Proyecto final

2025-11-07

```
library(tidymodels)
```

```
## -- Attaching packages --------------------------------------- tidymodels 1.2.0 --
```

```
## v broom        1.0.5      v recipes      1.0.10
## v dials        1.2.1      v rsample      1.2.1
## v dplyr        1.1.4      v tibble       3.2.1
## v ggplot2      3.5.0      v tidyr        1.3.1
## v infer        1.0.7      v tune         1.2.1
## v modeldata    1.4.0      v workflows    1.1.4
## v parsnip      1.2.1      v workflowsets 1.1.0
## v purrr        1.0.2      v yardstick    1.3.1
```

```
## -- Conflicts ------------------------------------------ tidymodels_conflicts() --
## x purrr::discard() masks scales::discard()
## x dplyr::filter()  masks stats::filter()
## x dplyr::lag()     masks stats::lag()
## x recipes::step()  masks stats::step()
## * Search for functions across packages at https://www.tidymodels.org/find/
```

```
library(randomForest)
```

```
## randomForest 4.7-1.2
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
##
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
## The following object is masked from 'package:dplyr':
##
##     combine
```

```
library(devtools)
```

```
## Loading required package: usethis
```

```
##
## Attaching package: 'devtools'
```

```
## The following object is masked from 'package:recipes':
##
##     check
```

```
colores <- c("#003f5c", "#7a5195", "#ef5
             675", "#ffa600")
```

Introducción

En el articulo de Ryan Tibsharini (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 coberturta válida en muestras finitas.

En el contexto de regresión, esta metodología permite contruir bandas de predicción que conserva la propiedad de cobertura deseada, independientemente del algoritmo usado para estimar la función de regresión

Objetivos

Sea $(X_i, Y_i) \sim P, i = 1, ..., n$ iid , las variables explicativas y dependiente de una distribucion P en $\mathcal{X}x\mathcal{Y}$. Podriamos pensar la dimension del conjunto de las variales explicativas en $\mathcal{X} = R^d$, mientras que la variable dependiente en el espacio de todos los reales $\mathcal{Y} = R$ 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 $(X_{n+1}, Y_{n+1}) \sim P$

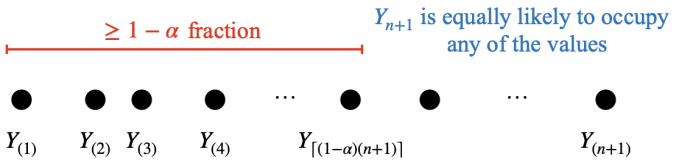
$$P(Y_{n+1} \in \hat{C}_n(x_{n+1})) \geq 1 - \alpha \ (1)$$

Donde la probabilidad es sobre los datos $(X_i, Y_i)\ i = 1, ..., n+1$ Por otra parte,si no asumimos ninguna teoría asintotica 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, lo que queremos lugrar con la ecuación (1).

La pregunta real sería, podriamos 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

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

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

Aquí $Y_{(1)}, Y_{(2)}, ...Y_n$ son los estadisticos de orden de la muestra $Y_{(1)} \leq Y_{(2)} \leq ..., Y_{(n)}$. 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 probabildiad $\frac{1}{n+1}$ de caer en cualquier posición del conjunto ordenado.

Método de Naive

Veamos la primera idea clave sobre regresion, donde observamos ambos X_i and $Y_i \ \in R \ i = 1, ..., n$, 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 resudios de los datos de entrenamiento,

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

tenemos $\hat{q_n} = [(1-\alpha)(n+1)]$ 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 condicion 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. Un problema de este método es que los intervalos de predicción pueden presentar una considerable subcobertura, dado que se estan empleando los residuos dentro de la muestra. Para evitar esto, se plantea la metodología de los intervalos de predicción conformales

Ejemplo práctico: Supongamos que tenemos un conjunto de datos $X = (1, 2, 3, 4)$ y $Y = (2, 4, 6, 8)$, donde ajustamos un modelo lineal simple $\hat{f}(x) = 1 + x$, y obtenemos $\hat{f}(x) = (2, 3, 4, 5)$ y los residuos $e_i = (0, 1, 2, 3)$ Luego calculamos el cuantil con cobertura $1-\alpha$, para ello calculamos $\hat{q_n} = [(0.8)(\ 5)] = 4$ Ordenamos los residuos de menos a mayor y vemos que el residuo $e_4 = 3$. Luego nos contruimos un intervalo de predicción para $X_{n+1} = 5$ $\hat{f(5)} = 6$. Por lo que el intervalo conformal con cobertura 0.8 es : $\hat{C(5)} = [6-3, 6+3] = [3, 9]$

Separación de la muestra de intervalos de prediccón

En esta priemra parte de esta sección, nos enfocaremos en la parte de regresion, es decir, que Y pertenece a todos los reales.

En concreto, dividimos el conjunto de entrenamiento en 2:

- T_1, es el conjunto de entrenamiento propimente dicho
- T_2 es el conjunto de calibración o testeo.

