puede elegir entre dos esquemas predefinidos de colores, denominados "echov" y "EK500", aunque también es posible especificar un vector de nombres válidos de colores para crear esquemas personalizados. La función `echogram()` también permite elegir la referencia en el eje x, entre distancia (por defecto), tiempo y número de *ping*. En la Figura 4 se presentan diferentes ecogramas ilustrando algunas de estas opciones. En todos los casos la escala de colores al lado derecho está expresada en dB.

```
>  echogram(eco.038)
>  echogram(eco.038, Svthr=-60, col.sep=1.5, scheme="EK500")
>  echogram(eco.038, xref="time", Svthr=-70)
>  echogram(eco.038, xref="ping", scheme=c("grey", "blue", "green", "black"))
```

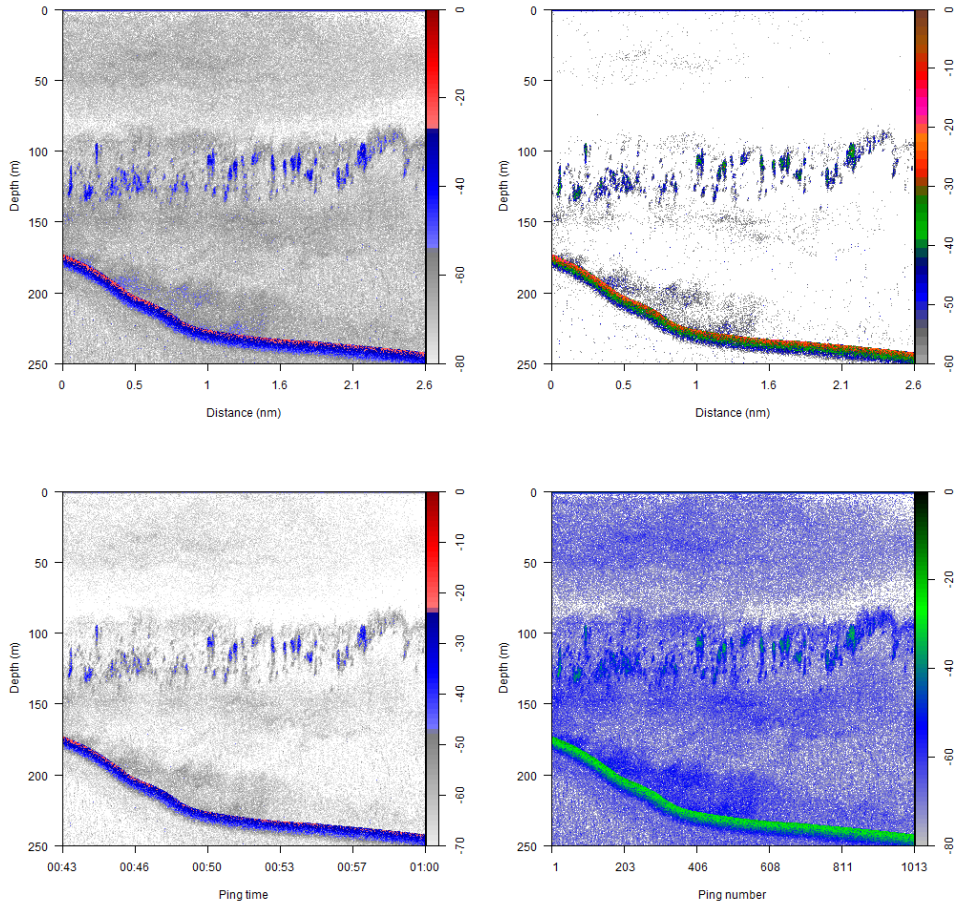

Figura 4: Ecogramas con diferentes opciones de visualización. Esquema de colores por defecto (arriba izquierda). Esquema de colores EK500 (arriba derecha). Referencia en eje x: tiempo (abajo izquierda). Referencia en eje x: número de ping y esquema de colores personalizado (abajo derecha).

Antes de aplicar un nuevo esquema de colores a un ecograma, la función `palette.echogram()`, que es invocada desde la función `echogram()`, permite diseñar y visualizar dicho esquema, tanto en términos de la combinación de colores elegida como de los valores mínimo y máximo de visualización y la separación entre colores (Figura 5).

```
> palette.echogram(Svthr=-60, Svmax=0, col.sep=3, scheme=c("grey", "blue",
+   "green", "black"), visu = TRUE)
```

