Inferencia Conformal

Proyecto Final
Estadística no Paramétrica

Bajac, Matías Padín, Jimena

2025

Introducción

En el articulo de Tibshirani (2023) sobre inferencia conformal, el autor presenta un marco general para cuantificar la incertidumbre en problemas de predicción sin importar supuestos paramétricos sobre la distribución P de los datos. La idea central consiste en transformar cualquier predictor puntual en un predictor para conjuntos que garantice cobertura válida en muestras finitas.

En el contexto de regresión, esta metodología permite construir bandas de predicción que conservan la propiedad de cobertura deseada, independientemente del algoritmo utilizado para estimar la función de regresión.

Objetivos

Sea $(X_i, Y_i) \sim P, i = 1, ..., n$ iid, las variables explicativas y dependientes de una distribución P en $\mathcal{X} \times \mathcal{Y}$. Se podría pensar en la dimensión del conjunto de las variables explicativas en $\mathcal{X} = \mathbb{R}^d$, mientras que las variables dependientes en el espacio de todos los reales, $\mathcal{Y} = \mathbb{R}$. Dado una probabilidad α llamado tasa de no cobertura, queremos encontrar una banda de predicción

$$\hat{C}_n : \mathcal{X} \to \{subconjunto\ de\ \mathcal{Y}\}$$

Con la propiedad que para un nuevo par de datos $(X_{n+1}, Y_{n+1}) \sim P$

$$P(Y_{n+1} \in \hat{C}_n(x_{n+1})) \geq 1 - \alpha \qquad (1)$$

Donde la probabilidad es sobre los datos $(X_i, Y_i),\ i = 1, ..., n + 1$.

Por otra parte, si no asumimos ninguna teoría asintótica, ni tampoco ninguna distribución en P, obtener una cobertura exacta es algo muy difícil en general. Podríamos hacer algo totalmente trivial para obtenerla:

$$\hat{C}_n(X_{n+1}) = \begin{cases} \mathcal{Y} & \text{con probabilidad } 1 - \alpha, \\ \varnothing & \text{con probabilidad } \alpha. \end{cases}$$

Siempre tendrá cobertura exacta $1 - \alpha$, lo cual es, técnicamente, lo que queremos lograr con la ecuación (1).

La pregunta real sería, se puede lograr la ecuación (1) en muestras finitas sin asumir ninguna distribución P, haciendo algo *"no trivial"*? En particular, queremos que nuestra estrategia se adapte a la dificultad del problema en el siguiente sentido: cuanto más fácil sea predecir Y_{n+1} a partir de las variables explicativas X_{n+1}, mas chico nos gustaría que fuera el conjunto $\hat{C}_n(X_{n+1})$.

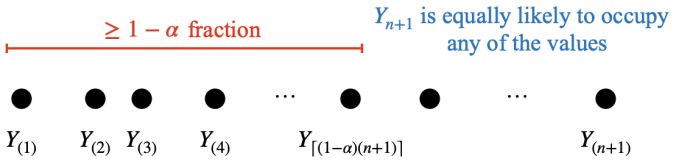
Supongamos que nuestro objetivo inicial es encontrar una cola de intervalo de predicción, $\hat{C}_n = (-\infty, \hat{q}_n]$.

Dada esta ecuación, un punto de partida natural sería fijar $\hat{q}_n$, como el cuantil muestral de nivel $1-\alpha$ de $Y_1, ..., Y_n$, el cual denotamos por

$$P(Y_{n+1} \leq \hat{q}_n) \approx 1 - \alpha$$

$$\hat{q}_n = \begin{cases} Y[(1-\alpha)(n+1)] \; si \; [(1-\alpha)(n+1)] \leq n, \\ \infty \; en \; caso \; contrario \end{cases}$$

Aquí $Y_{(1)}$, $Y_{(2)}, ...Y_{(n)}$ son los estadísticos de orden de la muestra Y_1, $Y_2, ...Y_n$ $(Y_{(1)} \leq Y_{(2)} \leq ..., Y_{(n)})$. Se verifica la cobertura en muestra finita de la ecuación (1) debido a la independencia de las variables. Por otra parte el rango de Y_{n+1} se distribuye uniforme en el conjunto $\{1, ..., n+1\}$, es decir, la predicción Y_{n+1} tiene probabilidad $\frac{1}{n+1}$ de caer en cualquier posición del conjunto ordenado.



