Cabritus

María Pallares y Martín Gramage

2025-02-25

```r
library(pracma)   # Funciones matemáticas avanzadas
library(signal)   # Procesamiento de señales y FFT
library(MASS)     # Regresión Ridge
library(glmnet)   # Regresión Lasso
library(entropy)  # Cálculo de entropía espectral
library(stats)    # Evaluación con AIC/BIC y estadísticas generales
library(tuneR)    # Escritura y manipulación de archivos WAV
library(ggplot2)  # Visualización de datos
library(lars)     # Algoritmo LARS para Lasso y selección de variables
```

Introducción

El descubrimiento del *cabritus hawaianus* supone un avance en paleontología al proporcionar evidencia acústica de una especie extinta a partir de lava solidificada. Sin embargo, la señal recuperada presenta un alto nivel de ruido, dificultando su análisis. Este estudio aplica modelos estadísticos y series de Fourier dentro de un filtrado no supervisado para eliminar ruido y reconstruir el alarido original. La evaluación del ajuste, mediante el error cuadrático medio (MSE) y la correlación con la señal original, mostró una reducción significativa del ruido, permitiendo una estimación más precisa de la bioacústica de esta especie extinta.

Materiales y métodos

Para la reconstrucción del alarido, se utilizó el archivo `cabritus.Rdata`, que contiene el objeto `cabritus` de clase `Wave`, del cual se extrajo el vector `cabritus@left` con 19764 valores a una frecuencia de 8000 Hz:

```r
load("cabritus.Rdata")
signal_noisy <- cabritus@left
N <- length(signal_noisy)
```

Siguiendo las indicaciones de la presentación del problema, se construyó una base de funciones trigonométricas con 10,000 términos (5,000 senos y 5,000 cosenos). A continuación, se seleccionaron las frecuencias con una correlación absoluta superior a 0.015 con la señal ruidosa, complementándose con funciones de baja correlación elegidas aleatoriamente.

Esta matriz de diseño se empleó para ajustar un modelo lineal, estimando los coeficientes mediante mínimos cuadrados ordinarios (OLS). Finalmente, el desempeño del ajuste se evaluó mediante el cálculo del MSE y la correlación con la señal original en una plataforma web con un límite de 25 validaciones.

La matriz de diseño generada fue almacenada en el archivo `freq_amp.RData`, asegurando su disponibilidad para futuros análisis y replicaciones del modelo.

Ajuste del modelo lineal

Para conseguir el objetivo del proyecto, se exploraron distintos enfoques estadísticos. Inicialmente, se consideraron métodos de reducción de dimensionalidad como la Regresión por Componentes Principales (PCR) y los Mínimos Cuadrados Parciales (PLS). Sin embargo, estos presentaban limitaciones.

Dado que estos métodos no resolvían eficazmente el problema de regularización, se optó por técnicas de regresión penalizada, como Ridge y Lasso, que permiten controlar la varianza de los coeficientes sin comprometer la interpretabilidad del modelo.

Los métodos Ridge y Lasso son formas de regresión penalizada que añaden un término de regularización para evitar el sobreajuste y mejorar la estabilidad del modelo cuando el número de predictores es grande.

La regresión **Ridge** minimiza la suma de los errores cuadráticos con una penalización sobre la norma $L2$ de los coeficientes:

$$\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 + \lambda \sum_{j=1}^{p} \beta_j^2$$

Donde λ es el parámetro de regularización, que controla cuánto se penalizan los coeficientes. Ridge es útil en este problema porque permite manejar la colinealidad entre las funciones de base de Fourier, evitando coeficientes excesivamente grandes y proporcionando una solución más estable.

Lasso introduce una penalización $L1$ en los coeficientes, modificando la función de pérdida de la siguiente manera:

$$\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 + \lambda \sum_{j=1}^{p} |\beta_j|$$

Este método no solo regula la magnitud de los coeficientes, sino que también fuerza algunos a cero, realizando una selección automática de variables. En este contexto, Lasso es útil para identificar las frecuencias más relevantes en la reconstrucción de la señal, eliminando aquellas que no aportan información significativa.

En ambos casos, el parámetro λ es crucial, ya que define el grado de penalización. Valores altos de λ reducen el sobreajuste, pero pueden hacer que el modelo pierda información valiosa. Por esta razón, se utiliza validación cruzada para seleccionar el λ óptimo, como se hizo con `cv.glmnet` en este estudio.