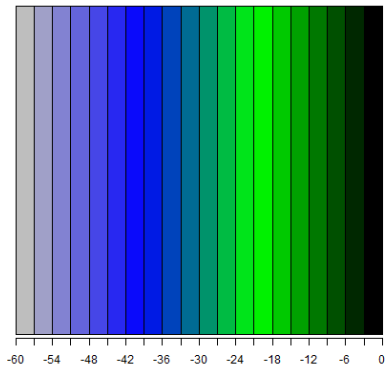

Figura 5: Visualización de una nueva paleta de colores personalizada.

```
$palette
 [1] "#BEBEBE" "#A0A0C8" "#8282D2" "#6464DC" "#4646E7" "#2828F1" "#0A0AFB"
 [8] "#001AE4" "#0043BB" "#006B93" "#00936B" "#00BB43" "#00E41A" "#00F100"
[15] "#00C900" "#00A100" "#007800" "#005000" "#002800" "#000000"

$breaks
 [1] -60 -57 -54 -51 -48 -45 -42 -39 -36 -33 -30 -27 -24 -21 -18 -15 -12  -9  -6
[20]  -3   0
```

3. Recortar y "blanquear" ecogramas

Dos funciones permiten eliminar áreas seleccionadas de un ecograma, ya sea recortando la matriz de datos (y vectores asociados) o haciendo igual a `NA` algunos valores de la misma.

La función `trim.echogram()` "recorta" un echograma vertical y longitudinalmente. Con el argumento `depth.max` se define el valor máximo de profundidad que se desea conservar, mientras que `ping.ini` y `ping.end` definen los *pings* inicial y final, respectivamente, que se desean conservar (Figura 6). En el futuro se prevee la posibilidad de seleccionar ambos *pings* mediante *clicks* en el ecograma a recortar.

```
>  eco.trim <- trim.echogram(eco.038, depth.max=200, ping.ini=100, ping.end=800)
>  echogram(eco.trim, Svthr=-60, col.sep=1.5, scheme="EK500")
```

Por otro lado, la función `mask.echogram()` crea una máscara para eliminar los valores de S_v de algunos metros desde la superficie (0 m) hacia abajo (`surf.off`) y de una capa referenciada al fondo (`bott.off`). En el segundo caso, el valor representa la distancia en metros por encima (si `bott.off` > 0) o por debajo (si `bott.off` < 0) del fondo detectado a partir de la cual se desea que se haga el blanqueo.

Si el argumento `mask=FALSE`, unicamente se genera una matriz de 1's y `NA`'s en la parte que se enmascara.

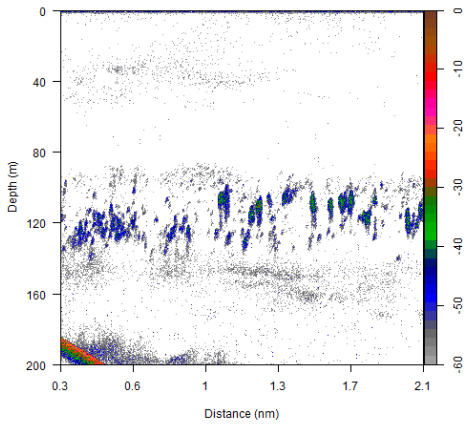

Figura 6: Ecograma recortado.

```
>  tmp <- eco.038 # hacemos una copia del ecograma a blanquear
>  masc <- mask.echogram(eco.038, surf.off=3, bott.off=0.5, mask=FALSE)
>  class(masc) # cuando mask=FALSE, el resultado es una matriz

[1] "matrix"

>  # las dimensiones de la máscara y de la matriz Sv son iguales
>  dim(masc); dim(tmp$Sv)

[1] 2576 1013
[1] 2576 1013

>  # la máscara consta de NA's y 1's
>  summary(as.vector(masc))

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
      1       1       1       1       1       1  342477
```

Para aplicar la máscara al ecograma se puede multiplicar la matriz anterior por la matriz de valores de S_v, o bien directamente usando el argumento `mask=TRUE` (Figura 7).

```
>  image(t(masc[nrow(masc):1, ]), col="lightgrey"); box() # imagen de la máscara
>  tmp$Sv <- tmp$Sv * masc # tmp puede  pasarse a la función echogram()
>  # o directamente se genera un ecograma con la máscara aplicada
>  eco.masc <- mask.echogram(eco.038, surf.off=3, bott.off=-2, mask=TRUE)
>  echogram(eco.masc, Svthr=-70, col.sep=1.5, scheme="EK500")
```

4. Determinar ruido de fondo en el mar

La función `noise.echogram()` incorpora el modelo de ruido de fondo (*background noise*) propuesto por De Robertis y Higginbottom (2007):

