Inferencia Conformal

Mauro Loprete y Maximiliano Saldaña

diciembre 2022

Tabla de contenidos

1 Introducción **3**

2 Inferencia conformal **4**
2.1 Un resultado previo 4
2.2 Método *naive* de construcción de intervalos 4
2.3 Intervalos de predicción conformales 5
2.3.1 Teorema 6
2.4 Intervalos de predicción conformales con muestras separadas 7
2.4.1 Teorema 7
2.4.2 Teorema 8
2.5 Intervalos conformales con múltiples separaciones de la muestra 8
2.6 Intervalos predictivos mediante Jackknife 9

3 Aplicación **10**
3.1 Método naive 11
3.2 Método conformal simple 14
3.3 Split 16

4 Conclusión **18**

5 Bibliografía **19**

A Anexo **20**
A.1 Paquete conformalInference 20
A.2 Implementación de un modelo de regresión lineal 21
A.3 Ejemplo para implementar Nadaraya–Watson 22
A.3.1 Ejemplo de predicción conformal para Nadaraya–Watson 23
A.4 Ejemplo regresión múltiple 29

1 Introducción

En Lei et al. (2016) los autores plantean un marco general para realizar inferencia predictiva sin supuestos distribucionales en un contexto de regresión, empleando la *inferencia conformal*. Mediante la metodología planteada se pueden obtener intervalos de confianza con validez en muestra finitas (no asintótica) para una variable de respuesta, empleando cualquier estimador de la función de regresión.

El problema se plantea de la siguiente manera: Se considera $Z_1, ..., Z_n \sim F$ i.i.d., donde $Z_i = (X_i, Y_i)$ es una variable aleatoria en $\mathbb{R}^d \times \mathbb{R}$, Y_i es la variable de respuesta y $X_i = X_i(1) ..., X_i(d)$ son las covariables. Se tiene tiene la función de regresión:

$$\mu(x) = E(Y|X = x), \quad x \in \mathbb{R}^d$$

Es de interés predecir la nueva respuesta Y_{n+1} a las covariables X_{n+1}, sin hacer supuestos sobre μ o F. Dado un nivel de cobertura α, el objetivo es construir un intervalo de predicción $C \subseteq \mathbb{R}^d \times \mathbb{R}$ basado en $Z_1, ..., Z_n$ que cumpla:

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

En esta expresión se supone que $Z_{n+1} = (X_{n+1}, Y_{n+1})$ proviene también de la distribución F y $C(x) = \{y \in \mathbb{R} : (x, y) \in C\}, \ x \in \mathbb{R}^d$

2 Inferencia conformal

La idea básica de la inferencia conformal, dadas las definiciones de la introducción, consiste en que para decidir si un valor y está incluido en el intervalo $C(X_{n+1})$ consideramos poner a prueba la hipótesis nula de que $Y_{n+1} = y$ y se construye un valor-p válido basado en los cuantiles empíricos de la muestra aumentada $Z_1, \dots, Z_n, Z_{n+1}$.

2.1 Un resultado previo

Sean $U_1, \dots, U_n$ una muestra i.i.d de una variable aleatoria continua. Para un nivel de no cobertura $\alpha \in (0, 1)$ y una observación U_{n+1}, nótese que:

$$P(U_{n+1} \leq \hat{q}_{1-\alpha}) \geq 1 - \alpha \tag{2.1}$$

Donde $\hat{q}_{1-\alpha}$ es el cuantil de la muestra $U_1, \dots, U_n$ definido por:

$$\hat{q}_{1-\alpha} = \begin{cases} U_{(\lceil (n+1)(1-\alpha) \rceil)} & \text{si } \lceil (n+1)(1-\alpha) \rceil \leq n \\ \infty & \text{en caso contrario} \end{cases} \tag{2.2}$$

Aquí $U_{(1)} \leq \dots \leq U_{(n)}$ son los estadísticos de orden de la muestra. Se verifica la cobertura en muestra finita de la Ecuación 2.1: dada la independencia de las variables, el rango de U_{n+1} en la muestra se distribuye uniforme en el conjunto $\{1, \dots, n+1\}$, entonces en nuestro problema de regresión observamos una muestra $Z_i = (X_i, Y_i) \in \mathbb{R}^d \times \mathbb{R} \sim P$ para la construcción de un intervalo de predicción para Y_{n+1} dado las nuevas covariables X_{n+1}, que sigue la ley de probabilidad P, podemos construir un intervalo de predicción siguiendo la metdología mencioanda a continuación.

2.2 Método *naive* de construcción de intervalos

Usando el resultado previo de la sección anterior y en el contexto de regresión planteado en la Sección Capítulo 1, un método sencillo para contruir un intervalo predictivo para Y_{n+1} ante el valor X_{n+1} es:

$$C_{naive}(X_{n+1}) = \left[\hat{\mu}(X_{n+1}) - \hat{F}_n^{-1}(1-\alpha), \hat{\mu}(X_{n+1}) + \hat{F}_n^{-1}(1-\alpha)\right] \tag{2.3}$$

donde $\hat{\mu}$ es un estimador de la función de regresión, $\hat{F}_n$ la distribución empírica de los residuos dentro de la muestra $|Y_i - \hat{\mu}(X_i)|$, $i = 1, \ldots, n$ y $F_n^{-1}(1-\alpha)$ el cuantil $1-\alpha$ de $\hat{F}_n$.

Este método es aproximadamente válido para muestras grandes, bajo la condición de que $\hat{\mu}$ sea lo suficientemente preciso, es decir, que $F_n^{-1}(1-\alpha)$ esté cerca del cuantil $1-\alpha$ de $|Y_i - \mu(X_i)|$. Para que esto se cumpla en general es necesario el cumplimiento de condiciones de regularidad tanto para la distribución F de los datos y para $\hat{\mu}$, como que el modelo esté correctamente especificado.

Un problema de este método es que los intervalos pueden presentar una considerable subcobertura, dado que se están empleando los residuos dentro de la muestra. Para subsanar esto, en Lei et al. (2016) se plantea la metodología de los intervalos de predicción conformales.