Proceso no supervisado

Dado que no se contaba con una referencia de la señal original, se implementó un enfoque de filtrado no supervisado, identificando patrones sin datos etiquetados. Por ello, se han considerado diversas ténicas estadísticas complementarias al proceso presentado que pudieran mejorar nuestros resultados y permitieran extraer características relevantes a partir de la señal ruidosa, asegurando la eliminación del ruido sin introducir sesgos adicionales en el proceso de reconstrucción.

Selección de la matriz de diseño

La selección de la matriz de diseño es un paso clave en la construcción del modelo, ya que influye directamente en su capacidad para capturar la información relevante de la señal. Inicialmente, se propuso una combinación de variables basadas en su correlación con la señal ruidosa junto con una selección aleatoria. Sin embargo, este enfoque puede beneficiarse de criterios adicionales de selección.

Transformada de Fourier

Primeramente, para optimizar este proceso se incorporó información espectral obtenida mediante la Transformada Rápida de Fourier (FFT), que permite identificar las frecuencias dominantes de la señal. Es una

herramienta matemática que permite descomponer una señal compleja en una suma de funciones sinusoidales simples, cada una con una frecuencia, amplitud y fase determinadas. En su forma continua, la Transformada de Fourier se define como:

$$X(f) = \int_{-\infty}^{\infty} x(t) e^{-2\pi i f t} \, dt$$

Donde $x(t)$ es la señal original en el dominio del tiempo, $X(f)$ representa su equivalente en el dominio de la frecuencia, f es la frecuencia y i es la unidad imaginaria.

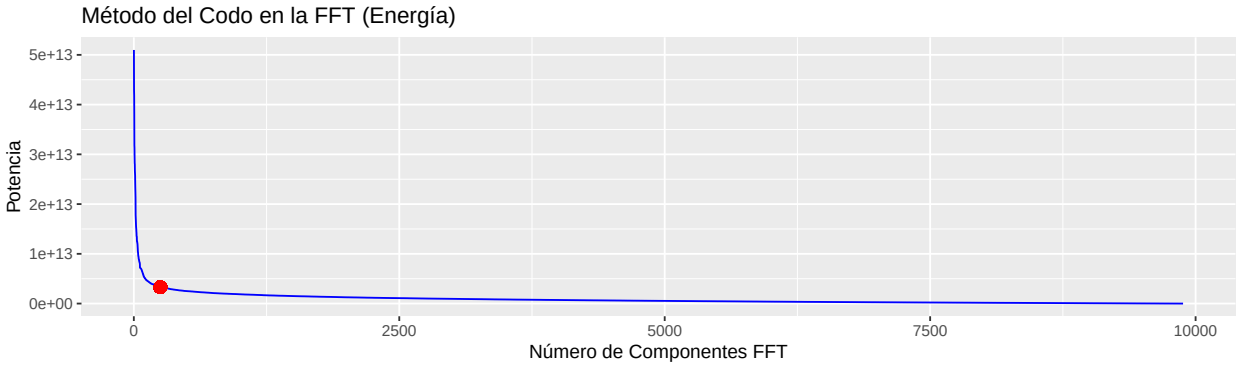
Para señales discretas, la Transformada Rápida de Fourier (FFT) permite calcular eficientemente la Transformada Discreta de Fourier (DFT). Las frecuencias con mayor potencia en el espectro suelen representar las características clave del sonido original, mientras que el ruido se distribuye de manera más uniforme. Identificando estas frecuencias dominantes, es posible filtrar el ruido y reconstruir la señal con mayor precisión, preservando su estructura fundamental.

```
compute_fft <- function(signal) {
  N <- length(signal)
  fft_values <- fft(signal)
  power_spectrum <- Mod(fft_values[1:(N/2)])^2
  freqs <- (0:(N/2 - 1)) / N
  return(data.frame(freq = freqs, power = power_spectrum))
}
```