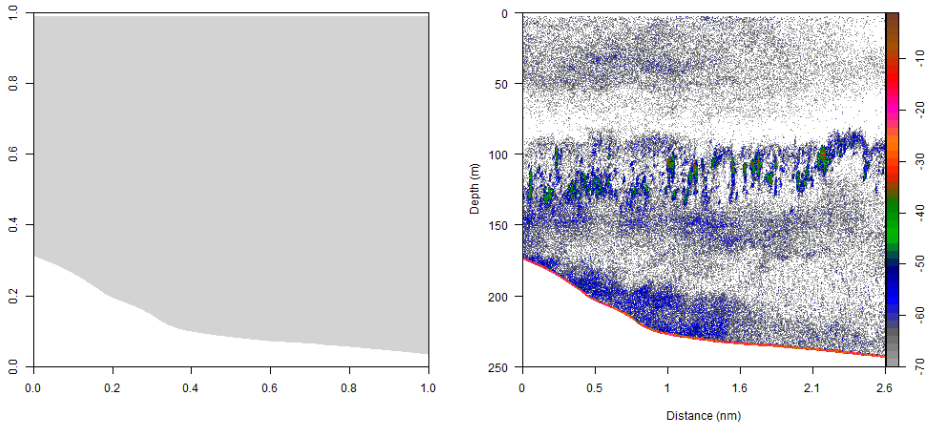

Figura 7: Máscara (izquierda) y ecograma con máscara aplicada (derecha).

$$S_{v,noise} = Noise_{1m} + 20\,log_{10}(r) + 2\alpha r$$

donde:

$Noise_{1m}$: nivel de ruido a 1 m del transductor
r : distancia al transductor
α : coeficiente de absorción

Para ilustrar el uso de esta función se utilizará un ecograma registrado hasta una profundidad de 500 m sin presencia evidente de organismos reflectores y donde el fondo está por debajo del rango de grabación. La función `noise.echogram()` puede invocarse usando como único argumento el nombre de un objeto de clase `echogram`, en cuyo caso es necesario hacer *click* en el ecograma para elegir el *ping* en donde se desea ajustar el modelo de ruido, sin embargo en el ejemplo siguiente se especifica directamente el *ping* 2. Aunque es posible representar el ruido y el modelo usando como mínimo los argumentos mencionados, la función produce un *warning* si no se especifica el coeficiente de absorción (α), que depende, entre otras cosas, de la frecuencia acústica. La función `swSoundAbsortion` del paquete `oce` calcula la absorción del sonido (db/m) en el mar para una frecuencia acústica dada en función de la salinidad, la temperatura y la presión, datos que tipicamente pueden provenir de un perfil de una sonda CTD.

En el ejemplo siguiente se utiliza un valor de α para 120 kHz calculado con datos de un lance CTD próximo al ecograma, así como dos valores de ruido a un metro del transductor ($Noise_{1m}$) para mostrar su efecto el ajuste del modelo a los datos de S_v (Figura 8).

```
>  eco <- read.echogram("D20151001-T140603_noise.hac", channel = 4)
>  echogram(eco, xref="ping")
>  noise1 <- noise.echogram(eco, ping=2, dB1m=-145, alpha=0.03550554, out=TRUE)
>  noise2 <- noise.echogram(eco, ping=2, dB1m=-131, alpha=0.03550554, out=TRUE)
>  echogram(noise2, xref="ping")
```

Una vez determinados los parámetros del modelo del ruido para un transductor en las condiciones ambientales dadas, se puede substraer dicho ruido de un ecograma, lo que se ilustra en la Figura 9 (la función `add.echogram()` se explica más adelante).