Método de Naive

Veamos la primera idea clave sobre regresión, donde observamos ambos $X_i \in \mathcal{X}$ y $Y_i \in \mathbb{R}$, $i = 1, ..., n$, y queremos un conjunto de predicción para Y_{n+1} basado en X_{n+1}. Supongamos que $\hat{f}_n$ es cualquier predictor puntual, entrenado en (X_i, Y_i), $i = 1, ..., n$. En otras palabras, $\hat{f}_n(x)$ predice el valor de y que esperamos observar en x.

Definimos los residuos de los datos de entrenamiento,

$$R_i = |Y_i - \hat{f}_n(X_i)|, \; i = 1, ..., n$$

donde tenemos $\hat{q_n} = \lceil (1-\alpha)(n+1) \rceil$ el mas chico de $R_1, ...R_n$, podemos definir el conjunto de predicción como $\hat{C}_n(x) = \{y : |y - \hat{f}_n(x)| \leq \hat{q_n}\}$.

O en otras palabras:

$$\hat{C}_n(x) = [\hat{f}_n(x) - \hat{q}_n, \hat{f}_n(x) + \hat{q}_n]$$

Este método es aproximadamente válido para muestras grandes, bajo la condición de que $\hat{f}_n(x)$ sea los suficientemente preciso, es decir, que $\hat{q_n}$ este cerca del cuantil $1-\alpha$ de R_i.

Este método suele subestimar el ancho real del intervalo porque reutiliza los mismos datos tanto para entrenar el modelo como para medir los residuos. Esto viola el principio de independencia entre calibración y entrenamiento, produciendo residuos artificialmente pequeños y, por lo tanto, intervalos demasiado optimistas.

Precicción Full Conformal

Ahora hacemos algo distinto a lo que hacíamos hasta ahora, entrenamos nuestro algoritmo de predicción con los datos $(X_1, Y_1), ..., (X_n, Y_n), (x, y)$. Es decir, usamos un conjunto de entrenamiento extendido, con $n + 1$ puntos (incluyendo el nuevo par (x, y)).

A partir de este conjunto, obtenemos un predictor puntual $\hat{f}_n(x, y)$.

Definimos los residuos como:

- para $i = 1, ..., n$

$$R_i^{(x,y)} = |Y_i - \hat{f}_n(x, y)(X_i)|$$

- Para el nuevo punto de prueba:

$$R_{n+1}^{(x,y)} = |y - \hat{f}_n(x, y)(x)|$$

Finalmente, definimos el conjunto conformal como:

$$\hat{C_n}(x) = \{y \in R_{n+1}^{(x,y)} \leq [(1 - \alpha)(n + 1)] - \textit{ésimo menor de los } R_1^{((x,y))}, ..., R_n^{x,y}\}$$

Es decir, para cada valor candidato y, se crea un cojunto de entrenamiento aumentado:

$$\{(X_1, Y_1), ..., (X_n, Y_n), (X_{n+1}, y)\}$$

Luego se evalúa si el residuo del nuevo punto de la prueba $R_{n+1}^{(x,y)}$ está entre los más pequeños de todos los residuos generados con ese conjunto extendido. Si cumple la condición, entonces ese y pertenece al intervalo de predicción.

Otra desventaja importante es que el método Full Conformal produce intervalos marginales, no condicionales. Esto significa que los intervalos no dependen de X y pueden terminar siendo prácticamente horizontales cuando el modelo es rígido. Por eso, Full Conformal no es utilizado en la práctica.

Intervalos predicitivos vía Jacknife

Este algoritmo es bastante similar al *Leave One Out (LOOV)*. Aquí la idea es separar el conjunto en entrenamiento y testeo, dejando una observación aparte para testear, es decir, entrenas $\hat{f}_{-i}$ dejando fuera la observación i, luego se calculan el residuo confromal R_i sin esa observación.

Se procede a ordenar los residuos R_i y se calcula el k-ésimo valor más chico, donde $k = \lceil n(1 - \alpha) \rceil$, ese valor q vendria a ser la mitad del ancho del intervalo.