Tiene sentido pensar que la intersección de ambos es vacia, $T_1 \cap T_2 = \emptyset$ y $T_1 \cup T_2 = \{1, 2, ..., n\}$, sea $n_1 = |T_1| \ y \ \ n_2 = |T_2|$

En el siguiente paso, entrenamos el predictor puntual usando los datos del conjunto de entrenamiento propiamente dicho (X_i, Y_i) donde $i \in T_1$ y lo denotamos como $\hat{f}_{n_1}$.Luego, vemos los resudios en el conjunto de calibración: $e_i = |Y_1 - \hat{f}_{n_1}| \ i \in T_2$

Por lo que llegamos a que el residuo mas chico del cuantil conformal es el siguiente: $\hat{q}_{n_2} = [(1-\alpha)(n_2+1)]$ R_i $i \in R_2$. Aquí se calcula un cuantil de nivel de cobertura $1-\alpha$ de los residuos R_i, para construir un **intervalo de predicción** que tenga cobertura aproximada $1-\alpha$, es decir, primero se ordenan los residuos R_i del conjunto T_2 de menor a mayor. Luego, se busca el cuantil empírico de orden $(1-\alpha)$, es decir, el residuo en la posición

$[(1-\alpha)(n_2+1)]$ donde n_2 es el tamaño del conjunto de calibración. Este valor se denota como: $\hat{q}_{n_2}$ y se utiliza para crear un intervalo de predicción al rededor de la estimación puntual, $C(x) = [\hat{f}_{n_1}(x) - \hat{q}_{n_2}, \hat{f_{n_1}}(x) + \hat{q}_{n_2}]$

La mayor garantía que podemos obtener es que:

$$P(y_{n+1} \in \hat{C}_n(X_{n+1})|(X_i, Y_i)i \in T_1) \in [1-\alpha, 1-\alpha+\frac{1}{n_2+1}]$$

Esto es lo mismo a decir que para cualquier α, n y algoritmo, el valor predicho estara comprendido entre

$$1-\alpha \leq P(Y_{n+1} \in \hat{C}(_{X_{n+1}}) \leq 1-\alpha+\frac{1}{1+n_2})$$

Finalmente:

$$Y_{n+1} \leq \hat{C}_n(X_n+1) \Leftrightarrow e_{n+1} \leq \hat{q}_{n2} \Leftrightarrow e_{n+1} \leq [(1-\alpha)(n_2+1)$$

Esto ocurre con probabilidad al menos $1-\alpha$ y a al menos $1-\alpha+\ \frac{1}{n+1}$

Full conformal prediciton

Ahora hacemos algo distinto a lo que haciamos 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 punutal $\hat{f}_n(x, y)$

Luego, 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:

$$\hat{C}_n(x) = \{y \in R_{n+1}^{(x,y)} \leq [(1-\alpha)(n+1)] - \acute{e}simo\ menor\ de\ los\ R_1^{((x,y))}, ..., R_n^{x,y}\}$$

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.

Intervalos predicitivos vía Jacknife

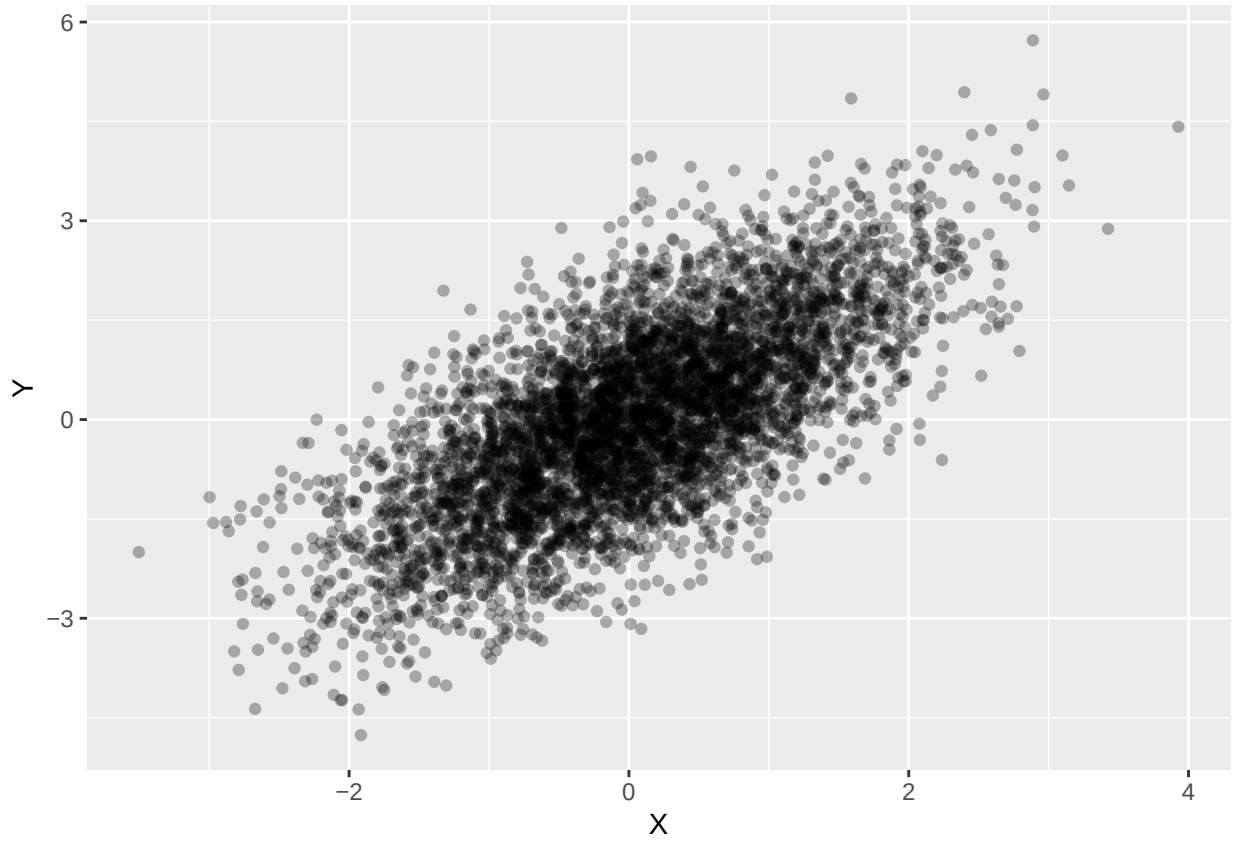
Este algoritmo es bastante similar al Leave One Out (LOOV). Aquí la idea es separar el conjunto en entrenamiento y testeo, dejando una observación aparte para testear, es decir, entrenas $\hat{f}_{-i}$ dejando fuera la observación i, luego se calculan el residuo confromal R_i sin esa observación. Luego se ordenan los residuos R_i y se calcula el k-ésimo valor más chico, donde $k = [n(1-\alpha)]$, 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 $x \in R^d$.