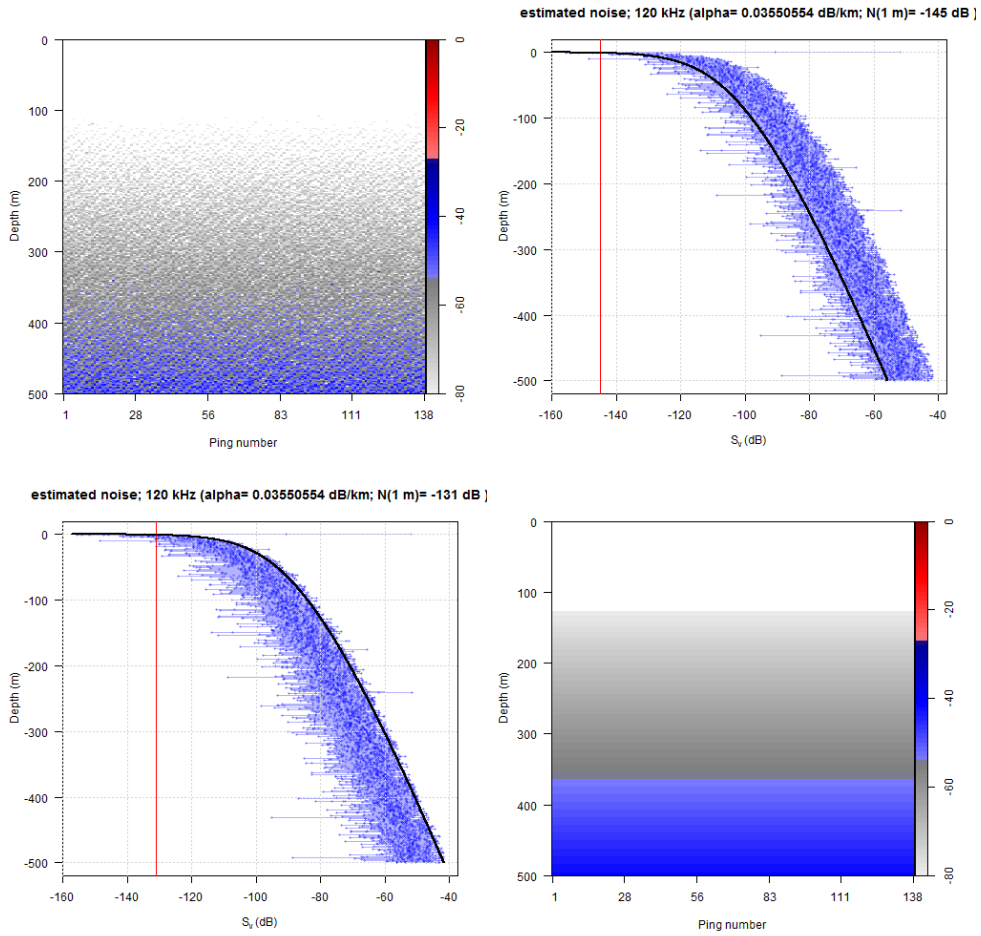

Figura 8: Ecograma a 120 kHz sin fondo detectado (arriba izquierda); modelo del ruido en el ping=2 con un valor incorrecto de $Noise_{1m}$ (arriba a la derecha); modelo ajustado correctamente (abajo, izquierda) y ecograma del ruido modelado resultante (abajo a la derecha). La línea roja vertical señala el valor del ruido a 1 m del transductor.

```
>  eco.120 <- read.echogram("D20140215-T004330.hac", channel=2)
>  noise.120 <- noise.echogram(eco.120, ping=2, dB1m=-131, alpha=0.03550554,
+                             out=TRUE, plot=FALSE)
>  eco.120nr <- add.echogram(eco.120, noise.120, operator="minus", domain="linear")

Warning in linear2dB(ans0): NaNs produced

>  echogram(eco.120)
>  echogram(eco.120nr)
```

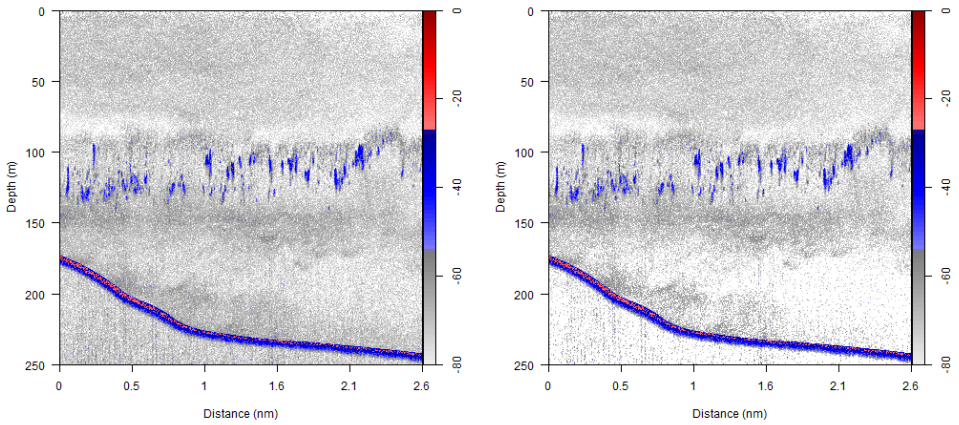

Figura 9: Ecograma a 120 kHz original (izquierda) y el resultante después de substraer el ruido modelado (derecha).

5. Combinar frecuencias

El tratamiento de ecogramas adquiridos con dos o más frecuencias representa una herramienta muy útil en el análisis de datos acústicos para el estudio de los organismos acuáticos. El principio básico de estos métodos consiste en destacar las diferencias de los grupos de organismos, como peces y zooplancton, con base en sus características reflectivas (Ballón et al., 2011).

Para ello se requiere realizar sumas o restas, en el dominio lineal o en dB, de las matrices de valores de S_v obtenidas con diferentes frecuencias acústicas.

