lonreg: Regresión lineal

con Template Model Builder (TMB)

Elmer Quispe-Salazar*

*Correspondencia. E-mail: qselmer@gmail.com

¿Qué es TMB?

TMB es una herramienta de programación en `R` y `C++` que permite implementar modelos con efectos aleatorios no lineales (incluyendo variables latentes). Al igual que ADMB (Automatic Differentiation Model Builder), TMB facilita la definición de la función de verosimilitud conjunta directamente en `C++`, aplica diferenciación automática, y emplea la aproximación de Laplace para obtener la verosimilitud marginal (modelos de efectos aleatorios y modelos jerárquicos) [Kristensen and Nielsen, 2023] . Además de permitir la paralelización de cálculos, acelerando la estimación de parámetros.

lonreg: Regresión lineal

El objetivo de este documento es proporcionar una serie de tutoriales que aborden diversos ejercicios propuestos en el repositorio de Template Model Builder (TMB), desde el más sencillo, como la regresión lineal simple (`linreg`), hasta el más complejo, como el modelo espacial Poisson GLMM, con una función de correlación que decae exponencialmente (`spatial`). Cada tutorial se enfocará en un ejemplo detallando su implementación y aplicación práctica. **Empezamos** con una caso práctico donde se compara los resultados obtenidos mediante TMB , lm[stats] , así como los valores verdaderos del modelo.

Caso práctico: discriminación morfométrica de peces pelágicos

Las características morfométricas se utilizan para describir la estructura poblacional de los peces. La variación en la forma del cuerpo, inducida por el ambiente, puede proporcionar información sobre la estructura poblacional. En este caso, exploramos cómo la condición corporal promedio de la sardina europea (*Sardina pilchardus*) se relaciona con un eje principal de variación morfométrica derivado de un análisis geométrico (Figura 1) .

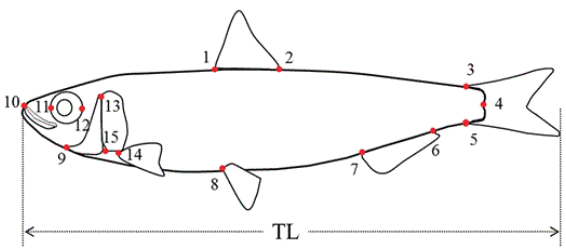


Figura 1: Localización de 15 landmarks en la sardina europea. Tomado de [Geladakis et al., 2017]

Para evaluar la relación entre la condición corporal promedio (CC) y los puntajes del primer componente principal canónico $(CAP1)$, ajustamos el siguiente modelo de regresión lineal simple:

$$Y = \beta_0 + \beta_1 \cdot x + \epsilon \tag{1}$$

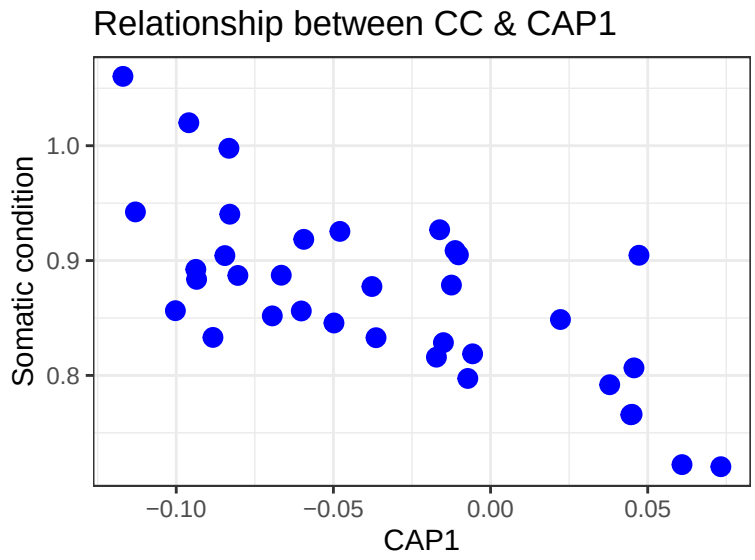
Los datos simulados se basaron [Geladakis et al., 2017], con los siguientes parámetros: **(1)** Intercepto (β_0) que representa el valor promedio de CC cuando $CAP1 = 0$. Para este conjunto de datos, el intercepto es $a = 0{,}8390$. **(2)** Pendiente (β_1) que representa la relación de incremento de $CAP1$ con respecto a CC. En este caso, la pendiente es $(b = -0{,}7336)$. **(3)** Desviación estándar (sd) que captura la variabilidad individual en los datos, es decir, las fluctuaciones en CC que no están explicadas directamente por $CAP1$. En este caso, la desviación estándar es $\sigma = 0{,}4800$.