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

Simulaciones

Se simulan 5000 datos provenientes de mezcla de normales, la cual podemos observar que hay una relacion lineal entre Y y X.

```
set.seed(2025)

n <- 5000

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

Y <- X + W

datos = data.frame(X,Y)

 datos %>%  ggplot() + geom_point(aes(x=X,y=Y),alpha=0.3)
```



Hacemos estimaciones para una regresion lineal simple y un random forest (no paramétrico) y nos guardamos los resiudos, para ello se utiliza el paquete tidymodels.

Modelo lineal simple

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

lm_fit <- fit(lm_wf, data = datos)
```

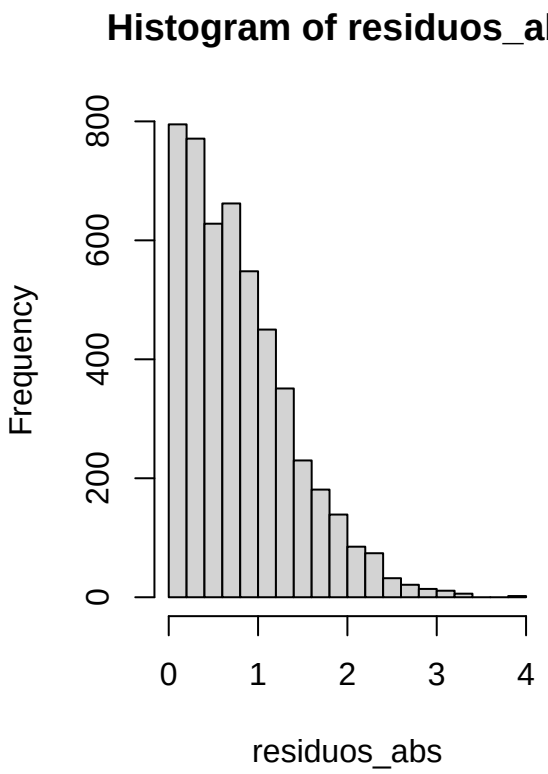
```
residuos= augment(lm_fit,new_data=datos) %>%
  select(.resid)
```

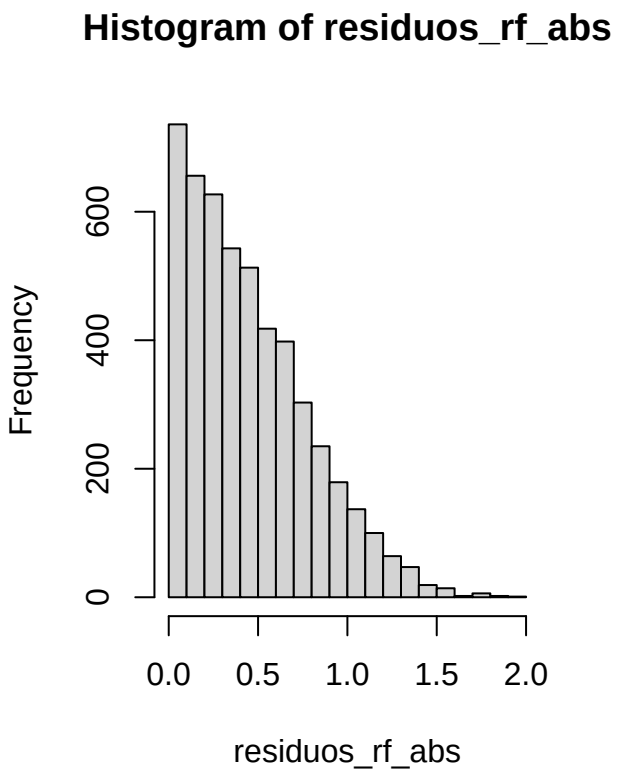
Random Forest

```
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
```

Vemos los histogramas de los resiudos tanto para un modelo de regresión lineal como también para el de random forest, podemos observar que los resiudos en valor absoluto del modelo lineal tiene mayor dispersion que el del random forest, esto puede influir a la hora de hacer intervalos conformales, ya que estos dependen fuertemente del estimador.