A pesar de su utilidad, la FFT produce un conjunto amplio de frecuencias, y no todas ellas necesariamente informativas. Algunas pueden corresponder a ruido, afectando a la calidad del modelo. Para abordar este problema se seleccionaron las frecuencias más relevantes aplicando el método del codo sobre la energía explicada, permitiendo identificar el subconjunto de componentes con mayor contribución a la señal.

```
# Función para encontrar el codo (geométrico)
find_elbow <- function(x, y) {
  line_vec <- c(tail(x, 1) - x[1], tail(y, 1) - y[1])
  line_vec <- line_vec / sqrt(sum(line_vec^2))
  vecs <- cbind(x - x[1], y - y[1])
  proj_lens <- vecs %*% line_vec
  proj_points <- t(t(proj_lens %*% t(line_vec)))

  distances <- sqrt(rowSums((vecs - proj_points)^2))
  which.max(distances)
}
# --- Codo sobre la FFT ----
fft_data <- compute_fft(signal_noisy)
fft_ordered <- fft_data[order(-fft_data$power), ]
power_sorted <- fft_ordered$power
# Aplicar método del codo a la energía
x_vals <- 1:length(power_sorted)
elbow_index_fft <- find_elbow(x_vals, power_sorted)
selected_freqs <- fft_ordered[1:elbow_index_fft, ]
# Graficar el primer codo (FFT)
df_fft <- data.frame(Components = 1:length(power_sorted), Power = power_sorted)
ggplot(df_fft, aes(x = Components, y = Power)) +
  geom_line(color = "blue") +
  geom_point(aes(x = elbow_index_fft, y = power_sorted[elbow_index_fft]), color = "red", size = 3) +
  ggtitle("Método del Codo en la FFT (Energía)") +
  xlab("Número de Componentes FFT") +
  ylab("Potencia")
```



No obstante, este criterio por sí solo tampoco garantiza que todas las frecuencias seleccionadas sean útiles. Algunas pueden seguir conteniendo ruido o información redundante. Por ello, se aplicó un análisis adicional basado en la entropía espectral.

Entropía espectral

La entropía espectral mide cómo se distribuye la energía en el dominio de la frecuencia. Se basa en la entropía de Shannon y se define como:

$$H = -\sum_{i=1}^{N} p_i \log_2(p_i)$$

Donde p_i es la proporción de energía en la frecuencia i, y N es el número total de componentes frecuenciales. Una entropía espectral baja indica que la energía está concentrada en pocas frecuencias dominantes, lo que sugiere una señal estructurada y con menor presencia de ruido. Una alta entropía espectral implica una distribución más homogénea de la energía, característica habitual de señales ruidosas.

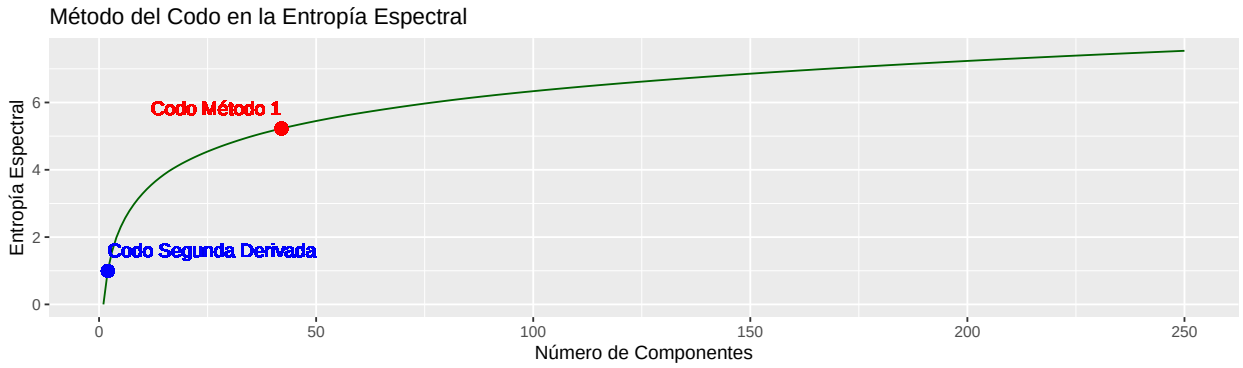
```
compute_spectral_entropy <- function(power_spectrum) {
  prob_dist <- power_spectrum / sum(power_spectrum)  # Normalización
  prob_dist <- prob_dist[prob_dist > 0]  # Evitar log(0)
  -sum(prob_dist * log2(prob_dist))  # Entropía de Shannon
}
```