2.3 Intervalos de predicción conformales

Para cada valor $y \in \mathbb{R}$ se construye un estimador de regresión aumentado $\hat{\mu}_y$, el cual se estima en el conjunto de datos aumentado $Z_1, \ldots Z_n, (X_{n+1}, y)$. Luego, se define:

$$R_{y,i} = |Y_i - \hat{\mu}_y(X_i)|, \;\; i = 1, \ldots, n \tag{2.4}$$

$$R_{y,n+1} = |y - \hat{\mu}_y(X_{n+1})| \tag{2.5}$$

Con el rango de $R_{y,n+1}$ entre los demás residuos de la muestra $R_{y,1}, \ldots, R_{y,n}$ se calcula:

$$\pi(y) = \frac{1}{n+1}\sum_{i=1}^{n+1} \mathbb{I}\{R_{y,i} \leq R_{y,n+1}\} = \frac{1}{n+1} + \frac{1}{n+1}\sum_{i=1}^{n} \mathbb{I}\{R_{y,i} \leq R_{y,n+1}\} \tag{2.6}$$

que es la proporción de los puntos de la muestra aumentada cuyos residuos dentro de la muestra son más pequeños que el residuo $R_{y,n+1}$. Como los datos son i.i.d. y suponiendo la simetría de $\hat{\mu}$, se puede apreciar que el estadístico $\pi(Y_{n+1})$ se distribuye uniforme en $1/(n+1), 2/(n+1), \ldots, 1$, lo cual implica:

$$P((n+1)\pi(Y_{n+1}) \leq \lceil (1-\alpha)(n+1) \rceil) \geq 1-\alpha \tag{2.7}$$

Esta expresión se puede interpretar como que $1 - \pi(Y_{n+1})$ da un valor-p válido conservador para la prueba de hipótesis donde $H_0) Y_{n+1} = y$.

Aplicando dicha prueba sobre todos los posibles valores de $y \in \mathbb{R}$, la ecuación Ecuación 2.7 lleva al intervalo de predicción conformal evaluado en X_{n+1}:

$$C_{conf}(X_{n+1}) = [y \in \mathbb{R} : (n+1)\pi(Y_{n+1}) \leq \lceil (1-\alpha)(n+1) \rceil] \tag{2.8}$$

Cada vez que se quiere obtener un intervalo de predicción en un nuevo conjunto de covariables se tienen que recalcular los pasos Ecuación 2.4, Ecuación 2.5, Ecuación 2.6 y Ecuación 2.8. En la práctica, se restringen los valores de y a una grilla discreta.

El procedimiento para obtener el intervalo se puede resumir en el Algoritmo 1.

Algoritmo 1: Intervalo de predicción conformal

Entrada: Datos (X_i, Y_i), $i = 1, \dots, n$, nivel de no cobertura $\alpha \in (0,1)$, algoritmo de regresión $\mathcal{A}$, puntos $\mathcal{X}_{nuevo}$ en los que construir intervalos de predicción y valores $\mathcal{Y}_{prueba} = \{y_1, y_2, \dots\}$ para comparar con la predicción.

Salida: Intervalos de predicción, en cada elemento de $\mathcal{X}_{nuevo}$

for $x \in \mathcal{X}_{nuevo}$ **do**
  for $y \in \mathcal{Y}_{prueba}$ **do**
    $\hat{\mu}_y = \mathcal{A}(\{(X_1, Y_1), \dots, (X_n, Y_n), (x, y)\})$
    $R_{y_i} = |Y_i - \hat{\mu}_y(X_i)|,\ i = 1, \dots, n$ y $R_{y,n+1} = |y - \hat{\mu}_y(x)|$
    $\pi(y) = (1 + \sum_{i=1}^{n} \mathbb{I}\{R_{y,i} \leq R_{y,n+1}\})/(n+1)$
  $C_{conf}(x) = [y \in \mathbb{R} : (n+1)\pi(Y_{n+1}) \leq \lceil (1-\alpha)(n+1) \rceil]$
Se devuelve $C_{conf}(x)$ para cada $X \in \mathcal{X}_{nuevo}$.

2.3.1 Teorema

El intervalo Ecuación 2.8 tiene cobertura válida para muestras finitas por construcción y a su vez no presenta sobrecobertura. Esto se puede expresar mediante las expresiones Ecuación 2.9 y Ecuación 2.10, respectivamente:

Sea (X_i, Y_i), $i = 1, \dots, n$ v.a. i.i.d, entonces para la nueva observación i.i.d. (X_{n+1}, Y_{n+1}):

$$P(Y_{n+1} \in C_{conf}(X_{n+1})) \geq 1 - \alpha \tag{2.9}$$

Adicionalmente, si se hace el supuesto que para todo $y \in \mathbb{R}$ los residuos dentro de la muestra $R_{y,i} = |Y_i - \hat{\mu}_y(X_i)|$, $i = 1, \dots, n$ tienen una distribución conjunta continua se cumple que:

$$P(Y_{n+1} \in C_{conf}(X_{n+1})) \leq 1 - \alpha + \frac{1}{n+1} \tag{2.10}$$

Observación. Nótese que las probabilidades aquí, al tomarse sobre la muestra aumentada i.i.d. implican cobertura promedio (o marginal). Esto no es lo mismo que la cobertura condicional $P(Y_{n+1} \in C_{conf}(x)|X_{n+1} = x) \geq 1 - \alpha \ \forall \ x \in \mathbb{R}^d$. Esta última es una propiedad más fuerte y no puede lograrse con intervalos predictivos de amplitud finita sin que el modelo y el estimador cumplan condiciones de regularidad y consistencia.

Observación. Si se mejora el estimador $\hat{\mu}$, en general el intervalo de predicción conformal decrece en tamaño. Esto se da debido a que un $\hat{\mu}$ más preciso lleva a residuos más pequeños y los intervalos conformales están definidos por los cuantiles de la distribución aumentada de los residuos.

2.4 Intervalos de predicción conformales con muestras separadas

Un problema práctico de los intervalos de inferencia conformal de la sección anterior es que tienen mucho costo computacional. Para poder concluir si $y \in C_{conf}(X_{n+1})$, para cualquier X_{n+1} y y, se tiene que reestimar el modelo en la muestra aumentada que incluye el nuevo punto X_{n+1} y recalcular y reordenar los nuevos residuos obtenidos.