Los datos simulados siguieron la relación lineal:

$$CC = a + b \cdot CAP1 + \mathcal{N}(0, \sigma^2) \tag{2}$$

Para lo cual:

```
library(pacman)
p_load(TMB, ggplot2, viridis)
set.seed(140423) # reproducibilidad
# datos simulados
a     <- 0.8390; b <-  -0.7336; sigma <- 0.05 # parámetros verdaderos
n     <- 35 # numero áreas muestreadas
x     <- runif(n, min = -0.12, max = 0.08)  # longitud total
noise <- rnorm(n, mean = 0, sd = sigma) #variabilidad individual
Y     <- a + b * x + noise # ancho corporal

data_df <- data.frame(x = x, Y = Y)
ggplot(data_df, aes(x = x, y = Y)) +
  geom_point(size = 3, color = "blue") +
  labs(title = "Relationship between CC & CAP1") +
  xlab("CAP1") + ylab("Somatic condition") +
  theme_bw()
```



Configuración del modelo

El archivo linreg.cpp es un modelo de regresión lineal desarrollado en C++ utilizando TMB. Este modelo estima tres parámetros: a, b, σ. La función objetivo es log-verosimilitud negativa (nll) de los datos observados, asumiendo errores con una distribución $\mathcal{N}(0, \sigma^2)$.

linreg.cpp:

```cpp
// Regresión lineal simple utilizando TMB.
#include <TMB.hpp> // Inclusión de la biblioteca TMB para modelos estadísticos.

template<class Type>
Type objective_function<Type>::operator() ()
{
  // Definición de los datos de entrada.
  DATA_VECTOR(Y); // Vector de observaciones de la variable dependiente.
  DATA_VECTOR(x); // Vector de observaciones de la variable independiente.

  // Definición de los parámetros del modelo.
  PARAMETER(a); // Intercepto de la regresión.
  PARAMETER(b); // Pendiente de la regresión.
  PARAMETER(logSigma); // Logaritmo de la desviación estándar de los errores.

  // Reporte de la varianza.
  ADREPORT(exp(2*logSigma)); // Reporta la varianza).

  // Cálculo de la función de verosimilitud negativa (NLL).
  Type nll = -sum(dnorm(Y, a + b * x, exp(logSigma), true)); // Suma de los logaritmos de las probab

  // Retorno de la función de verosimilitud negativa.
  return nll;
}
```

Compilar, cargar y MakeADFun

Al compilar y cargar el modelo linreg.dll, `MakeADFun` construye una función objetivo con derivadas en C++ que incorpora los datos y los parámetros iniciales. Al establecer `obj$hessian <- TRUE`, indicamos que, durante la optimización, también se calculará la matriz Hessiana.

```
compile("tmb/linreg.cpp")  # compilar
```

```
## using C++ compiler: 'G__~1.EXE (GCC) 12.3.0'
```

```
## [1] 0
```

```
dyn.load(dynlib("tmb/linreg")) # cargar el modelo

data        <- list(Y = Y, x = x) #datos
parameters  <- list(a = 0, b = 0, logSigma = log(1))  # parametros iniciales
obj         <- MakeADFun(data = data, parameters = parameters, DLL = "linreg")
obj$hessian <- TRUE
```