Por último, se entrena $\hat{f}$ con todos los datos y se devuelve el intervalo de predicción conformal para un nuevo punto $x \in \mathbb{R}^d$.

$$C_{jack}(x) = [\hat{f}(x) - \hat{d}, \hat{f}(x) + \hat{d}]$$

Simulaciones

```
library(tidymodels)
library(randomForest)
library(devtools)
```

Se simulan 5000 datos provenientes de mezcla de normales, en los cuales podemos observar que existe una relación lineal entre Y y X.

```
set.seed(2025)

n <- 5000

W <- rnorm(n)
X <- rnorm(n)
Y <- X + W

datos = data.frame(X,Y)
```

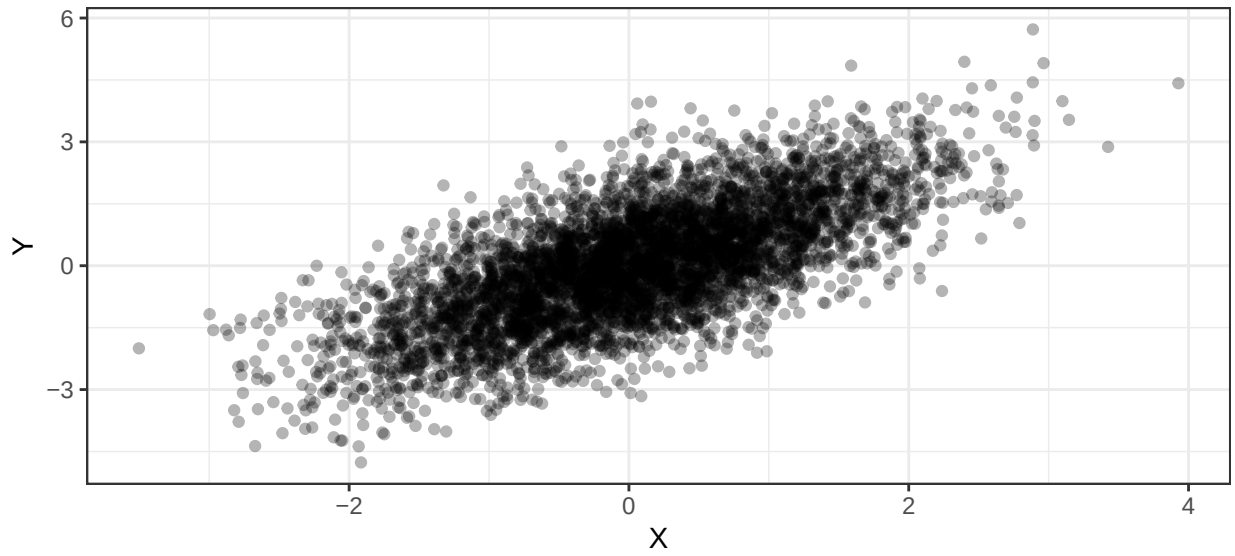
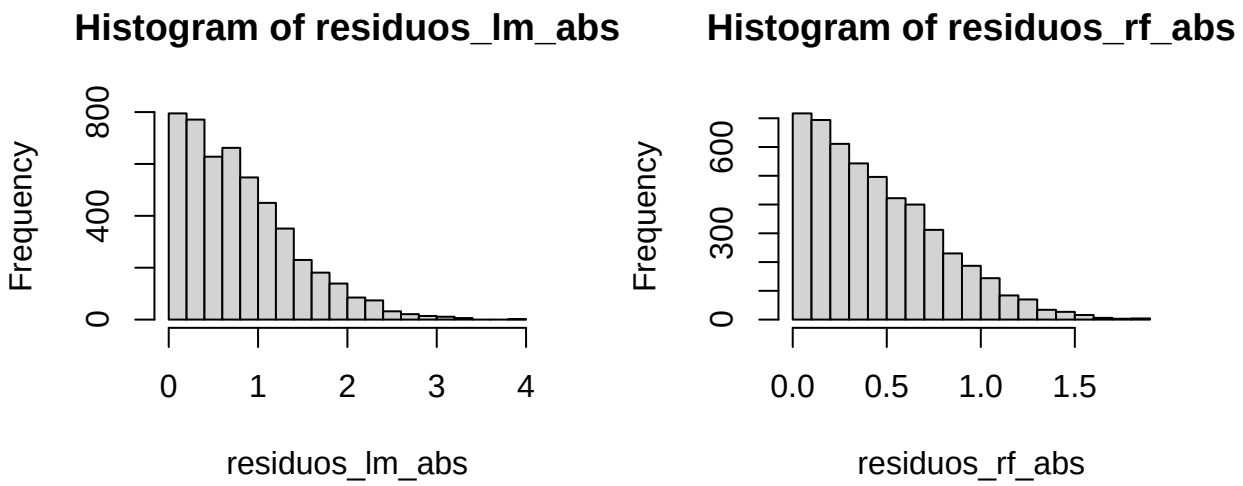


Figure 1: Grafico de puntos simulados (X,Y).