Para enfrentar esta problemática se puede hacer uso de una metodología denominada por Lei et al. (2016) como predicción conformal separada (*split conformal prediction*). Su costo computacional es menor (es el del paso de estimación únicamente) y tiene menos requerimientos de memoria (solo hay que guardar las variables seleccionadas cuando se evalúa el ajuste en los nuevos puntos X_i, $i \in \mathcal{I}_2$). Se presenta en el Algoritmo 2.

Algoritmo 2: Intervalos de predicción conformales con muestras separadas

Entrada: Datos (X_i, Y_i), $i = 1, \ldots, n$, nivel de no cobertura $\alpha \in (0, 1)$, algoritmo de regresión $\mathcal{A}$.

Salida: Intervalos de predicción, sobre $x \in \mathbb{R}^d$

1 Se separa la muestra al azar en dos subconjuntos de igual tamaño $\mathcal{I}_1$ e $\mathcal{I}_2$.
2 $\hat{\mu}_y = \mathcal{A}(\{(X_i, Y_i) : i \in \mathcal{I}_1\})$
3 $R_i = |Y_i - \hat{\mu}_y(X_i)|$, $i \in \mathcal{I}_2$
4 $d =$ el k-ésimo valor más pequeño en $\{R_i : i \in \mathcal{I}_2\}$, donde $k = \lceil (n/2 + 1)(1 - \alpha) \rceil$
5 Se devuelve $C_{split}(x) = [\hat{\mu} - d, \hat{\mu} + d]$ para todo $x \in \mathbb{R}^d$.

2.4.1 Teorema

Sea (X_i, Y_i), $i = 1, \ldots, n$ v.a. i.i.d, entonces para la nueva observación i.i.d. (X_{n+1}, Y_{n+1}):

$$P(Y_{n+1} \in C_{splt}(X_{n+1})) \geq 1 - \alpha \tag{2.11}$$

Adicionalmente, si se hace el supuesto que los residuos R_i, $i \in \mathcal{I}_2$ tienen una distribución conjunta continua se cumple que:

$$P(Y_{n+1} \in C_{split}(X_{n+1})) \leq 1 - \alpha + \frac{2}{n+2} \tag{2.12}$$

2.4.2 Teorema

Los intervalos de predicción conformales con muestras separadas dan una garantía aproximada de cobertura dentro de la muestra. Esto se puede expresar como que existe una constante $c > 0$ tal que, para cualquier $\epsilon > 0$:

$$P\left(\frac{2}{n}\sum_{i\in\mathcal{I}_2} \mathbb{I}\{Y_I \in C_{split}(X_i) - (1-\alpha) \geq \epsilon\}\right) \leq 2\exp\left(-cn^2(\epsilon - 4/n)^2\right) \tag{2.13}$$

Esto implica cobertura dentro de la muestra para la muestra $\mathcal{I}_2$, revirtiendo los roles de $\mathcal{I}_1$ e $\mathcal{I}_2$ se puede extender para toda la muestra.

Observación. También se puede aplicar este método con una separación no balanceada de la muestra, con $|\mathcal{I}_1| = \rho n$ e $|\mathcal{I}_2| = (1-\rho)n$, para $\rho \in (0,1)$. Esto puede ser útil en situaciones donde el procedimiento de regresión es complejo y puede resultar beneficioso elegir $\rho > 0,5$, para que $\hat{\mu}$ sea más preciso.

2.5 Intervalos conformales con múltiples separaciones de la muestra

Al considerar diferentes divisiones estamos introduciendo una fuente **adicional** de aleatoriedad a nuestra estimación, por tanto, el método de dividir la muestra reduce el costo computacional pero introduce ruido provocando una mayor incertidumbre en nuestras predicciones.

Una manera de corregirlo es combinar las diferentes inferencias realizadas en N particiones independientes, construyendo así, $C_{split,1}$,$\cdots$,$C_{split,N}$, en donde cada intervalo es construido a un nivel de significación $\alpha^* = 1 - \alpha/N$ y su relación esta dada de la siguiente manera:

$$C^N_{split}(x) = \cap_{j=1}^N C_{split,j}(x) \quad x \in R^d$$

Como se puede apreciar, al utilizar este procedimiento hay un precio que debemos de pagar, los intervalos se vulven mas anchos a medida que N crece, esto puede verse de la siguiente manera:

Como sabemos, la cobertura marginal de nuestra predicción tiene una cobertura de al menos $1-\alpha$ por construcción, además con el algoritmo 2 podemos acotar aún mas esta probabilidad:

$$1-\alpha \leq P(Y_{n+1} \in C_{split}\left(X_{n+1}\right)) \leq 1-\alpha+\frac{2}{n+2} \leq 1 \tag{2.14}$$

En nuestro caso, cada partición esta construida con un nivel de significación $\alpha^* = \alpha/N$, por tanto remplazando esta identidad en 2.14, obtenemos la siguiente expresión

$$1-\frac{\alpha}{N} \leq P(Y_{n+1} \in C_{split}\left(X_{n+1}\right)) \leq 1-\frac{\alpha}{N}+\frac{2}{n+2} \leq 1$$

Es por esto que si aumentamos N el término α^* se vuelve aún mas chico y debido a que estamos frente a una medida de probabilidad esta se acerca cada vez más a uno, por tanto, nuestros intervalos se vuelven aún mas estrechos.

2.6 Intervalos predictivos mediante Jackknife

Esta metodología emplea los cuantiles de los residuos de validación cruzada dejando una observación fuera (*leave-one-out*) para calcular los intervalos de predicción.

Algoritmo 3: Intervalo de predicción conformal mediante Jackknife.

Entrada: Datos (X_i, Y_i), $i = 1, \ldots, n$, nivel de no cobertura $\alpha \in (0, 1)$, algoritmo de regresión $\mathcal{A}$.