Optimización del modelo

Usamos optim para encontrar los valores que minimizan la función objetivo del modelo, y lm[stats] para un ajuste basado en mínimos cuadrados ordinarios, con fines comparativos.

```
# Optimizacion empleando TMB
start_time <- Sys.time()
opt        <- optim(obj$par, obj$fn, obj$gr)
end_time   <- Sys.time()
t_tmb      <- end_time - start_time
(opt)
```

```
## $par
##          a          b   logSigma
##  0.8361519 -0.9785962 -2.9578012
##
## $value
## [1] -53.8606
##
## $counts
## function gradient
##      152       NA
##
## $convergence
## [1] 0
##
## $message
## NULL
```

```
# Ajuste empleando lm
start_time <- Sys.time()
lmx        <- lm(Y ~ x, data = data_df)
```

```r
end_time   <- Sys.time()
t_lm       <- end_time - start_time
(lmx)
```

```
##
## Call:
## lm(formula = Y ~ x, data = data_df)
##
## Coefficients:
## (Intercept)            x
##      0.8362      -0.9786
```

Resultados

Los valores estimados utilizando `TMB` y `lm` se comparan con los valores verdaderos, mostrando tambiénel tiempo de procesamiento.

```r
results <- data.frame(
  Parámetros = c("Pendiente (b)", "Intercepto (a)",
                 "Desv. est. (sd)", "Tiempo (s)"),
  Simulación = c(b, a, sigma, NA),
  TMB        = c(opt$par[2], opt$par[1], exp(opt$par[3]), t_tmb),
  LM         = c(lmx$coefficients[2], lmx$coefficients[1],
                 summary(lmx)$sigma, t_lm)
)

knitr::kable(results, caption = "Parámetros estimados con TMB y lm", digits = 3)
```

Cuadro 1: Parámetros estimados con TMB y lm

	Parámetros	Simulación	TMB	LM
b	Pendiente (b)	-0.734	-0.979	-0.979
a	Intercepto (a)	0.839	0.836	0.836
logSigma	Desv. est. (sd)	0.050	0.052	0.053
	Tiempo (s)	NA	0.004	0.004

```r
(se <- sdreport(obj)) #Std. Error
```

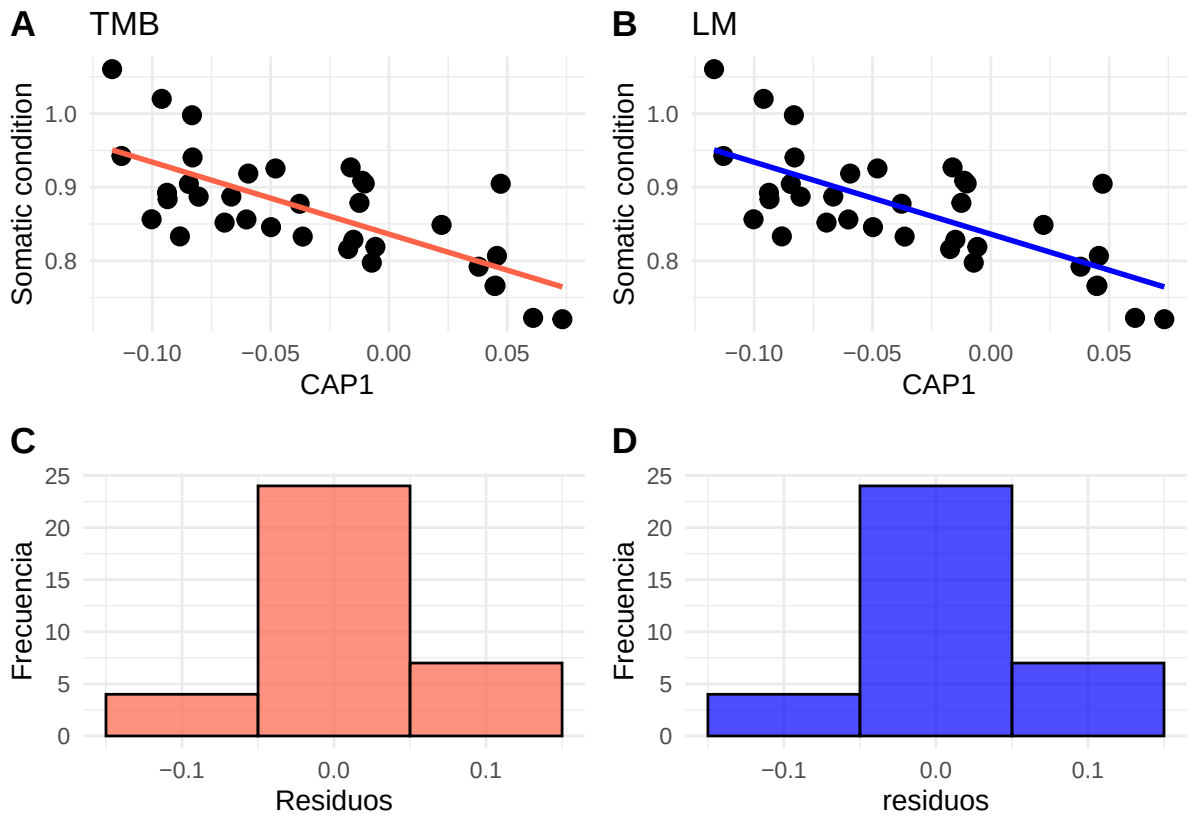
```
## outer mgc:  0.01475796
## outer mgc:  12.96246
## outer mgc:  12.99197
## outer mgc:  0.4518904
## outer mgc:  0.4223745
## outer mgc:  0.07073003
## outer mgc:  0.06926686
## outer mgc:  0.005394069
```

```
## sdreport(.) result
```

```
##             Estimate Std. Error
## a          0.8361519 0.01032871
## b         -0.9785962 0.16158123
## logSigma -2.9578012 0.11952419
## Maximum gradient component: 0.01475796
```