```
residuos_abs = abs(residuos$.resid)
residuos_rf_abs = abs(residuos_rf)
par(mfrow = c(1,2))
hist(residuos_abs)
hist(residuos_rf_abs)
```





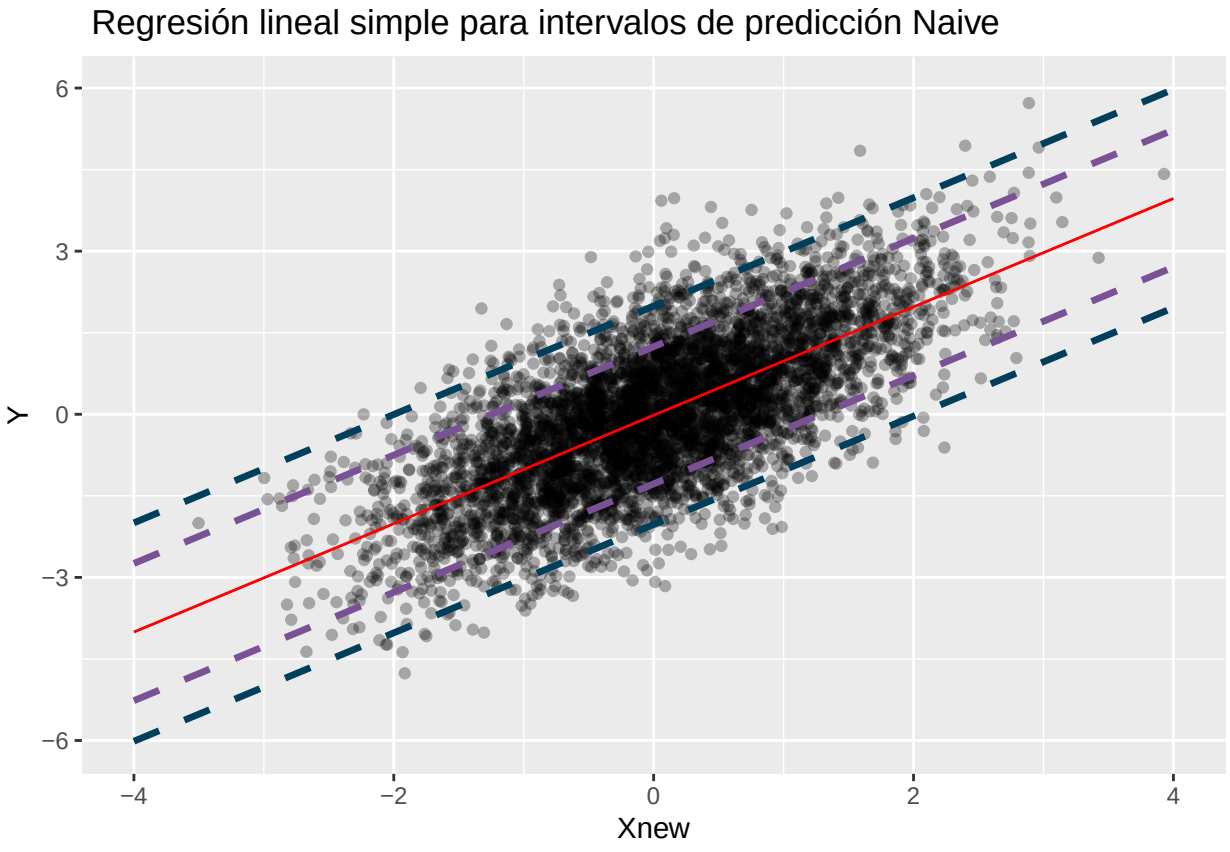
Metodo Naive

```
Xnew = seq(-4,4,.25)

muHat_lm <- predict(
  lm_fit,
  new_data = data.frame(X = Xnew)
)


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

C.X_lm_lwr_975 <- c(
  muHat_vector - quantile(residuosVec, .975)
)

C.X_lm_uppr_975 <- c(
  muHat_vector + quantile(residuosVec, .975)
)


C.X_lm_lwr_90 <- c(
  muHat_vector - quantile(residuosVec, .90)
)
```

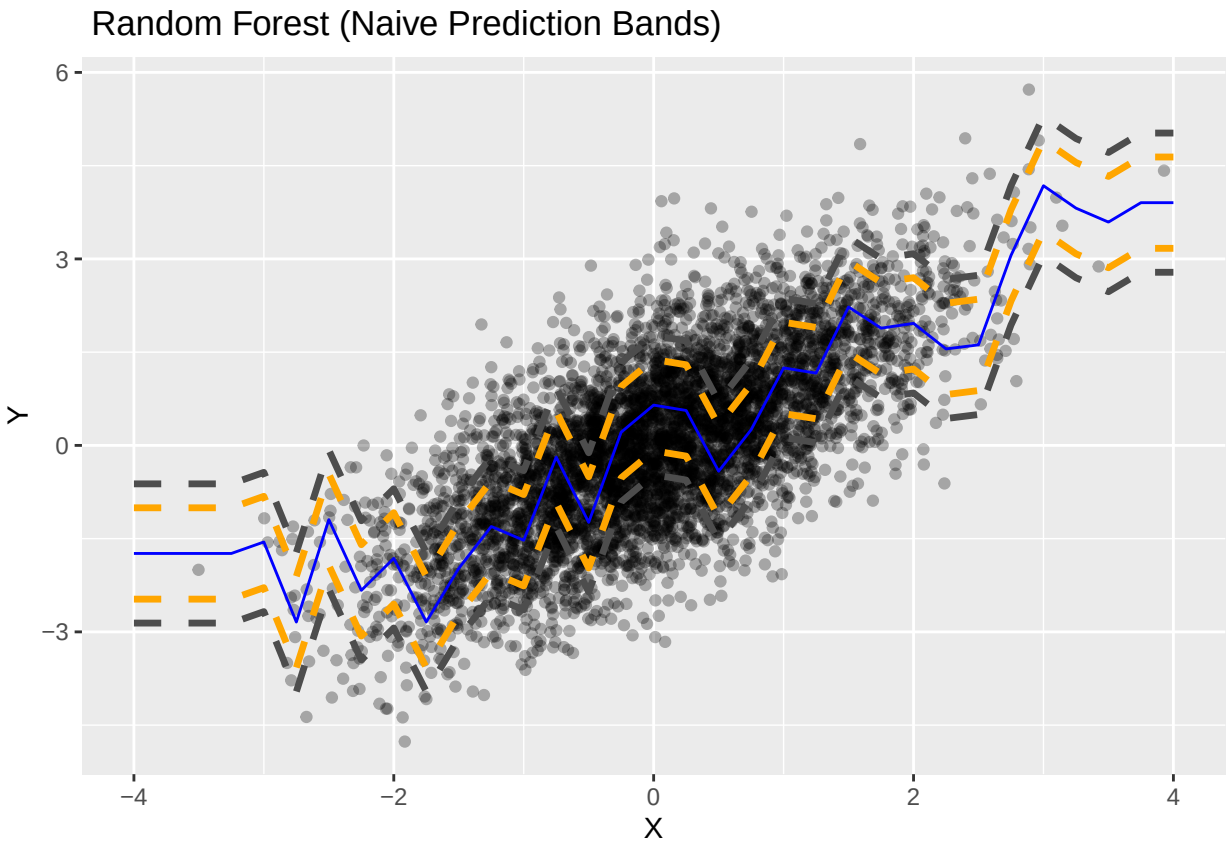
```r
C.X_lm_uppr_90 <- c(
  muHat_vector + quantile(residuosVec, .90)
)