Durante la adquisición, los datos bi- o multifrecuencia quedan contenidos dentro del mismo archivo .raw, y por ende en el mismo archivo .hac, pero es necesario importar cada una por separado usando el argumento `channel` de la función `read.echogram()`.

```
> # importar primera frecuencia
> eco.1 <- read.echogram("D20140215-T004330.hac", channel=1)
> attr(eco.1$Sv, 'frequency')

[1] "38 kHz"

> # importar segunda frecuencia
```

```
> eco.2 <- read.echogram("D20140215-T004330.hac", channel=2)
> attr(eco.2$Sv, 'frequency')

[1] "120 kHz"
```

5.1. Eliminar *pings* no coincidentes

Para poder sumar o restar dos matrices de S_v de ecogramas correspondientes, se require que ambas tengan las mismas dimensiones. De no ser así, la función `match.echogram()` verifica que el tiempo de cada *ping* en ambas matrices sea el mismo, conservando solamente los *pings* coincidentes y eliminando los que están solamente en una de las dos matrices.

```
>  # en la matriz de 38 kHz hay un ping extra
>  dim(eco.1$Sv); dim(eco.2$Sv)

[1] 2576 1013
[1] 2576 1012
```

La función `match.echogram()` verifica la coincidencia de los *pings* y elimina el que está de más en el ecograma de 38 kHz

```
>  match <- match.echogram(eco.1, eco.2)
>  eco.1 <- match$echogram1
>  eco.2 <- match$echogram2
>  # el número de pings y el tiempo de cada uno es el mismo en ambas frecuencias
>  dim(eco.1$Sv); dim(eco.2$Sv)

[1] 2576 1012
[1] 2576 1012
```

5.2. Sumar ecogramas

Antes de efectuar sumas o restas entre ecogramas correspondientes, además de asegurarse que los *pings* en ambas frecuencias coincidan, es conveniente aplicar máscaras para eliminar los ecos del fondo y por debajo de este y de los primeros metros en la superficie, donde tipicamente hay ruido.

```
>  eco.1m <- mask.echogram(eco.1, surf.off=3, bott.off=0.3, mask=TRUE)
>  eco.2m <- mask.echogram(eco.2, surf.off=3, bott.off=0.3, mask=TRUE)
```

Ahora se pueden hacer adiciones con la función `add.echogram()`, en la que además de especificar el ecograma 1 y el ecograma 2, se debe especificar la operación ("plus" o "minus") a realizar con el argumento `operator` y si dicha operación se realizará en dB o en el dominio lineal. En el primer caso la operación es aplicada directamente con los valores de S_v que están expresados en decibeles. Cuando se especifica `domain="linear"`, los valores de S_v son transformados previamente de acuerdo con $10^{(Sv/10)}$ y el resultado de la operación de adición o resta (X) es transformado de nuevo a decibeles según: $10 \times log_{10}(X)$ (ver ejemplo de la substracción del ruido).

```
> eco.sum <- add.echogram(eco.1m, eco.2m, operator="plus", domain="dB")
```

El ecograma resultante de esta adición se presenta en el panel de la izquierda de la Figura 10, donde se aprecia que ésta operación incrementó las diferencias entre los ecos débiles (tonos verdes y azules) y los fuertes (tonos rojos).

En ecogramas donde la cantidad de pixeles correspondientes a ecos débiles y a ecos fuertes no es muy dispar, un histograma puede ser muy útil para definir un valor umbral de la suma: $S_{v38} + S_{v120}$ que separe a ambos grupos. En el ejemplo esto no es muy claro, pero podríamos usar un valor de -118 dB (panel de la derecha en la Figura 10).

```
> Min <- floor(min(as.vector(eco.sum$Sv), na.rm=TRUE))
> echogram(eco.sum, Svthr=Min, col.sep=1.5, scheme="EK500")
> Sv <- as.vector(eco.sum$Sv)
> rango <- range(Sv, na.rm=TRUE)
> umbral <- -118
> hist(Sv, breaks=seq(floor(rango[1]), ceiling(rango[2])+1, 5))
> abline(v=umbral, col="red")
```