Se van a estimar los modelos para una *regresión lineal simple* (`lm`) y un *random forest* (`tidymodels`).

```
# Modelo lineal
lm_wf =workflow() %>%
  add_model(parsnip::linear_reg() %>% set_engine("lm")) %>%
  add_formula(Y ~ X)

lm_fit <- fit(lm_wf, data = datos)

residuos_lm= augment(lm_fit,new_data=datos) %>%
  select(.resid)
```

```r
# Random Forest
rf_spec <- rand_forest(mode = "regression") %>%
  set_engine("ranger")

rf_wf <- workflow() %>%
  add_model(rf_spec) %>%
  add_formula(Y ~ X)

rf_fit <- fit(rf_wf, data = datos)

pred_rf <- predict(rf_fit, new_data = datos) %>%
  pull(.pred)

residuos_rf <- datos$Y - pred_rf
```



Los histogramas de los residuos en valor absoluto del *modelo lineal* tiene mayor dispersión que el del *random forest*, lo que puede influir a la hora de hacer intervalos conformales, ya que estos dependen fuertemente del estimador.

Parte del código utilizado en esta sección se basa en el tutorial de CDSA-II sobre inferencia conformal (CDSA-II 2023), adaptado para el contexto de este trabajo.

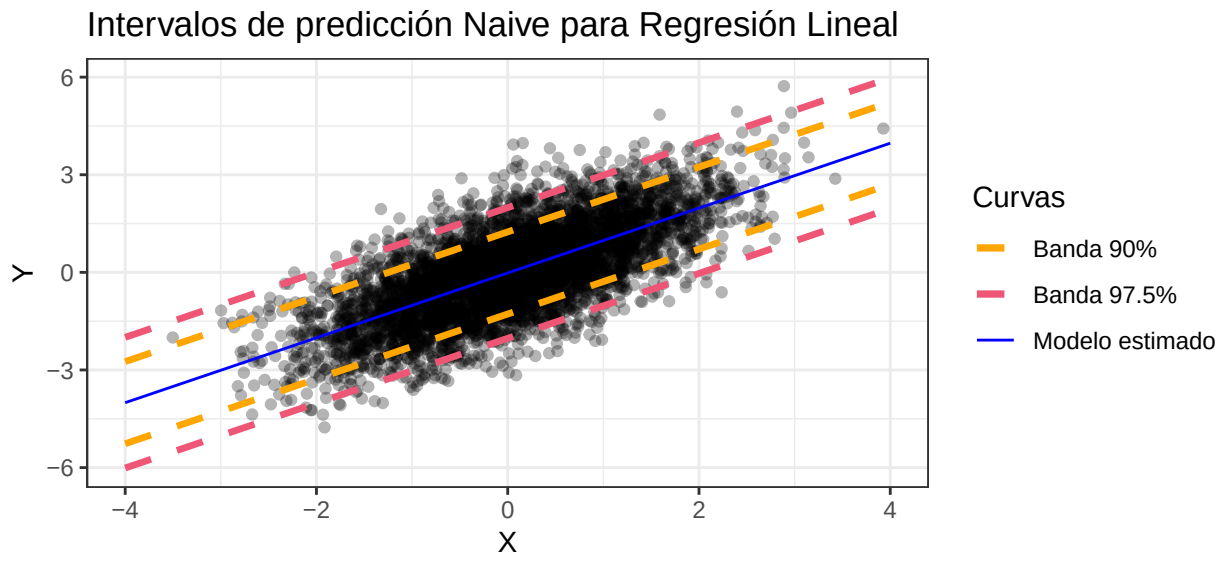
Metodo Naive

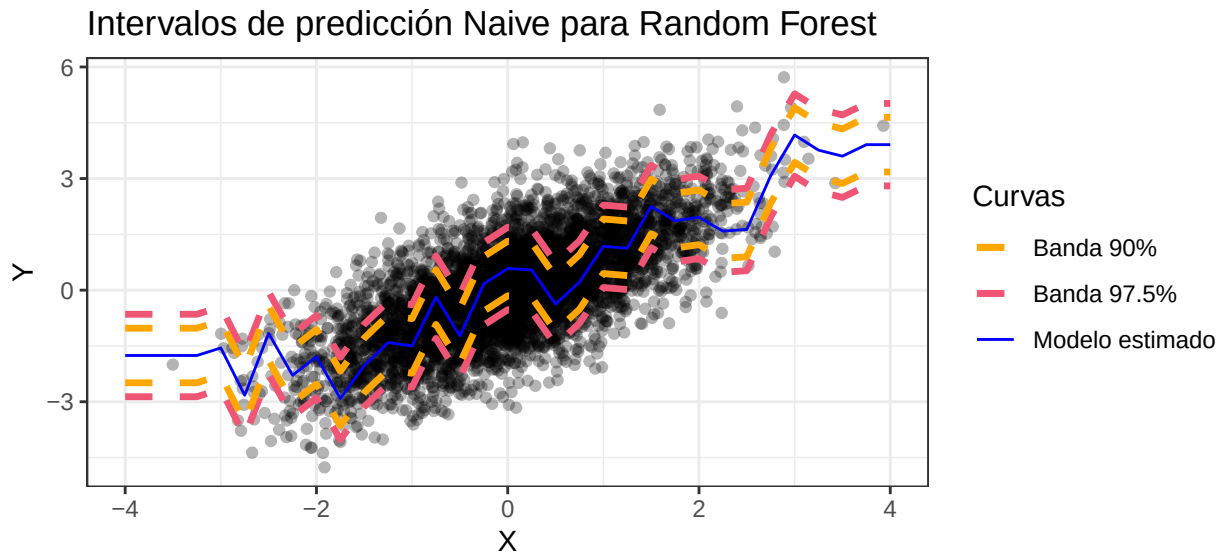
```r
# Definimos grilla
Xnew = seq(-4,4,.25)

# Funcion para calcular C.X
C.X <- function(muHat, residuos, alpha){