## Random forest usando tidymodels

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

muHat_rf <- predict(
  rf_fit,
  new_data = data.frame(X = Xnew)
)


muHat_vector_rf <- muHat_rf$.pred


C.X_rf_lwr <- function(alpha) {
  muHat_rf - quantile(residuos_rf, probs = alpha)
}

C.X_rf_lwr_975 = C.X_rf_lwr(0.975)
C.X_rf_lwr90 = C.X_rf_lwr(0.90)

C.X_rf_lwr975 =C.X_rf_lwr_975$.pred
C.X_rf_lwr90=C.X_rf_lwr90$.pred

C.X_rf_uppr <- function(alpha)(
  muHat_rf + quantile(residuos_rf, probs=alpha)
)
C.X_rf_uppr_975 = C.X_rf_uppr(0.975)
C.X_rf_uppr90 = C.X_rf_uppr(0.90)

C.X_rf_uppr975 =C.X_rf_uppr_975$.pred
C.X_rf_uppr90=C.X_rf_uppr90$.pred

resultados_conf_naive <- data.frame(
  Xnew,
  muHat_vector,
  muHat_vector_rf,
```

```
  C.X_lm_lwr_975,
  C.X_lm_uppr_975,
  C.X_lm_lwr_90,
  C.X_lm_uppr_90,
   C.X_rf_lwr975,
  C.X_rf_uppr975,
  C.X_rf_lwr90,
  C.X_rf_uppr90
)


ggplot(resultados_conf_naive, aes(x = Xnew)) +
  geom_point(data = datos, aes(x = X, y = Y), color = "black",alpha=0.3) +  # puntos originales

  # Modelo Lineal
  geom_line(aes(y = muHat_vector), color = "red", size = 0.5) +
  geom_line(aes(y = C.X_lm_lwr_975), color = colores[1], linetype = "dashed",size=1.2) +
  geom_line(aes(y = C.X_lm_uppr_975), color = colores[1], linetype = "dashed",size=1.2) +
    geom_line(aes(y = C.X_lm_lwr_90), color = colores[2], linetype = "dashed",size=1.2) +
  geom_line(aes(y = C.X_lm_uppr_90), color = colores[2], linetype = "dashed",size=1.2)  + labs(title=" R
```

```
## Warning: Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.
## i Please use `linewidth` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```



```
ggplot(resultados_conf_naive, aes(x = Xnew)) +
  geom_point(data = datos, aes(x = X, y = Y), color = "black",alpha=0.3) +
# Modelo Random Forest
geom_line(aes(y = muHat_vector_rf), color = "blue",size = 0.5) +
geom_line(aes(y = C.X_rf_lwr975), linetype = "dashed",color = "gray30" ,size=1.2)+
geom_line(aes(y = C.X_rf_uppr975), color = "gray30", linetype = "dashed",size=1.2) +
  geom_line(aes(y = C.X_rf_lwr90), linetype = "dashed",color = colores[4] ,size=1.2)+
geom_line(aes(y = C.X_rf_uppr90), color = colores[4], linetype = "dashed",size=1.2) +


labs(title = " Random Forest (Naive Prediction Bands)",
       x = "X",
      y = "Y",
      )
```



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

##Split Conformal

El método de intervalos conformales split es mas eficiente computacionalmente.

Modelo Lineal

```
splitConfPredict <- function(Xin) {


  nData <- nrow(datos)
```

```
datos$index <- 1:nData
datos$split <- 1
datos$split[sample(datos$index, floor(nrow(datos) / 2), replace = F)] <- 2
  fitlm.spl <- lm(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)
}

# Intervalo split

Csplit <- t(sapply(Xnew, FUN = splitConfPredict))



# Resultados

resultados_conf_split <- data.frame(
  Xnew,
  muHat_vector, # La estimacion puntual es la misma
  Csplit_lwr = Csplit[, 1],
  Csplit_uppr = Csplit[, 3],
  Cxa_lm_lwr =   C.X_lm_lwr_975,
  Cxa_lm_uppr = C.X_lm_uppr_975
)
```

```
ggplot(resultados_conf_split , aes(x = Xnew)) +
  geom_point(data = datos, aes(x = X, y = Y), color = "black") +  # puntos originales

  # Modelo Lineal intervalo naive
  geom_line(aes(y = muHat_vector), color = "red", size = 1.2) +
  geom_line(aes(y = C.X_lm_lwr_975), color = "red", linetype = "dashed") +
  geom_line(aes(y = C.X_lm_uppr_975), color = "red", linetype = "dashed") +

  # intervalo conformal split
  geom_line(aes(y = Csplit_lwr), color = "blue", size = 1.2,linetype = "dashed") +
  geom_line(aes(y = Csplit_uppr ), color = "blue", linetype = "dashed") +


  labs(title = "Comparación: Regresión Lineal, intervalo conformal naive vs intervalo split",
       x = "X",
       y = "Y")
```