El ajuste del modelo se visualiza comparando los datos observados con las líneas de regresión ajustadas, además de la distribución de residuos, tanto para el modelo basado en `TMB` como para el modelo `lm`.

```
lm_y  <- lmx$coefficients[1] + lmx$coefficients[2] * x
tmb_y <- opt$par[1] + opt$par[2] * x

p1 <- ggplot(data_df, aes(x = x, y = Y)) +
  geom_point(size = 3, color = "black") +
  geom_line(aes(y = tmb_y), color = "tomato", linewidth = 1) +
  labs(title = "TMB") +
  xlab("CAP1") + ylab("Somatic condition") +
  theme_minimal()

p2 <- ggplot(data_df, aes(x = x, y = Y)) +
  geom_point(size = 3, color = "black") +
  geom_line(aes(y = lm_y), color = "blue", linewidth = 1) +
  labs(title = "LM") +
  xlab("CAP1") + ylab("Somatic condition") +
  theme_minimal()

# Calcular residuos para el modelo TMB
tmb_r <- data_df$Y - (opt$par[1] + opt$par[2] * data_df$x)

# Crear histograma de residuos
r1 <- ggplot(data.frame(residuos = tmb_r), aes(x = residuos)) +
    geom_histogram(binwidth = 0.1, fill = "tomato", color = "black", alpha = 0.7) +
    labs(title = "", x = "Residuos", y = "Frecuencia") +
    theme_minimal()

# Calcular residuos para el modelo lm
lm_r <- residuals(lmx)

# Crear histograma de residuos
r2 <- ggplot(data.frame(residuos = lm_r), aes(x = residuos)) +
  geom_histogram(binwidth = 0.1, fill = "blue", color = "black", alpha = 0.7) +
  labs(title = "", y = "Frecuencia") +
  theme_minimal()

cowplot::plot_grid(p1, p2, r1, r2,  labels = c("A", "B", "C", "D"),  ncol = 2)
```



Referencias

[Geladakis et al., 2017] Geladakis, G., Nikolioudakis, N., Koumoundouros, G., and Somarakis, S. (2017). Morphometric discrimination of pelagic fish stocks challenged by variation in body condition. *ICES Journal of Marine Science*, 75(2):711–718.

[Kristensen and Nielsen, 2023] Kristensen, K. and Nielsen, A. (2023). Template model builder (tmb).