Salida: Intervalos de predicción sobre $x \in \mathbb{R}^d$.

for $i \in \{1, \ldots, n\}$ **do**
  $\hat{\mu}^{(-i)} = \mathcal{A}(\{(X_l, Y_l) : l \neq i\})$
  $R_i = |Y_i - \hat{\mu}^{(-i)}(X_i)|$
$d =$ el k-ésimo valor más pequeño en $\{R_i : i \in \{1, \ldots, n\}\}$, con $k = \lceil n(1-\alpha) \rceil$
Se devuelve $C_{jack}(x) = [\hat{\mu}(x) - d, \hat{\mu}(x) + d]$ para todo $x \in \mathbb{R}^d$

Tiene la ventaja que emplea más de la muestra que se aparta para entrenar cuando se calculan los residuos, lo cual frecuentemente lleva a intervalos de menor amplitud. Como desventaja, los intervalos que se obtienen no garantizan cobertura válida fuera de la muestra cuando se trabaja con muestras finitas e incluso asintóticamente la cobertura depende de condiciones del estimador.

3 Aplicación

En esta sección se presentan aplicaciones de las metoodlogías de inferencia conformal mediante el software *R* (R Core Team, 2022). Los ejemplos de programación manual de los intervalos están basados en los presentados en Samii (2019).

En primera instancia, simulamos un conjunto de datos para realizar las estimaciones. En este caso es una mixtura de normales de media 0 y varianza 1.

```
set.seed(12345)
n <- 1000

U <- rnorm(n)
X <- rnorm(n)

Y <- X + U
regData <- data.frame(X,Y)
```

Por construcción, los datos tienen una relación lineal, lo cual se puede ver visualmente en la Figura 3.1

Luego, realizamos estimaciones de la función de regresión y guardamos los residuos. En primera instancia una estimación paramétrica por mínimos cuadrados:

```
fitlm <- lm(Y~X, data=regData)

eVec_lm <- abs(fitlm$residuals)
```

Y también una estimación no paramétrica usando el estimador Naradaya-Watson:

```
fitnw <- ksmooth(
  X,
  Y,
  kernel = "normal",
  bandwidth = 0.2
)
```

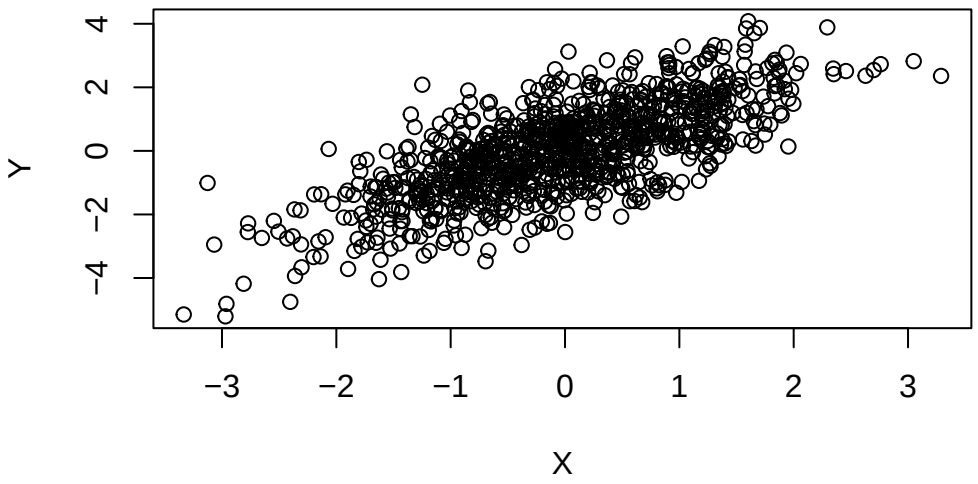

Figura 3.1: Gráfico de dispersión de los datos de ejemplo.

```
eVec_nw <- abs(fitnw$y-Y)
```

Los histogramas de los residuos en valor absoluto para ambas estimaciones se pueden observar en la Figura 3.2. Nótese como la estimación no paramétrica es más dispersa en este caso. Dado que los métodos de inferencia conformal dependen fuertemente del estimador, los intervalos probablemente se vean afectados por esta diferencia.

3.1 Método naive

```
Xnew <- seq(-4,4,0.25)

# Estimacion paramétrica
muHat_lm <- predict(
  fitlm,
  newdata=data.frame(X=Xnew)
)

# Intervalos al 5% de confianza para lm
```

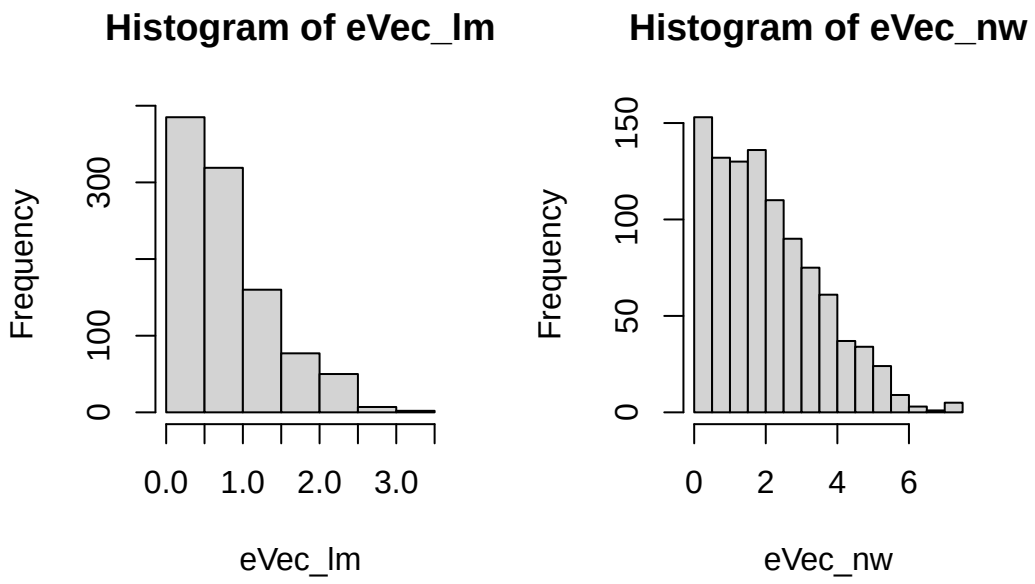

Figura 3.2: Histogramas de los residuos en valores absolutos

```
C.X_lm_lwr <- c(
  muHat_lm - quantile(eVec_lm, .975)
)

C.X_lm_uppr <- c(
  muHat_lm + quantile(eVec_lm, .975)
)

# Estimacion no paramétrica
muHat_nw <-  ksmooth(
  X,
  Y,
  kernel = "normal",
  bandwidth = 0.2,
  x.points = Xnew
)

# Intervalos al 5% de confianza para nw
C.X_nw_lwr <- c(
  muHat_nw$y - quantile(eVec_nw, .975)
```

```
)

C.X_nw_uppr <- c(
  muHat_nw$y + quantile(eVec_nw, .975)
)

resultados_conf_naive <- data.frame(
  Xnew,
  muHat_lm,
  muHat_nw$y,
  C.X_lm_lwr,
  C.X_lm_uppr,
  C.X_nw_lwr,
  C.X_nw_uppr
)
```

Luego, gráficamos ambas estimaciones con sus respectivos intervalos Figura 3.3. Se puede apreciar como los intervalos dependen de la calidad del estimador, en este caso la estimación no paramétrica no fue optimizada y como resultado el intervalo es mucho más amplio que el de la regresión lineal.