Lo mismo para el modelo de regresión Random Forest

```
splitConfPredict <- function(Xin) {
```

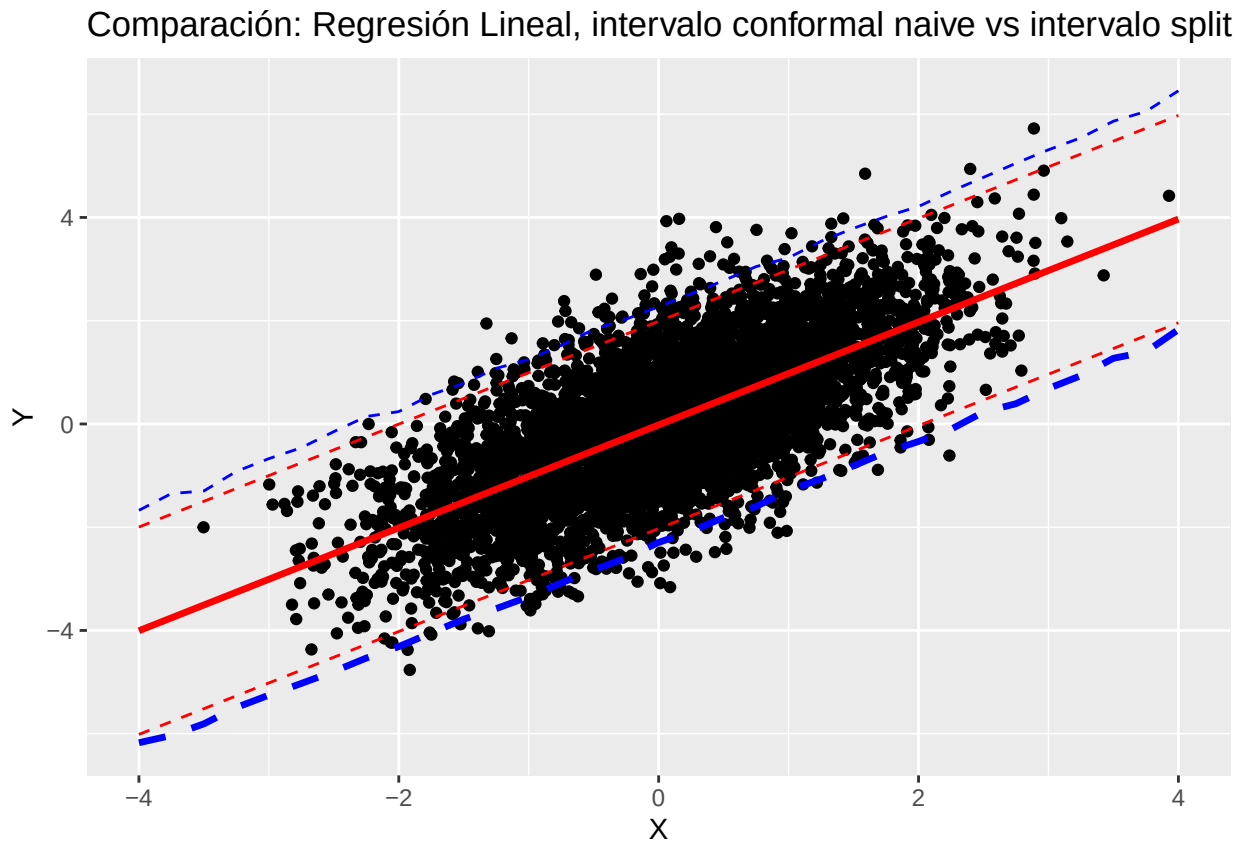


Figure 1: Intervalo de predición conformal Naive (rojo) vs Split (azul) para un modelo de regresion Lineal

```
  nData <- nrow(datos)
datos$index <- 1:nData
datos$split <- 1
datos$split[sample(datos$index, floor(nrow(datos) / 2), replace = F)] <- 2
  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)
}

# Intervalo split

Csplit <- t(sapply(Xnew, FUN = splitConfPredict))

# Resultados

resultados_conf_split <- data.frame(
  Xnew,
  muHat_vector_rf, # La estimacion puntual es la misma
  Csplit_lwr = Csplit[, 1],
  Csplit_uppr = Csplit[, 3],
  Cxa_rf_lwr =   C.X_rf_lwr975,
  Cxa_rf_uppr = C.X_rf_uppr975
)
```

```
ggplot(resultados_conf_split , aes(x = Xnew)) +
  geom_point(data = datos, aes(x = X, y = Y), color = "black") +  # puntos originales

  # Modelo Lineal intervalo naive
  geom_line(aes(y = muHat_vector_rf), color = "red", size = 1.2) +
  geom_line(aes(y = Cxa_rf_lwr), color = "red", linetype = "dashed") +
  geom_line(aes(y = Cxa_rf_uppr), color = "red", linetype = "dashed") +

  # intervalo conformal split
  geom_line(aes(y = Csplit_lwr), color = "blue", size = 1.2,linetype = "dashed") +
  geom_line(aes(y = Csplit_uppr ), color = "blue", linetype = "dashed")
```

Metrica de evaluacioón

```
resultados_conf_naive
```

```
##     Xnew muHat_vector muHat_vector_rf C.X_lm_lwr_975 C.X_lm_uppr_975
## 1  -4.00  -4.00536530      -1.7383502    -6.01376443    -1.996966169
## 2  -3.75  -3.75617535      -1.7383502    -5.76457448    -1.747776218
## 3  -3.50  -3.50698540      -1.7383502    -5.51538453    -1.498586268
## 4  -3.25  -3.25779545      -1.7383502    -5.26619458    -1.249396317
```