   lwr = muHat - quantile(residuos, probs = alpha)
   uppr = muHat + quantile(residuos, probs = alpha)
   return(tibble(lwr, uppr))}
```

```r
# Regresion lineal
muHat_lm <- predict(lm_fit,
                    new_data = data.frame(X = Xnew))

muHat_vector <- muHat_lm$.pred
residuosVec = residuos_lm$.resid

C.X_lm_975 <- C.X(muHat_vector, residuosVec, .975)
C.X_lm_90 <- C.X(muHat_vector, residuosVec, .90)
```

```r
# Random forest
muHat_rf <- predict(rf_fit,
                    new_data = data.frame(X = Xnew))

muHat_vector_rf <- muHat_rf$.pred

C.X_rf_975 <- C.X(muHat_vector_rf, residuos_rf, .975)
C.X_rf_90 <- C.X(muHat_vector_rf, residuos_rf, .90)
```

```r
resultados_conf_naive <- data.frame(
  Xnew,
  muHat_vector,
  muHat_vector_rf,
  C.X_lm_lwr975  = C.X_lm_975$lwr,
  C.X_lm_uppr975 = C.X_lm_975$uppr,
  C.X_lm_lwr90   = C.X_lm_90$lwr,
  C.X_lm_uppr90  = C.X_lm_90$uppr,
  C.X_rf_lwr975  = C.X_rf_975$lwr,
  C.X_rf_uppr975 = C.X_rf_975$uppr,
  C.X_rf_lwr90   = C.X_rf_90$lwr,
  C.X_rf_uppr90  = C.X_rf_90$uppr)
```





Como era de esperarse, la banda de confianza conformal para *Random Forest* es mas angosta que la del *Modelo Lineal* para ambas coberturas de predicción de un nuevo X.