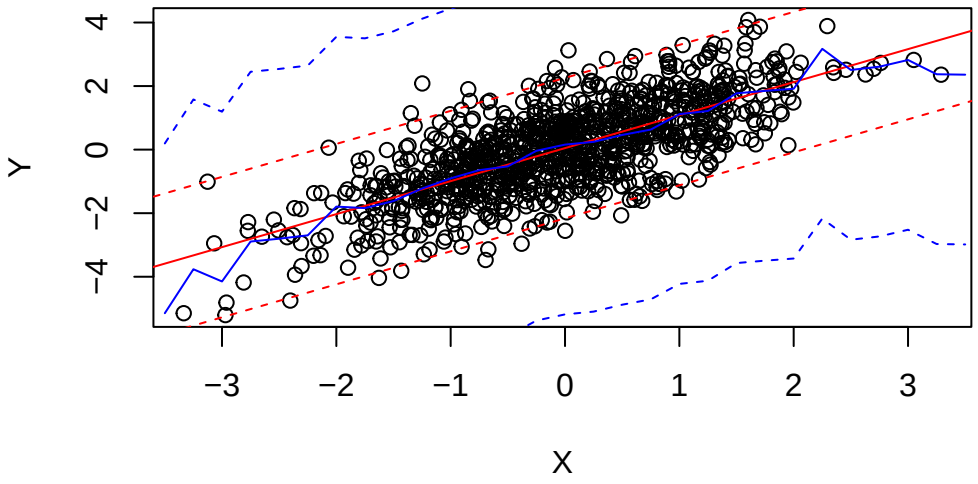

Figura 3.3: Gráfico de dispersión de datos de prueba, con estimaciones e intervalos mediante modelo lineal MCO (rojo) y Naradaya-Watson (azul).

3.2 Método conformal simple

```
#nEval <- 200

yCand <- seq(
  from = min(Y)-1,
  to = max(Y)+1,
  by = 0.1
)

confPredict <- function(y, Xin){

  nData <- nrow(regData)
  regData.a <- rbind(regData,c(Xin, y))

  fitlm.a <- lm(Y~X, data=regData.a)

  resOut <- abs(fitlm.a$residuals)
  resOut_new <- resOut[length(resOut)]

  pi.y <- mean(
    apply(
      as.matrix(resOut),
      1,
      function(x){x<=resOut_new}
    )
  )

  testResult <- pi.y*(nData+1) <= ceiling(.975*(nData+1))

  return(testResult)
}



Cxa <- matrix(
  0,
  nrow = length(Xnew),
  ncol=2
  )
```

```r
for(i in 1:length(Xnew)){
  Cxa[i,] <- range(
    yCand[sapply(yCand, confPredict, Xin=Xnew[i])]
  )
}



resultados_conf_simple <- data.frame(
  Xnew,
  muHat_lm, # La estimacion puntual es la misma
  Cxa_lm_lwr = Cxa[,1],
  Cxa_lm_uppr = Cxa[,2],
  C.X_lm_lwr,
  C.X_lm_uppr
)
```

En la Figura 3.4 se presenta tanto los intervalos con el método conformal simple como con el naive. Se puede notar que los intervalos son casi idénticos. La rugosidad se debe a que estamos trabajando en una grilla y no el recorrido continuo.

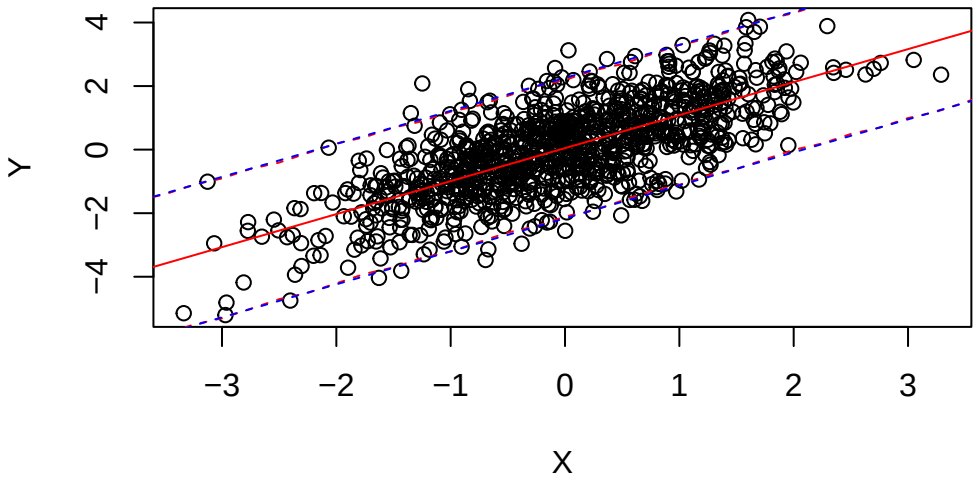

Figura 3.4: Gráfico de dispersión de datos de prueba, con estimaciones mediante regresión lineal e intervalos conformales simples (rojo) y naive (azul).