Debido a que nos encontramos en un marco no supervisado, la entropía espectral surge como una herramienta intrínseca muy práctica para evaluar las frecuencias dominantes en nuestra señal ruidosa: con lo que poder elegir qué frecuencias priorizar en la selección de variables de nuestra matriz de diseño. Así, de las 250 componentes con mayor potencia obtenidas anteriormente, se seleccionó el número que mejor combinaba un alto contenido energético con una baja entropía espectral. Para ello se identificó un segundo codo basado en la minimización de la entropía espectral. Sin embargo, la curva en este caso era mucho más suave, y el resultado obtenido no fue concluyente. Al contrastarlo con un segundo método basado en la segunda derivada, se obtuvo un resultado diferente. Por ello, ambos puntos se establecieron como extremos de un intervalo para probar los siguientes pasos de forma manual, evaluando diferentes cantidades de frecuencias principales en la construcción de la matriz de diseño. Finalmente, se determinó que el número óptimo de frecuencias principales utilizadas se encontraba cercano a 10.

```
# --- Codo sobre la Entropía ---
# Calcular la entropía para los subconjuntos crecientes de frecuencias seleccionadas
entropies <- sapply(1:nrow(selected_freqs), function(i) {
  compute_spectral_entropy(selected_freqs$power[1:i])
})
# Aplicar el método del codo a la curva de entropía
```

```r
elbow_index_entropy <- find_elbow(1:length(entropies), entropies)
optimal_freqs <- selected_freqs[1:elbow_index_entropy, ]


# Función para encontrar el segundo codo usando la segunda derivada
find_second_elbow <- function(y) {
  # Calcular la primera derivada (pendiente)
  first_derivative <- diff(y)
  # Calcular la segunda derivada (curvatura)
  second_derivative <- diff(first_derivative)
  # Encontrar el índice donde la segunda derivada es máxima
  elbow_index <- which.max(abs(second_derivative)) + 1  # +1 porque 'diff' reduce el tamaño del vector
  return(elbow_index)
}
# --- Codo sobre la Entropía basado en la segunda derivada ---
# Calcular la entropía para los subconjuntos crecientes de frecuencias seleccionadas
entropies <- sapply(1:nrow(selected_freqs), function(i) {
  compute_spectral_entropy(selected_freqs$power[1:i])
})
# Aplicar el método del codo con segunda derivada
elbow_index_entropy_second_derivative <- find_second_elbow(entropies)
optimal_freqs_second_derivative <- selected_freqs[1:elbow_index_entropy_second_derivative, ]
# ---- Visualización ----
# Crear un dataframe con las entropías
df_entropy <- data.frame(Components = 1:length(entropies), Entropy = entropies)
# Graficar el segundo codo (Entropía) con ambos métodos
ggplot(df_entropy, aes(x = Components, y = Entropy)) +
  geom_line(color = "darkgreen") +
  geom_point(aes(x = elbow_index_entropy, y = entropies[elbow_index_entropy]),
             color = "red", size = 3) +
  geom_text(aes(x = elbow_index_entropy, y = entropies[elbow_index_entropy],
                label = "Codo Método 1"),
            vjust = -1, hjust = 1, color = "red") +
  geom_point(aes(x = elbow_index_entropy_second_derivative,
                 y = entropies[elbow_index_entropy_second_derivative]),
             color = "blue", size = 3) +
  geom_text(aes(x = elbow_index_entropy_second_derivative,
                y = entropies[elbow_index_entropy_second_derivative],
                label = "Codo Segunda Derivada"),
            vjust = -1, hjust = 0, color = "blue") +
  ggtitle("Método del Codo en la Entropía Espectral") +
  xlab("Número de Componentes") +
  ylab("Entropía Espectral")
```




```
print(paste("Primer codo (FFT) en la componente:", elbow_index_fft))
```

```
## [1] "Primer codo (FFT) en la componente: 250"
```

```
print(paste("Segundo codo (Entropía) en la componente:", elbow_index_entropy))
```

```
## [1] "Segundo codo (Entropía) en la componente: 42"
```

```
print(paste("Segundo codo (Entropía, segunda derivada) en la componente:", elbow_index_entropy_second_d
```

```
## [1] "Segundo codo (Entropía, segunda derivada) en la componente: 2"
```