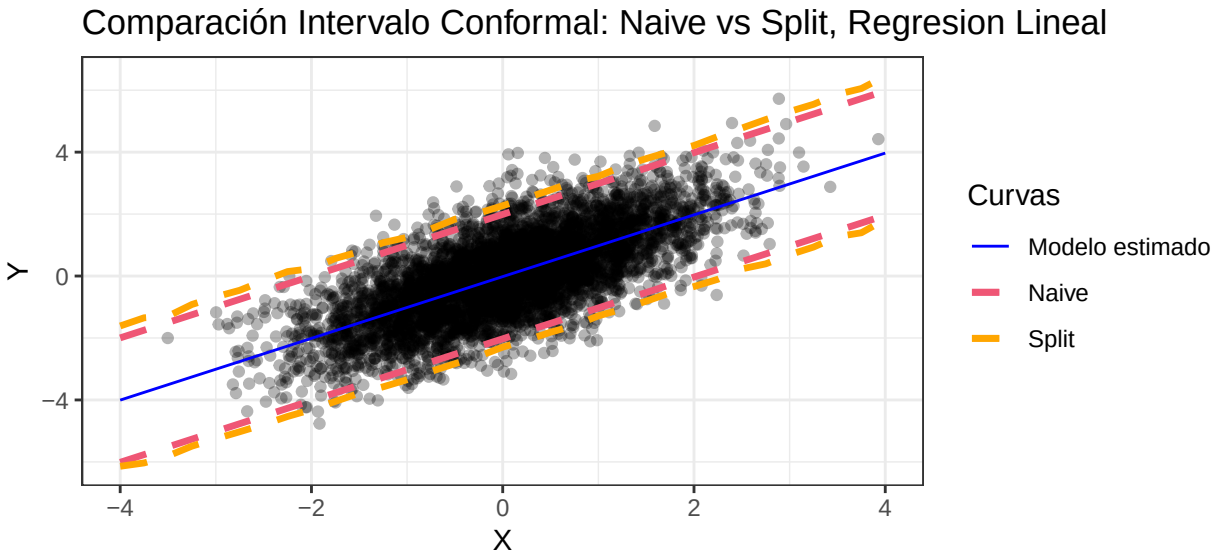
Split Conformal

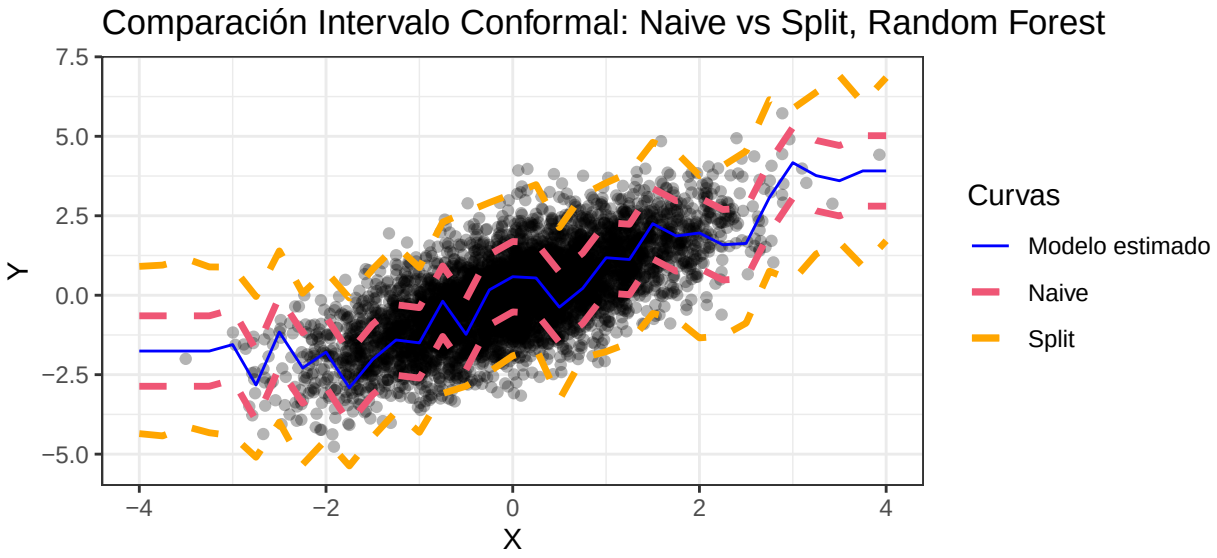
El método de intervalos conformales split es mas eficiente computacionalmente.

```
splitConfPredict <- function(Xin, modelo = c('lm', 'rf')) {
  nData <- nrow(datos)
  datos$index <- 1:nData
  datos$split <- 1
  datos$split[sample(datos$index, floor(nrow(datos) / 2), replace = F)] <- 2
  if (modelo=='lm'){
     fitlm.spl <- lm(Y ~ X, data = subset(datos, split == 1))
  } else {
     fitlm.spl <- randomForest(Y ~ X, data = subset(datos, split == 1))
  }
  resOut <- abs(
    subset(datos, split == 2)$Y -
      predict(fitlm.spl, newdata = subset(datos, split == 2))
  )
  kOut <- ceiling(((nData / 2) + 1) * (.975))
  resUse <- resOut[order(resOut)][kOut]

  Y.hat <- predict(fitlm.spl, newdata = data.frame(X = Xin))
  C.split <- c(Y.hat - resUse, Y.hat, Y.hat + resUse)
  return(C.split)
}
```

```
# Regresión lineal
Csplit_lm <- t(sapply(Xnew, FUN = function(x){splitConfPredict(x,modelo='lm')}))

# Random forest
Csplit_rf <- t(sapply(Xnew, FUN = function(x){splitConfPredict(x,modelo='rf')}))
```

```r
resultados_conf_split <- data.frame(
  Xnew,
  muHat_vector, # La estimación puntual es la misma
  muHat_vector_rf,
  Csplit_lm_lwr = Csplit_lm[, 1],
  Csplit_lm_uppr = Csplit_lm[, 3],
  Cxa_lm_lwr =   C.X_lm_975$lwr,
  Cxa_lm_uppr = C.X_lm_975$uppr,
  Csplit_rf_lwr = Csplit_rf[, 1],
  Csplit_rf_uppr = Csplit_rf[, 3],
  Cxa_rf_lwr =   C.X_rf_975$lwr,
  Cxa_rf_uppr = C.X_rf_975$uppr)
```