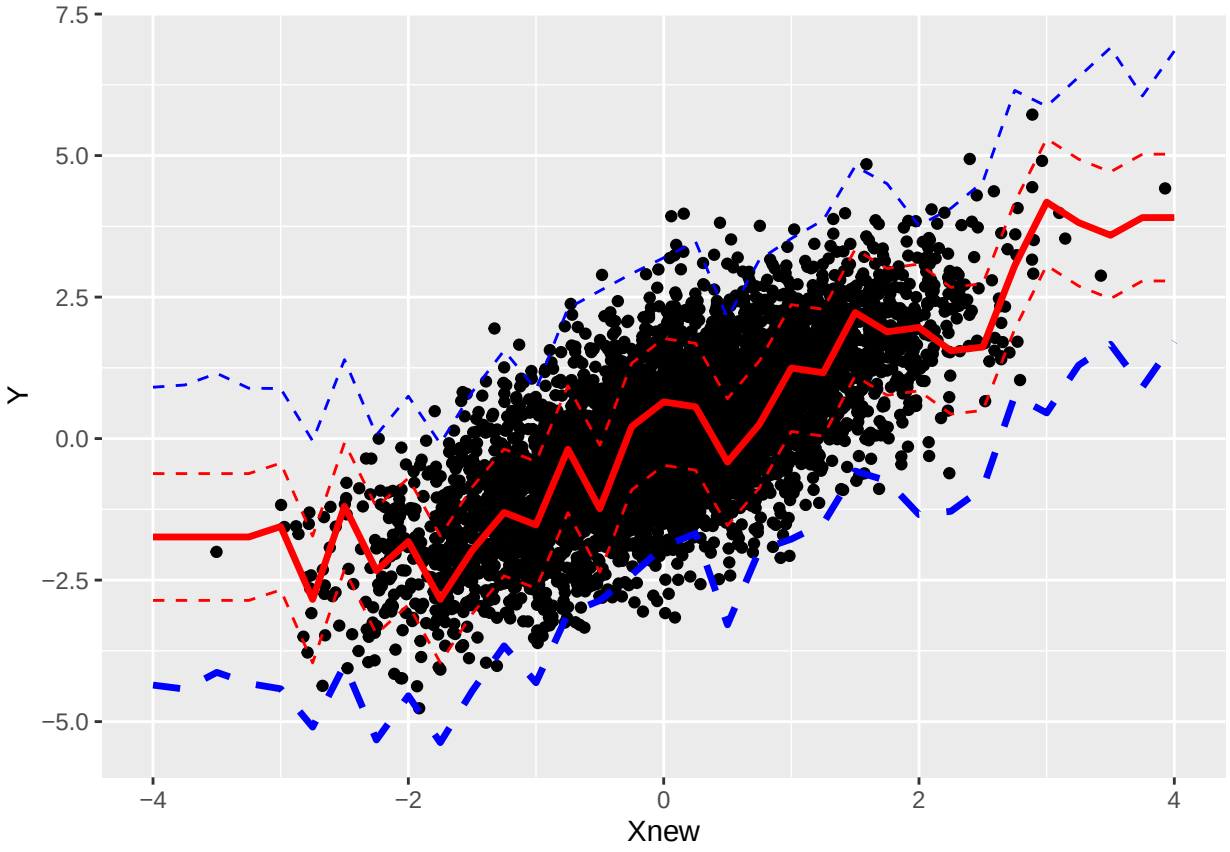


Figure 2: Intervalos de predicción conformal naive (rojo) vs split (azul) para un modelo Random Forest

```
## 5  -3.00  -3.00860550      -1.5537272    -5.01700463    -1.000206366
## 6  -2.75  -2.75941555      -2.8414010    -4.76781468    -0.751016416
## 7  -2.50  -2.51022560      -1.1940146    -4.51862473    -0.501826465
## 8  -2.25  -2.26103564      -2.3321343    -4.26943478    -0.252636514
## 9  -2.00  -2.01184569      -1.8170222    -4.02024482    -0.003446564
## 10 -1.75  -1.76265574      -2.8395355    -3.77105487     0.245743387
## 11 -1.50  -1.51346579      -1.9775241    -3.52186492     0.494933338
## 12 -1.25  -1.26427584      -1.3043341    -3.27267497     0.744123288
## 13 -1.00  -1.01508589      -1.5258247    -3.02348502     0.993313239
## 14 -0.75  -0.76589594      -0.1876383    -2.77429507     1.242503190
## 15 -0.50  -0.51670599      -1.2389672    -2.52510512     1.491693140
## 16 -0.25  -0.26751604       0.2126007    -2.27591517     1.740883091
## 17  0.00  -0.01832609       0.6483316    -2.02672522     1.990073042
## 18  0.25   0.23086386       0.5621132    -1.77753527     2.239262992
## 19  0.50   0.48005381      -0.4159153    -1.52834532     2.488452943
## 20  0.75   0.72924376       0.2523734    -1.27915537     2.737642894
## 21  1.00   0.97843371       1.2467090    -1.02996542     2.986832844
## 22  1.25   1.22762366       1.1630715    -0.78077547     3.236022795
## 23  1.50   1.47681362       2.2267139    -0.53158552     3.485212746
## 24  1.75   1.72600357       1.8859389    -0.28239556     3.734402696
## 25  2.00   1.97519352       1.9632704    -0.03320561     3.983592647
## 26  2.25   2.22438347       1.5519287     0.21598434     4.232782598
## 27  2.50   2.47357342       1.6171678     0.46517429     4.481972548
## 28  2.75   2.72276337       3.0557385     0.71436424     4.731162499
## 29  3.00   2.97195332       4.1775181     0.96355419     4.980352450
## 30  3.25   3.22114327       3.8163864     1.21274414     5.229542400
## 31  3.50   3.47033322       3.5920594     1.46193409     5.478732351
## 32  3.75   3.71952317       3.9046951     1.71112404     5.727922302
## 33  4.00   3.96871312       3.9046951     1.96031399     5.977112252
##    C.X_lm_lwr_90 C.X_lm_uppr_90 C.X_rf_lwr975 C.X_rf_uppr975 C.X_rf_lwr90
## 1    -5.26732004   -2.743410562   -2.85825967    -0.61844080  -2.47306451
## 2    -5.01813009   -2.494220612   -2.85825967    -0.61844080  -2.47306451
## 3    -4.76894013   -2.245030661   -2.85825967    -0.61844080  -2.47306451
## 4    -4.51975018   -1.995840710   -2.85825967    -0.61844080  -2.47306451
## 5    -4.27056023   -1.746650760   -2.67363668    -0.43381781  -2.28844153
## 6    -4.02137028   -1.497460809   -3.96131044    -1.72149157  -3.57611528
## 7    -3.77218033   -1.248270858   -2.31392404    -0.07410517  -1.92872888
## 8    -3.52299038   -0.999080908   -3.45204376    -1.21222489  -3.06684861
## 9    -3.27380043   -0.749890957   -2.93693159    -0.69711272  -2.55173644
## 10   -3.02461048   -0.500701006   -3.95944492    -1.71962605  -3.57424977
## 11   -2.77542053   -0.251511056   -3.09743349    -0.85761462  -2.71223833
## 12   -2.52623058   -0.002321105   -2.42424353    -0.18442466  -2.03904837
## 13   -2.27704063    0.246868846   -2.64573410    -0.40591523  -2.26053895
## 14   -2.02785068    0.496058796   -1.30754778     0.93227109  -0.92235263
## 15   -1.77866073    0.745248747   -2.35887665    -0.11905778  -1.97368150
## 16   -1.52947078    