Entropía global vs frecuencias principales

Además, la comparación entre las entropías obtenidas con los diferentes números de frecuencias principales ofreció información valiosa sobre el nivel de ruido presente. Puesto que la gran diferencia entre la entropía global calculada a partir de todas las frecuencias obtenidas tras la FFT, comparada con la de las frecuencias dominantes seleccionadas, indica que el ruido está dispersando la energía en el espectro. Por lo que se espera que la introducción de esta información en la selección de las variables de nuestro modelo sea beneficiosa.

```
global_entropy <- compute_spectral_entropy(fft_data$power)
codo_powers1 <- power_sorted[1:2]; codo_powers2 <- power_sorted[1:10]; codo_powers3 <- power_sorted[1:4
# Calcular la entropía de las componentes principales
codo_entropy1 <- compute_spectral_entropy(codo_powers1); codo_entropy2 <- compute_spectral_entropy(codo
print(paste("Entropía global:", round(global_entropy, 4)));
```

```
## [1] "Entropía global: 12.345"
```

```
print(paste("Entropía en el codo 2:", round(codo_entropy1, 4)))
```

```
## [1] "Entropía en el codo 2: 0.9947"
```

```
print(paste("Entropía en el codo 10:", round(codo_entropy2, 4)))
```

```
## [1] "Entropía en el codo 10: 3.2884"
```

```
print(paste("Entropía en el codo 42:", round(codo_entropy3, 4)))
```

```
## [1] "Entropía en el codo 42: 5.2261"
```

Pesos en la selección de variables para la matriz de diseño

De esta manera, la selección final de variables se realizó combinando la correlación con la señal ruidosa, y la distancia a las frecuencias dominantes identificadas mediante la FFT (que habían sido previamente seleccionadas priorizando aquellas con alto contenido energético y baja entropía espectral). Para aprovechar la información de este segundo componente se ponderaron los pesos de la siguiente forma:

```
best_weights <- c(0.2, 0.8) # correlación, proximidad con fft
num_total_vars <- 800
selected_prop <- 0.8
```

Asimismo, se mantuvo la inclusión de variables aleatorias para conservar la robustez del modelo y evitar el sobreajuste, pero se redujo su proporción.

Resultados

Para los diferentes números de frecuencias principales seleccionadas tras la aplicación de FFT y filtrado por entropía espectral se obtuvieron los siguientes valores de MSE respectivamente:

- 42 frecuencias: 3756
- 15 frecuencias: 3691
- 11 frecuencias: 3691
- 10 frecuencias: 3681
- 8 frecuencias: 3691
- 2 frecuencias: 3693

Los resultados obtenidos muestran que el enfoque propuesto ha logrado reducir significativamente el ruido en la señal del *cabritus hawaianus*.

Inicialmente, se probaron los modelos lm.ridge y cv.lars sin aplicar filtrado previo con la Transformada Rápida de Fourier (FFT) ni selección de variables basada en entropía espectral. En este primer enfoque, la matriz de diseño incluía todas las funciones trigonométricas generadas sin realizar una selección optimizada de las frecuencias más relevantes. Como resultado, estos métodos no lograron una regularización eficiente, obteniendo valores de MSE entre 4600 y 4000. Esto reflejaba que el modelo no alcanzaba un equilibrio óptimo entre ajuste y reducción de ruido, ya que en algunos casos se observaba sobreajuste y, en otros, una pérdida excesiva de información estructural en la señal.