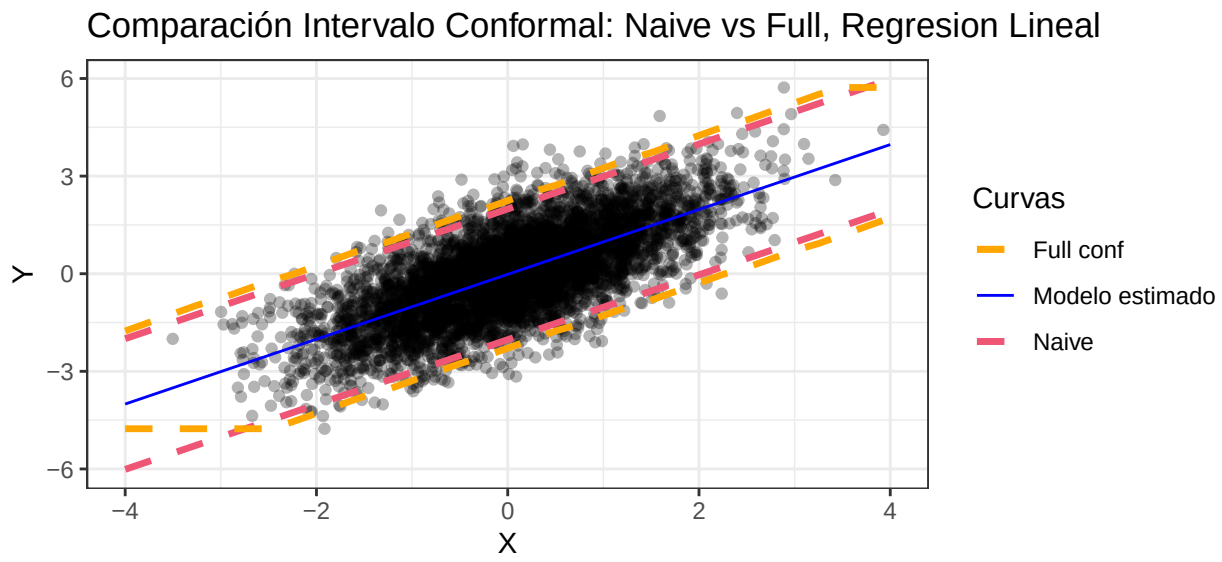
En ambos modelos se observa que Split mantiene una cobertura más conservadora que Naive, con bandas levemente más anchas, lo cual es coherente con el uso de un conjunto de calibración separado.