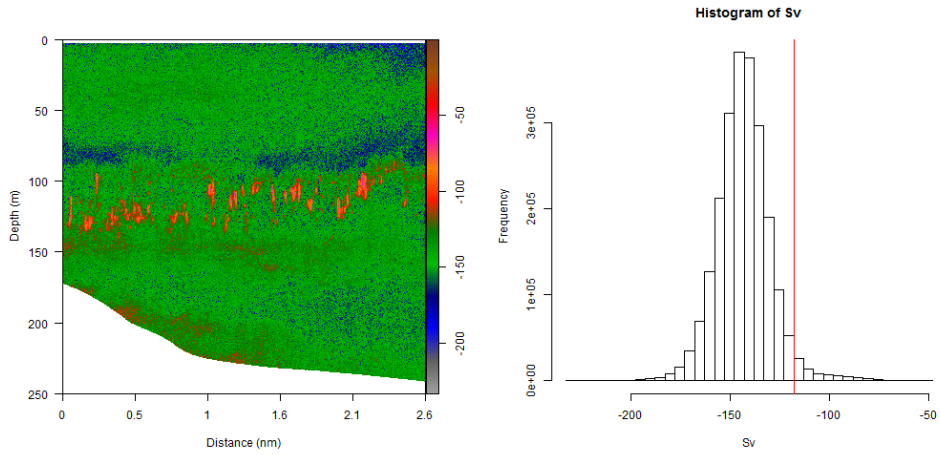

Figura 10: Suma de Sv.38 + Sv.120 (izquierda). Histograma de la suma de Sv(derecha).

Con este valor umbral se crean dos máscaras, una para ecos fuertes conservando solo los valores superiores al umbral ($[S_{v38} + S_{v120}] >$ -118) y otra para ecos débiles, inferiores al umbral ($[S_{v38} + S_{v120}] <$ -118).

```
> masc.ef <- eco.sum$Sv > umbral # ecos fuertes
> masc.ed <- eco.sum$Sv < umbral # ecos débiles
```

En principio, estas máscaras se pueden aplicar a ambas frecuencias para tener los ecogramas tanto de ecos fuertes como débiles a 38 kHz y a 120 kHz. Sin embargo, en el ejemplo aplicamos la máscara de ecos fuertes al ecograma de 38 kHz y la máscara de ecos débiles al ecograma de 120 kHz.

```
>  # ecos fuertes en 38 kHz
>  ef.038 <- eco.1m
>  ef.038$Sv <- ef.038$Sv * masc.ef
>  ef.038$Sv[ef.038$Sv == 0] <- NA
>
>  # ecos débiles en 120 kHz
>  ed.120 <- eco.2m
>  ed.120$Sv <- ed.120$Sv * masc.ed
>  ed.120$Sv[ed.120$Sv == 0] <- NA
```

En la Figura 11 se representan los ecogramas originales después de aplicar la máscara para quitar la señal del fondo y los resultantes de la operación de filtrado descrita antes.

```
>  # ecogramas originales
>  echogram(eco.1m, Svthr=-80, scheme="EK500", col.sep=2.5)
>  echogram(eco.2m, Svthr=-80, scheme="EK500", col.sep=2.5)
>
>  # ecogramas filtrados
>  echogram(ef.038, Svthr=-80, scheme="EK500", col.sep=2.5)
>  echogram(ed.120, Svthr=-80, scheme="EK500", col.sep=2.5)
```

5.3. Extraer valores puntuales de Sv

La función `sample.echogram()` permite la obtención de los valores de S_v de pixeles seleccionados. Esta selección se puede hacer de dos maneras. La más simple consiste en invocar la función, que por defecto despliega el ecograma a muestrear y espera a que el usuario haga *click* en los puntos de interés. Al terminar la selección de puntos se debe seleccionar *Stop* a la izquierda de la ventana gráfica o presionar la tecla escape del teclado.

Dado el comportamiento interactivo de esta función, en este ejemplo se representan algunos pixeles previamente seleccionados (Figura 12).

```
>  pts <- sample.echogram(eco.1)
```

El resultado de `sample.echogram()` es un *data frame* con las coordenadas de los puntos seleccionados (x y y) en unidades de la ventana gráfica, además del tiempo del *ping*, la profundidad y el valor de S_v en el pixel seleccionado.

```
>  pts

  id          x          y           d         pingTime     depth     Sv
1  1 0.06561054 0.48263878 0.000338162 15/02/2014 00:44 129.25037 -45.87
2  2 0.37485539 0.57550056 0.000039500 15/02/2014 00:49 106.06775 -71.27
3  3 0.43815112 0.55531322 0.000039300 15/02/2014 00:50 111.11167 -76.26
4  4 0.57921017 0.56742562 0.000416618 15/02/2014 00:53 108.10472 -34.51
5  5 0.76005511 0.60174411 0.000453269 15/02/2014 00:56  99.56886 -60.73
6  6 0.81611705 0.48465752 0.000093300 15/02/2014 00:57 128.76538 -74.49
7  7 0.86313673 0.03246105 0.000398429 15/02/2014 00:58 241.67151 -44.49
```