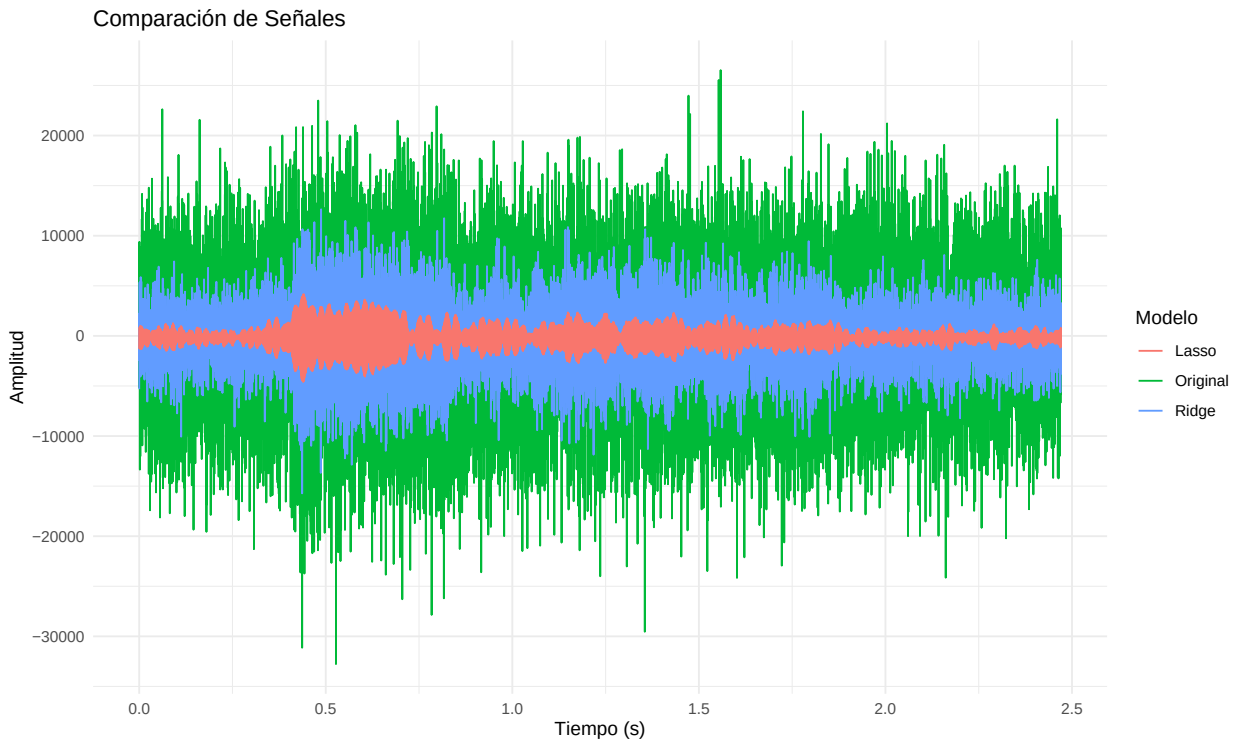
Los resultados obtenidos con lm.ridge y cv.lars no lograron un equilibrio óptimo entre reducción de ruido y conservación de la estructura de la señal. En contraste, cv.glmnet se empleó con un enfoque más refinado, incorporando siempre filtrado mediante *FFT y entropía espectral* para seleccionar de manera óptima las frecuencias más relevantes. Esta estrategia permitió optimizar la regularización del modelo y mejorar su capacidad de generalización, seleccionando automáticamente el parámetro λ mediante validación cruzada, evitando ajustes manuales y reduciendo el riesgo de sobreajuste. Además:

- Mayor estabilidad en Ridge, logrando una mejor reducción del ruido sin perder información estructural.
- Optimización en Lasso, seleccionando variables de manera más precisa sin eliminar información relevante.
- Mejora en los valores de MSE, obteniendo un *MSE de 3681* con Ridge, lo que representa una mejora significativa respecto a los métodos anteriores.

El uso de cv.glmnet junto con *FFT y entropía espectral* permitió una selección más precisa de los predictores, eliminando componentes irrelevantes y asegurando que la reconstrucción de la señal fuera más fiel a la estructura del alarido del cabritus hawaianus.