3.3 Split

El método de intervalos conformales con separaciones de la muestra (*splits*) es más eficiente computacionalmente.

```
splitConfPredict <- function(Xin){
  nData <- nrow(regData)
  regData$index <- 1:nData
  regData$split <- 1
  regData$split[sample(regData$index, floor(nrow(regData)/2), replace=F)] <- 2
  fitlm.spl <- lm(Y~X, data=subset(regData, split==1))
  resOut <- abs(
    subset(regData, split==2)$Y -
     predict(fitlm.spl, newdata=subset(regData, 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)
}

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

resultados_conf_split <- data.frame(
  Xnew,
  muHat_lm = muHat_lm, # La estimacion puntual es la misma
  Csplit_lwr = Csplit[,1],
  Csplit_uppr = Csplit[,3],
  Cxa_lm_lwr = Cxa[,1],
  Cxa_lm_uppr = Cxa[,2]
)
```

En la Figura 3.5 se puede observar los intervalos con el método con *split*. Estos intervalos parecen ser un tanto más amplios que los calculados mediante el método simple.

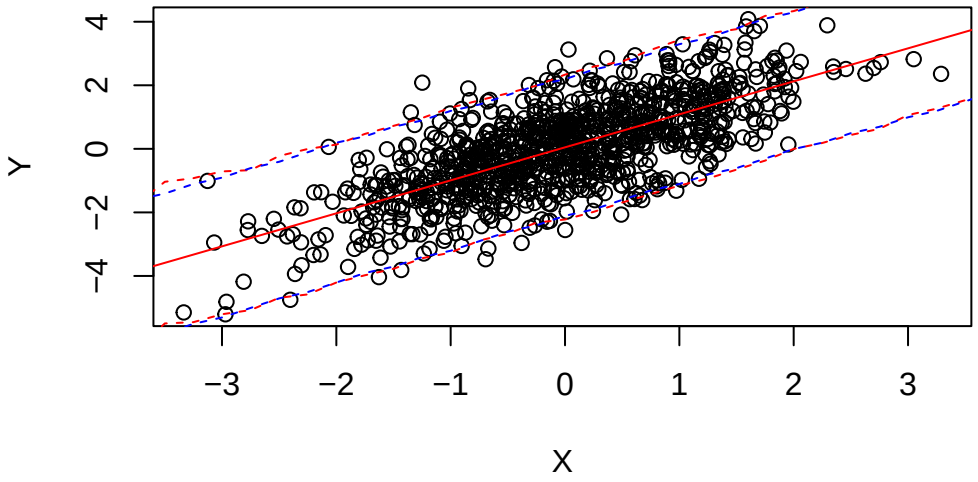

Figura 3.5: Gráfico de dispersión de datos de prueba, con estimaciones mediante regresión lineal e intervalos conformales split (rojo) y simples (azul).

4 Conclusión

Si bien en la práctica los intervalos de predicción pueden llegar a ser confusos para personas sin conocimiento de estadística, es importante destacar su utilidad en la toma de decisiones, cualquier metodología que aborde sobre la mejora de estos es de gran relevancia; como la metodología expuesta en este articulo.

Con este articulo se ha revisado cómo implementar intervalos de predicción conformales para cualquier modelo de regresión. Estos alogritmos requieren únicamente asumir que los datos tienen independencia entre sí, pero ninguna forma funcional sobre los regresores, matriz de diseño o distribución de probabilidad para los residuos. Esto último puede llegar a ser una gran ventaja para modelos en donde no es tan claro la construcción de intervalos de predicción, que usualmente son aproximados con boostrap, como pueden ser los métodos de ensamble o en aquellos modelos donde se hacen supuestos fuertes sobre la distribución de los datos basados en resultados asintóticos.

El considerar este tipo de metodologías nos permite reducir la incertidumbre respecto a un intervalo de predicción clásico como pudimos ver en el Sección 2.3.1. Es importante mencionar que se cuenta con una implementación por parte de los autores de Lei et al. (2016) en la librería **conformalInferece** (Tibshirani et al., 2019) -ver Sección A.1 - de R con modelos de regresión ampliamente utilizados en la práctica, como son los modelos de regresión lineal, splines, glm, random forest, etc.

Cabe destacar que la inferencia conformal es aplicada frecuentemente en la actualidad a modelos de machine learning. Estas metdologías pueden complementar los métodos basados únicamente en boostrap para estimación de varianzas para cualquier algortimo de regresión,ya que al agregar el supuesto de que el estimador es consistente se puede obtener un intervalo de predicción consistente, con cobertura condicional apropiada.

5 Bibliografía

Lei, J., G'Sell, M., Rinaldo, A., Tibshirani, R. J., & Wasserman, L. (2016). *Distribution-Free Predictive Inference For Regression*. arXiv. https://doi.org/10.48550/ARXIV.1604.04173

R Core Team. (2022). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing. https://www.R-project.org/

Samii, C. (2019). *Conformal Inference Tutorial*. https://cdsamii.github.io/cds-demos/conformal/conformal-tutorial.html

Tibshirani, R., Diquigiovanni, J., Fontana, M., & Vergottini, P. (2019). *conformalInference: Tools for conformal inference in regression*.

A Anexo

A.1 Paquete conformalInference

El paquete de *R conformalInference* (Tibshirani et al., 2019) ya tiene implementadas todas estas herramientas de inferencia conformal.

En concreto, la función `conformal.pred` permite realizar intervalos de predicción conformales para cualquier modelo de regresión, además de su versión menos costosa en términos computacionales, los intervalos con *splits*, mediante `conformal.split`.

```
names(formals(conformal.pred))
```

```
[1] "x"               "y"               "x0"              "train.fun"
[5] "predict.fun"     "alpha"           "w"               "mad.train.fun"
[9] "mad.predict.fun" "num.grid.pts"    "grid.factor"     "verbose"
```

Los argumentos de la función `conformal.pred` son los siguientes:

- x -> Vector de variables independientes (nxp)
- y -> Vector de variables dependientes n
- xo -> Matriz de variables independientes para predcir un nuevo y (n0xp)
- train.fun -> Función para mejorar la performance del modelo, varía en cada una de sus especificaciones.
- predcit.fun -> Función que se utilizará para predcir el nuevo valor de y
- alpha -> Valor de cobertura
- ...

La documentación completa se encuentra en el siguiente link: https://github.com/ryantibs/conformal/blob/master/conformalInference.pdf

Para cada argumento *.fun están implementados los siguientes modelos por defecto:

- **Modelo de regresión lineal** (Implementación propia de MCO usando descomposición de Cholesky) `lm.funs()`
- **Random Forest** (via randomForets) `rf.funs()`
- **Splines** (via splines) `smooth.splines.funs()`

- **Elastic Net** (via glmnet) `elastic.funs()`

Para el caso de `train.fun` y `predict.fun` se pueden implementar funciones propias, dependiendo del modelo su implementación puede variar.