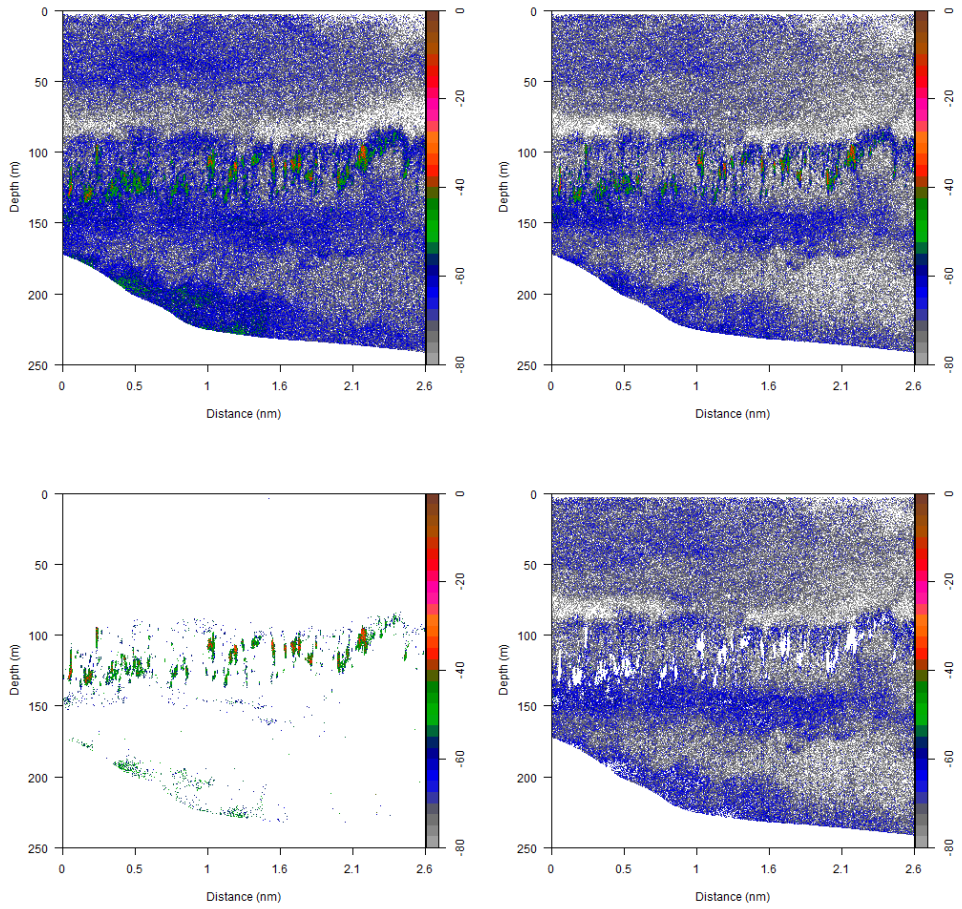

Figura 11: Ecogramas originales a 38 kHz (arriba a la izquierda) y a 120 kHz (arriba a la derecha) y los filtrados para ecos fuertes en 38 kHz (abajo a la izquierda) y para ecos débiles en 120 kHz (abajo a la derecha).

En la otra modalidad de uso de `sample.echogram()`, la función acepta las coordenadas de los puntos de interés, que se pudieron haber obtenido antes por el procedimiento descrito arriba, entonces, para la otra frecuencia:

```
>  echogram(eco.2)
>  pts2 <- sample.echogram(eco.2, plot=FALSE, coords=pts[ , 2:3])
```

```
>  pts2

  id          x          y            d            pingTime     depth     Sv
1  1 0.06561054 0.48263878 3.381621e-04 2014-02-15 00:44:36 129.25037 -45.08
2  2 0.37485539 0.57550056 3.945369e-05 2014-02-15 00:49:51 106.06775 -57.17
3  3 0.43815112 0.55531322 3.926897e-05 2014-02-15 00:50:56 111.11167 -77.78
4  4 0.57921017 0.56742562 4.166181e-04 2014-02-15 00:53:20 108.10472 -49.86
5  5 0.76005511 0.60174411 4.532688e-04 2014-02-15 00:56:23  99.56886 -57.05
```

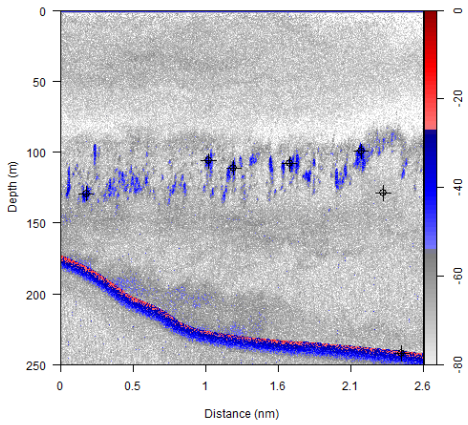

Figura 12: Ecograma a 38 kHz con pixeles muestreados.