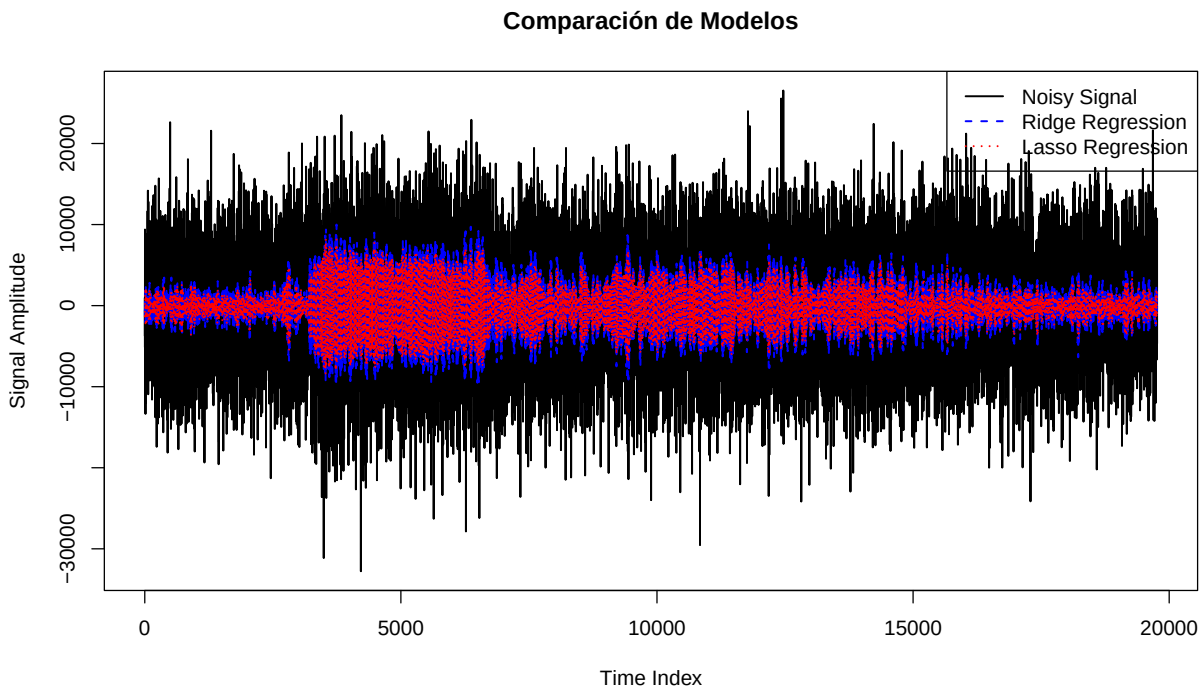
```
#lm.ridge y cv.lars
load("freq_amp.Rdata")  # Carga la matriz de diseño preprocesada
set.seed(123)
ridge_model1 <- lm.ridge(signal_noisy ~ freq.amp, lambda = 0.1)  # Ajuste Ridge
ridge_pred1 <- as.vector(predict(lm(signal_noisy ~ freq.amp), newdata = data.frame(freq.amp)))
lasso_model1 <- lars(freq.amp, signal_noisy, type = "lasso")  # Ajuste Lasso
lasso_pred1 <- predict(lasso_model1, newx = freq.amp, mode = "fraction", s = 0.1)$fit
p <- ggplot(data.frame(Time = seq_along(signal_noisy) / 8000, Original = signal_noisy, Ridge = ridge_pr
  geom_line(aes(y = Original, color = "Original")) +
  geom_line(aes(y = Ridge, color = "Ridge")) +
  geom_line(aes(y = Lasso, color = "Lasso")) +
```

```
  labs(title = "Comparación de Señales", x = "Tiempo (s)", y = "Amplitud", color = "Modelo") +
  theme_minimal()
print(p)
```




```
#Segundo método
top_n <- 10
selected_freqs <- fft_data[order(-fft_data$power), ][1:top_n, "freq"]
# Selección de variables en X
correlations <- apply(X, 2, function(col) abs(cor(col, signal_noisy)))
fourier_freqs <- rep(1:num_basis, each = 2) / N
freq_distances <- sapply(fourier_freqs, function(f) min(abs(f - selected_freqs)))
cor_norm <- (correlations - min(correlations)) / (max(correlations) - min(correlations))
freq_norm <- (1 - freq_distances / max(freq_distances))
total_score <- best_weights[1] * cor_norm + best_weights[2] * freq_norm
num_selected_vars <- round(num_total_vars * selected_prop)
num_random <- num_total_vars - num_selected_vars
selected_vars <- order(-total_score)[1:num_selected_vars]
set.seed(42)
random_vars <- sample(setdiff(1:ncol(X), selected_vars), num_random)
final_vars <- c(selected_vars, random_vars)
X_selected <- X[, final_vars]
# Ajuste lineal: Lasso y Ridge
ridge_model <- cv.glmnet(X_selected, signal_noisy, alpha = 0)
lasso_model <- cv.glmnet(X_selected, signal_noisy, alpha = 1)
# Predicciones
ridge_pred <- as.vector(predict(lm(signal_noisy ~ X_selected), newdata = data.frame(X_selected)))
lasso_pred <- predict(lasso_model, newx = X_selected, s = "lambda.min")
# Guardado y exportación
# ---Audio---
ridge_wav <- Wave(left = ridge_pred, samp.rate = 8000, bit = 16)
```

```r
lasso_wav <- Wave(left = as.vector(lasso_pred), samp.rate = 8000, bit = 16)
# ---Datos numéricos---
write.csv(ridge_pred, "ridge_predicciones.csv", row.names = FALSE)
write.csv(lasso_pred, "lasso_predicciones.csv", row.names = FALSE)
# Presentación de resultados
plot_results <- function(original_signal, ridge_pred, lasso_pred) {
  plot(original_signal, type = "l", col = "black", lwd = 1.5, main = "Comparación de Modelos",
       ylab = "Signal Amplitude", xlab = "Time Index")
  lines(ridge_pred, col = "blue", lwd = 1.5, lty = 2)
  lines(lasso_pred, col = "red", lwd = 1.5, lty = 3)
  legend("topright", legend = c("Noisy Signal", "Ridge Regression", "Lasso Regression"),
         col = c("black", "blue", "red"), lty = c(1, 2, 3), lwd = 1.5)
}
plot_results(signal_noisy, ridge_pred, lasso_pred)
```



Estos resultados sugieren que la combinación de filtrado espectral y regresión penalizada proporciona una mejora significativa en la reconstrucción de señales ruidosas, permitiendo una mayor fidelidad en la representación del alarido del *cabritus hawaianus*.