A.2 Implementación de un modelo de regresión lineal

```
train.fun = function(x,y,out=NULL) {
  n = nrow(x); p = ncol(x)
  v = rep(1,p)

  if (intercept) {
    x = cbind(rep(1,n),x)
    v = c(0,v)
  }
  if (!is.null(out)) {
    chol.R = out$chol.R
  }
  else {
    chol.R = vector(mode="list",length=m)
    for (j in 1:m) {
      chol.R[[j]] = chol(crossprod(x) + lambda[j]*diag(v))
    }
  }
  beta = matrix(0,p+intercept,m)
  for (j in 1:m) {
    beta[,j] = chol.solve(chol.R[[j]], t(x) %*% y)
  }

  return(list(beta=beta,chol.R=chol.R))
}
```

```
predict.fun = function(out,newx) {
  if (intercept) {
    newx = cbind(rep(1,nrow(newx)),newx)
  }
  return(newx %*% out$beta)
}
```

A.3 Ejemplo para implementar Nadaraya–Watson

```r
train.fun_ks <- function(x,y,out=NULL) {

  kernel = "normal"
  bandwidth = 0.2

  out <- ksmooth(
    X,
    Y,
    kernel,
    bandwidth = bandwidth
  )

  list(
    x = out$x,
    y = out$y,
    kernel = kernel,
    bandwidth = bandwidth
  )

}

predict.fun_ks <- function(out, newx) {
  return(
    ksmooth(
      x = out$x,
      y = out$y,
      out$kernel,
      out$bandwidth,
      x.points = newx
    )$y
  )
}
```

Si bien en el paquete tiene implementado diferentes modelos, como es el ejemplo de la regresión lineal no paramétrica de Nadaraya–Watson, su resultado final no tiene asociado el método `predict` y devuelve una lista. Haciendo las modificaciones correspondientes, es posible utilizarlo en la función `conformal.pred`.

A.3.1 Ejemplo de predicción conformal para Nadaraya–Watson

Hagamos una prueba, intentado predecir el valor de Y para $X = -1.5$, como en el ejemplo anterior:

```
res <- conformal.pred(
  x = X,
  y = Y,
  x0 = Xnew,
  train.fun = train.fun_ks,
  predict.fun = predict.fun_ks,
  num.grid.pts = 100
)

res$pred
```

```
             [,1]
 [1,]          NA
 [2,]          NA
 [3,] -4.87630759
 [4,] -3.43797196
 [5,] -3.77131184
 [6,] -2.99890859
 [7,] -2.80025703
 [8,] -2.66018213
 [9,] -1.89625155
[10,] -1.85606811
[11,] -1.59033654
[12,] -1.20228221
[13,] -0.89280425
[14,] -0.63782155
[15,] -0.50224837
[16,] -0.06792671
[17,]  0.14472917
[18,]  0.24117116
[19,]  0.45286299
[20,]  0.68495717
[21,]  1.11046865
[22,]  1.25119791
[23,]  1.75292340
[24,]  1.89956944
[25,]  2.00778508
```

```
[26,]  2.95576648
[27,]  2.53958502
[28,]  2.60220691
[29,]  2.79494116
[30,]  2.46827256
[31,]  2.36625673
[32,]          NA
[33,]          NA
```

Ahora, probemos con la versión utilizando particiones:

```r
resplit <- conformal.pred.split(
  x = X,
  y = Y,
  x0 = Xnew,
  train.fun = train.fun_ks,
  predict.fun = predict.fun_ks
)

resplit$pred
```

```
            [,1]
 [1,]          NA
 [2,]          NA
 [3,] -4.87630759
 [4,] -3.43797196
 [5,] -3.77131184
 [6,] -2.99890859
 [7,] -2.80025703
 [8,] -2.66018213
 [9,] -1.89625155
[10,] -1.85606811
[11,] -1.59033654
[12,] -1.20228221
[13,] -0.89280425
[14,] -0.63782155
[15,] -0.50224837
[16,] -0.06792671
[17,]  0.14472917
[18,]  0.24117116
[19,]  0.45286299
[20,]  0.68495717
```

```
[21,]  1.11046865
[22,]  1.25119791
[23,]  1.75292340
[24,]  1.89956944
[25,]  2.00778508
[26,]  2.95576648
[27,]  2.53958502
[28,]  2.60220691
[29,]  2.79494116
[30,]  2.46827256
[31,]  2.36625673
[32,]          NA
[33,]          NA
```