Full Conformal

No se va a presentar el intervalo Full Conformal para el modelo Random Forest debido a su costo computacional extremadamente alto. Para cada valor candidato y se debe reentrenar el modelo aumentado y calcular los residuos nuevamente, lo cual implica reentrenar cientos de árboles por cada uno de los 200 puntos en la grilla.

```
yCand <- seq(from=min(datos$Y), to=max(datos$Y), length=200)

fullconfPredict <- function(y, Xin, alpha, modelo=c('lm','rf')){
  nData <- nrow(datos)
  regData.a <- rbind(datos, data.frame(X = Xin, Y = y))
  if (modelo=='lm'){
     fit.a <- lm(Y~X, data=regData.a)
  } else {
     fit.a <- randomForest(Y~X, data=regData.a)
  }
  yhat   <- predict(fit.a, newdata = regData.a)
  resOut <- abs(regData.a$Y - yhat)
  resOut_new <- tail(resOut,1)
  pi.y <- pi.y <- mean(resOut <= resOut_new)
  testResult <- pi.y*(nData+1) <= ceiling(alpha*(nData+1))

  return(testResult)
}

get_full_interval <- function(Xin, modelo = c("lm", "rf"),alpha) {
  dentro <- sapply(yCand,FUN = function(y){fullconfPredict(y, Xin, modelo = modelo,alpha)})

  if (any(dentro)) {return(range(yCand[dentro]))
  } else {return(c(NA, NA))}
}
```

```
# Regresion lineal
Cxa_lm_mat <- t(sapply(Xnew, FUN = function(x) get_full_interval(x, modelo = "lm",alpha=.975)))

#Random forest
#Cxa_rf_mat <- t(sapply(Xnew, FUN = function(x) get_full_interval(x, modelo = "rf",alpha=.975)))

resultados_full_conf <- data.frame(
  Xnew,
  muHat_vector, # La estimacion puntual es la misma
  muHat_vector_rf,
  Cfull_lm_lwr = Cxa_lm_mat[, 1],
  Cfull_lm_uppr = Cxa_lm_mat[, 2],
  Cxa_lm_lwr =   C.X_lm_975$lwr,
  Cxa_lm_uppr = C.X_lm_975$uppr
#  Cfull_rf_lwr = Cxa_lm_mat[, 1],
#  Cfull_rf_uppr = Cxa_rf_mat[, 2],
#  Cxa_rf_lwr =   C.X_rf_975$lwr,
#  Cxa_rf_uppr = C.X_rf_975$uppr
)
```



Como se mencionó anteriormente, el Full conformal compara residuos globales en vez de centarrse en las predicciones puntuales. Para el modelo de regresión lineal (rígido), las bandas resultan más anchas y se adaptan muy poco a x, ofreciendo poca utilidad respecto a metodos mas simples como Split.

Jacknife

Para visualizar los intervalos generados por el método de Jacknife existen varios paquetes en R, tales como `conformalInference`, mamntenido por Ryan Tibshirani, `jackknifePlus` o `conformalInference.loo`, sin embargo estos están habilitados solo para Mac/Linux.

Conclusiones

En este trabajo se revisaron distintos métodos, presentados en el articulo de Tibshirani (2023), utilizados para construir intervalos de predicción conformales, resaltando su utilidad en situaciones donde no se desea asumir ninguna distribución sobre los datos ni depender de resultados asintóticos. Esto representa una ventaja importante en modelos donde la construcción de intervalos de predicción no es evidente, ya que en esos casos suele recurrirse al bootstrap para obtener aproximaciones, como ocurre en los métodos de ensamble, o bien se depende de supuestos más fuertes.

En la etapa de simulación con mezcla de normales, se analizaron las bandas conformales generadas para un modelo de regresión lineal simple y un random forest. Se observó que el método Naive tiende a producir intervalos demasiado optimistas debido a que utiliza los mismos datos para entrenar y evaluar el modelo, lo cual genera subcobertura. El método Split Conformal, en cambio, ofrece intervalos válidos en muestras finitas al separar explícitamente datos de entrenamiento y calibración, aunque a costa de utilizar menos información para ajustar el modelo. Asimismo, se implementó el método Full Conformal, destacando tanto su validez teórica como su elevado costo computacional.

En conjunto, los métodos explorados muestran cómo la inferencia conformal proporciona herramientas robustas y flexibles para cuantificar incertidumbre sin supuestos paramétricos. Entre ellos, Split Conformal se presenta como más eficiente entre validez teórica y costo computacional.

Bibliografía

CDSA-II. 2023. “Conformal Prediction Tutorial.” Online tutorial. https://cdsamii.github.io/cds-demos/conformal/conformal-tutorial.html.

Tibshirani, Ryan. 2023. "Advanced Topics in Statistical Learning." Lecture notes, Carnegie Mellon University. https://www.stat.cmu.edu/~ryantibs/.