Discusión y conclusiones

Este estudio ha demostrado la viabilidad del filtrado no supervisado mediante modelos lineales con series de Fourier para la reconstrucción de señales ruidosas, específicamente en el caso del *cabritus hawaianus*. La combinación de regresión penalizada con Ridge y Lasso permitió reducir significativamente el ruido, logrando una reconstrucción más fiel de la señal original.

Uno de los avances clave en este trabajo fue la optimización de la selección de frecuencias principales mediante la Transformada Rápida de Fourier (FFT) y la entropía espectral. Inicialmente, se probaron métodos tradicionales como *lm.ridge* y *cv.lars*, obteniendo valores de MSE entre 4600 y 4000, lo que indicaba que estos enfoques no lograban un equilibrio óptimo entre ajuste y regularización. Posteriormente, la implementación

de *cv.glmnet* optimizó la selección del parámetro de regularización λ de manera automática, permitiendo obtener un MSE de 3680, lo que representa una mejora significativa respecto a los métodos anteriores.

El refinamiento del número óptimo de frecuencias principales seleccionadas tras la FFT, utilizando el método del codo y la minimización de la entropía espectral, fue fundamental para mejorar el ajuste del modelo. Se determinó que seleccionar aproximadamente 10 frecuencias principales permitía capturar la estructura relevante de la señal sin introducir ruido innecesario. Este resultado es clave para futuros estudios en los que se busque optimizar la reconstrucción de señales ruidosas a partir de componentes espectrales.

A pesar de estos avances, existen diversas oportunidades de mejora. En primer lugar, sería interesante refinar aún más los pesos asignados a los criterios de selección de variables en la matriz de diseño (correlación, similitud con la FFT y entropía espectral) para optimizar la reconstrucción. Asimismo, la inclusión de filtros convolucionales (Schmid, Rath, & Diebold, 2022) podría mejorar la calidad del ajuste en diferentes etapas del procesamiento:

- *Filtro sinc modificado:* Aplicable en una fase inicial para eliminar ruido de alta frecuencia sin distorsionar la señal subyacente.
- *Filtro de Savitzky-Golay:* Útil en el preprocesamiento para suavizar la señal mientras se preservan los detalles estructurales.
- *Filtro de Whittaker-Henderson:* Beneficioso en la fase final del ajuste para mejorar la suavidad de la señal reconstruida.

La combinación de estos filtros con el enfoque basado en modelos lineales y regresión penalizada podría proporcionar mejoras sustanciales en la fidelidad de la reconstrucción, reduciendo aún más el ruido y optimizando la extracción de información relevante del alarido del *cabritus hawaianus*.

Finalmente, una posible extensión futura de este trabajo podría incluir la exploración de técnicas no lineales, como redes neuronales o modelos de descomposición espectral avanzados, para mejorar la reconstrucción en presencia de ruido más complejo.

Bibliografía

Schmid, M., Rath, D., & Diebold, U. (2022). Why and how Savitzky–Golay filters should be replaced. ACS Measurement Science Au, 2(2), 185–196. https://doi.org/10.1021/acsmeasuresciau.1c00054

J. G. Liu and G. L. K. Morgan, "FFT Selective and Adaptive Filtering for Removal of Systematic Noise in ETM+ Imageodesy Images," in IEEE Transactions on Geoscience and Remote Sensing, vol. 44, no. 12, pp. 3716-3724, Dec. 2006, doi: 10.1109/TGRS.2006.881752. keywords: {Adaptive filters;Satellites;Remote sensing;Earthquakes;Pattern analysis;Instruments;Orbits;Frequency domain analysis;Fast Fourier transforms;Filtering;Destriping;Enhanced Thematic Mapper Plus (ETM+) imageodesy;fast Fourier transform (FFT) selective and adaptive filters;systematic noise removal},

Toh, A. M., Togneri, R., & Nordholm, S. (Año). Spectral entropy as speech features for speech recognition. The University of Western Australia & Western Australian Telecommunications Research Institute.