0.994438697   -0.90730869     1.33251018  -0.52211354
## 17   -1.28028083    1.243628648   -0.47157779     1.76824108  -0.08638264
## 18   -1.03109087    1.492818599   -0.55779622     1.68202265  -0.17260107
## 19   -0.78190092    1.742008549   -1.53582472     0.70399415  -1.15062957
## 20   -0.53271097    1.991198500   -0.86753602     1.37228285  -0.48234086
## 21   -0.28352102    2.240388451    0.12679954     2.36661841   0.51199469
## 22   -0.03433107    2.489578401    0.04316205     2.28298092   0.42835720
## 23    0.21485888    2.738768352    1.10680449     3.34662335   1.49199964
## 24    0.46404883    2.987958303    0.76602949     3.00584836   1.15122465
```

```
## 25    0.71323878    3.237148253    0.84336096     3.08317983   1.22855612
## 26    0.96242873    3.486338204    0.43201922     2.67183809   0.81721438
## 27    1.21161868    3.735528155    0.49725837     2.73707724   0.88245353
## 28    1.46080863    3.984718105    1.93582906     4.17564793   2.32102421
## 29    1.70999858    4.233908056    3.05760864     5.29742751   3.44280379
## 30    1.95918853    4.483098007    2.69647694     4.93629581   3.08167210
## 31    2.20837848    4.732287957    2.47214993     4.71196880   2.85734508
## 32    2.45756843    4.981477908    2.78478567     5.02460454   3.16998083
## 33    2.70675838    5.230667859    2.78478567     5.02460454   3.16998083
##    C.X_rf_uppr90
## 1     -1.0036359
## 2     -1.0036359
## 3     -1.0036359
## 4     -1.0036359
## 5     -0.8190130
## 6     -2.1066867
## 7     -0.4593003
## 8     -1.5974200
## 9     -1.0823079
## 10    -2.1048212
## 11    -1.2428098
## 12    -0.5696198
## 13    -0.7911104
## 14     0.5470759
## 15    -0.5042529
## 16     0.9473150
## 17     1.3830459
## 18     1.2968275
## 19     0.3187990
## 20     0.9870877
## 21     1.9814233
## 22     1.8977858
## 23     2.9614282
## 24     2.6206532
## 25     2.6979847
## 26     2.2866429
## 27     2.3518821
## 28     3.7904528
## 29     4.9122324
## 30     4.5511007
## 31     4.3267736
## 32     4.6394094
## 33     4.6394094
```

```
mean(if_else(resultados_conf_naive$Xnew >= resultados_conf_naive$C.X_lm_lwr_975 &  resultados_conf_naive
```

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

```
mean(if_else(resultados_conf_naive$Xnew >= resultados_conf_naive$C.X_rf_lwr975 & resultados_conf_naive$
```

```
## [1] 0.7878788
```

para split conformal

Conclusiones:

Con este articulo se ha visto la metodología de los distintos métodos para cuantificar los intervalos de confianza, estos algoritmos requieren asumir que los datos tienen independencia entre sí para cualquier modelo de regresión,pero sin suponer ninguna distribución en los datos ni tampoco una distrbución de probabilidad para los residuos.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 fuertes sobre la distribución de los datos basados en resultados asintóticos. En cuanto a la etapa de simulación de mezcla de normales, vimos los intervalos de predicción conformal para un modelo de regersion lineal y un random forest, siendo que el método de separacion de muestras es mas eficiente computacionalmente y en particular, para el random forest bajo el metodo de split, los intervalos son mas amplios que con el método Naive, esto es porque el método de split Conformal construye los intervalos usando una muestra de calibración separada, por lo que no reutiliza los datos de entrenamiento. Esto garantiza cobertura válida en muestras finitas bajo independencia de los datos.