Como sabemos, el resultado es el mismo. Elegir un método u otro impactará en la estimación de los invervalos de predicción, veamos cada uno de ellos:

```r
intervalo <- function(puntual,lo,uper) {
  glue::glue(
    "[{puntual} - {lo};{puntual} + {uper}]",
    lo = round(puntual - lo,2),
    uper = round(uper - puntual,2),
    puntual = round(puntual,2)
  )
}


# Invervalo con el método conformal

intervalo(
  res$pred,
  res$lo,
  res$up
)
```

```
[NA - NA;NA + NA]
[NA - NA;NA + NA]
[-4.88 - -4.42;-4.88 + 10.2]
[-3.44 - -2.98;-3.44 + 8.76]
[-3.77 - -3.31;-3.77 + 9.1]
[-3 - -2.54;-3 + 8.32]
[-2.8 - -2.34;-2.8 + 8.13]
```

```
[-2.66 - -2.2;-2.66 + 7.99]
[-1.9 - -1.44;-1.9 + 7.22]
[-1.86 - -1.4;-1.86 + 7.18]
[-1.59 - -1.13;-1.59 + 6.92]
[-1.2 - -0.74;-1.2 + 6.53]
[-0.89 - -0.43;-0.89 + 6.22]
[-0.64 - -0.18;-0.64 + 5.96]
[-0.5 - -0.04;-0.5 + 5.83]
[-0.07 - 0.39;-0.07 + 5.39]
[0.14 - 0.6;0.14 + 5.18]
[0.24 - 0.7;0.24 + 5.08]
[0.45 - 0.91;0.45 + 4.87]
[0.68 - 1.15;0.68 + 4.64]
[1.11 - 1.57;1.11 + 4.21]
[1.25 - 1.71;1.25 + 4.07]
[1.75 - 2.21;1.75 + 3.57]
[1.9 - 2.36;1.9 + 3.43]
[2.01 - 2.47;2.01 + 3.32]
[2.96 - 3.42;2.96 + 2.37]
[2.54 - 3;2.54 + 2.79]
[2.6 - 3.06;2.6 + 2.72]
[2.79 - 3.26;2.79 + 2.53]
[2.47 - 2.93;2.47 + 2.86]
[2.37 - 2.96;2.37 + 2.96]
[NA - NA;NA + NA]
[NA - NA;NA + NA]
```

```r
# Intervelo con el método de split


intervalo(
  resplit$pred,
  resplit$lo,
  resplit$up
)
```

```
[NA - NA;NA + NA]
[NA - NA;NA + NA]
[-4.88 - 2.92;-4.88 + 2.92]
[-3.44 - 2.92;-3.44 + 2.92]
[-3.77 - 2.92;-3.77 + 2.92]
[-3 - 2.92;-3 + 2.92]
```

```
[-2.8 - 2.92;-2.8 + 2.92]
[-2.66 - 2.92;-2.66 + 2.92]
[-1.9 - 2.92;-1.9 + 2.92]
[-1.86 - 2.92;-1.86 + 2.92]
[-1.59 - 2.92;-1.59 + 2.92]
[-1.2 - 2.92;-1.2 + 2.92]
[-0.89 - 2.92;-0.89 + 2.92]
[-0.64 - 2.92;-0.64 + 2.92]
[-0.5 - 2.92;-0.5 + 2.92]
[-0.07 - 2.92;-0.07 + 2.92]
[0.14 - 2.92;0.14 + 2.92]
[0.24 - 2.92;0.24 + 2.92]
[0.45 - 2.92;0.45 + 2.92]
[0.68 - 2.92;0.68 + 2.92]
[1.11 - 2.92;1.11 + 2.92]
[1.25 - 2.92;1.25 + 2.92]
[1.75 - 2.92;1.75 + 2.92]
[1.9 - 2.92;1.9 + 2.92]
[2.01 - 2.92;2.01 + 2.92]
[2.96 - 2.92;2.96 + 2.92]
[2.54 - 2.92;2.54 + 2.92]
[2.6 - 2.92;2.6 + 2.92]
[2.79 - 2.92;2.79 + 2.92]
[2.47 - 2.92;2.47 + 2.92]
[2.37 - 2.92;2.37 + 2.92]
[NA - NA;NA + NA]
[NA - NA;NA + NA]
```

Como podemos ver el método de split es más amplio, en concreto existe una mayor incertidumbre a la izquierda del valor predicho. Para comparar la eficiencia de los intervalos, comparemos su apertura relativa, respecto al método conformal entero:

```
(resplit$lo-resplit$up) / (res$lo-res$up)
```

```
          [,1]
[1,]        NA
[2,]        NA
[3,] 1.0082432
[4,] 1.0082432
[5,] 1.0082432
[6,] 1.0082432
[7,] 1.0082432
```

```
 [8,] 1.0082432
 [9,] 1.0082432
[10,] 1.0082432
[11,] 1.0082432
[12,] 1.0082432
[13,] 1.0082432
[14,] 1.0082432
[15,] 1.0082432
[16,] 1.0082432
[17,] 1.0082432
[18,] 1.0082432
[19,] 1.0082432
[20,] 1.0082432
[21,] 1.0082432
[22,] 1.0082432
[23,] 1.0082432
[24,] 1.0082432
[25,] 1.0082432
[26,] 1.0082432
[27,] 1.0082432
[28,] 1.0082432
[29,] 1.0082432
[30,] 1.0082432
[31,] 0.9858378
[32,]        NA
[33,]        NA
```

Podemos ver que el método de splitting es más amplio, pero solo en poco más de un 2%.

A.4 Ejemplo regresión múltiple

Tomando como ejemplo el dataset `state.x77` de R, vamos a intentar predecir la variable `life_exp` utilizando como variables explicativas `income`, `hs_grad`, `frost` y `area`.

Vamos a entrenar el modelo con todas las observaciones, a excepción de una, que será la que queremos predecir, en concreto, la ciudad de Colorado.

```
# Semilla

set.seed(1234)

# Para arreglar los nombres

normalize_names <- function(names) {
  upper = tolower(names)
  newNames = gsub(" ", "_", upper)
  return(newNames)
}

df <- as.data.frame(state.x77)

names(df) <- normalize_names(names(df))

# Para elegir la observación a predecir

test <- runif(1,1,nrow(df))

# Vector de variables independientes


Y = df[-test,c('life_exp')]


# Matriz de diseño

X = df[-test,c('income','hs_grad','frost','area')]

row.names(X) <- NULL
X = data.matrix(X, rownames.force = TRUE)
```

```r
# Matriz de diseño en los nuevos puntos

x0 = data.matrix(
  df[test,c('income','hs_grad','frost','area')],
  rownames.force = TRUE
)

# No funciona con lm.funs, NW usamos RF!

engine_rf = rf.funs(
  ntree = 500,
  varfrac = 0.333
)

train.fun = engine_rf$train.fun
predict.fun = engine_rf$predict.fun

res <- conformal.pred(
  x = X,
  y = Y,
  x0 = x0,
  train.fun = train.fun,
  predict.fun = predict.fun
)


intervalo(
  res$pred,
  res$lo,
  res$up
)
```

```
[71.01 - 1.32;71.01 + 0.54]
```

```r
glue::glue(
  "La expecativa de vida de la {ciudad} es {df[test,'life_exp']}",
  ciudad = row.names(df[test,])
)
```

```
La expecativa de vida de la Colorado es 72.06
```

Para la realización de este ejemplo, se intentó realizar la predicción con el modelo de regresión lineal tanto en su versión paramétrica como su versión de núcleo,el resultado esta repleto de valores NA, por lo que se optó por probar diferentes modelos de regresión no paramétrica, en concreto, Random Forest, el cual nos dio un resultado satisfactorio.