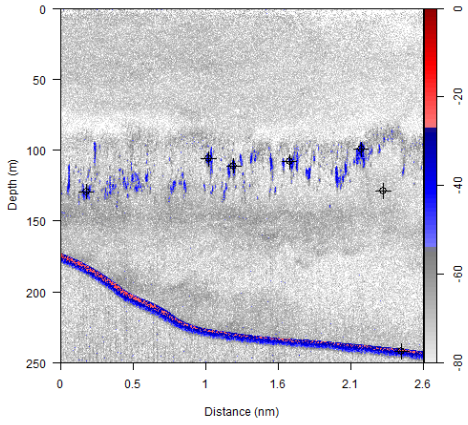

Figura 13: Ecograma a 120 kHz con pixeles muestreados.

```
6   6 0.81611705 0.48465752 9.334547e-05 2014-02-15 00:57:20 128.76538 -75.13
7   7 0.86313673 0.03246105 3.984293e-04 2014-02-15 00:58:09 241.67151 -40.27
```

6. Unir ecogramas

A través de la función `join.echogram()` es posible unir dos o más ecogramas de archivos .hac consecutivos en uno solo. Sin embargo, se debe tener en cuenta que el tamaño de los ecogramas importados en R es muy grande (en el ejemplo siguiente, cada uno de los ecogramas es de aproximadamente 20 MB), por lo que este procedimiento depende de la memoria disponible para R en la PC, pudiendo resultar poco práctico para procesar varios archivos de gran tamaño.

```
>   # importar dos archivos *.hac consecutivos
>   eco.120a <- read.echogram("D20140215-T002634.hac", channel = 2)
>   eco.120b <- read.echogram("D20140215-T004330.hac", channel = 2)
```

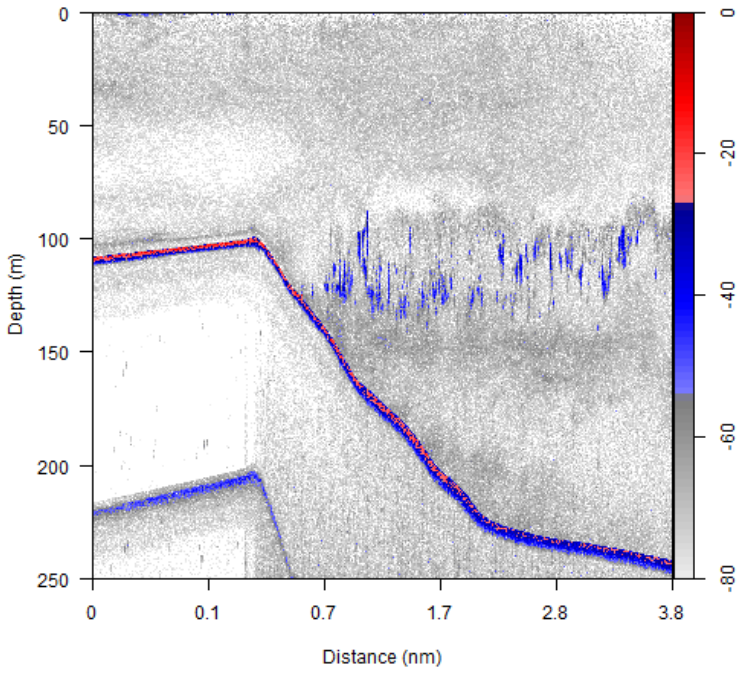

Figura 14: Ecograma a 120 kHz resultante de la unión de dos archivos *.hac.

```
>
>  # unirlos en un único ecograma
>  eco.120ab <- join.echogram(eco.120a, eco.120b)
```

El ecograma resultante se presenta en la Figura 14

```
>  echogram(eco.120ab)
```

Referencias

Ballón, M., A. Bertrand, A. Lebourges-Dhaussy, M. Gutiérrez, P. Ayón, D. Grados y F. Gerlotto. 2011. Is there enough zooplankton to feed forage fish populations off Peru? An acoustic (positive) answer. Progress in Oceanography **91**:360–381. 14

Berger, L., C. Durand, C. Marchalot y N. Diner, 2005. MOVIES+ User Manual version 4.3. Technical Report DNIS/ESI/DLE/DTI/00-051, IFREMER. 7

De Robertis, A. y I. Higginbottom. 2007. A post-processing technique to estimate the signal-to-noise ratio and remove echosounder background noise **64**:1282–1291. URL http://dx.doi.org/10.1093/icesjms/fsm112. 11

ICES, 2005. Description of the ICES *HAC* Standard Data Exchange Format, Version 1.60. Technical Report 278, ICES Cooperative Research Report. 3, 7