

Notas Curso de Estadística

Maikol Solís

Actualizado el 01 julio, 2020

Índice general

1. Introducción	9
2. Estimación de densidades	11
2.1. Histograma	11
2.1.1. Construcción Estadística	11
2.1.2. Construcción probabilistica	12
2.1.3. Propiedades estadísticas	13
2.1.4. Sesgo	13
2.1.5. Varianza	14
2.1.6. Error cuadrático medio	14
2.1.7. Error cuadrático medio integrado	16
2.1.8. Ancho de banda óptimo para el histograma	17
2.2. Estimación No-paramétrica de densidad	20
2.2.1. Primera construcción	20
2.2.2. Otra construcción	21
2.3. Propiedades Estadísticas	24
2.3.1. Varianza	25
2.3.2. Sesgo	26
2.3.3. Error cuadrático medio y Error cuadrático medio integrado	28

2.3.4.	Ancho de banda óptimo	28
2.4.	Escogiendo el ancho de banda	29
2.4.1.	Referencia normal	30
2.4.2.	Validación Cruzada	31
2.4.3.	Intervalos de confianza para estimadores de densidad no paramétricos	33
2.5.	Laboratorio	34
2.5.1.	Efecto de distintos Kernels en la estimación	34
2.5.2.	Efecto del ancho de banda en la estimación	37
2.5.3.	Ancho de banda óptimo	41
2.5.4.	Validación cruzada	45
2.5.5.	Temas adicionales	46
2.6.	Ejercicios	51
3.	Jackknife y Bootstrap	53
3.1.	Caso concreto	54
3.2.	Jackknife	55
3.3.	Bootstrap	61
3.4.	Ejercicios	69
4.	Estimación de densidades con Bayes	71
4.1.	Introducción a la estimación Bayesiana	71
4.1.1.	Preliminares	71
4.1.2.	Ejemplo sencillo	73
4.1.3.	Datos reales	75
4.2.	Previa de histograma	79
4.3.	Métodos Monte Carlo	82
4.4.	Una moneda	82

ÍNDICE GENERAL	5
4.4.1. Ejemplo del viajero	82
4.4.2. Cadenas de Markov	85
4.4.3. El algoritmo de Metropolis-Hasting	86
4.4.4. ¿Por qué el algoritmo de Metropolis Hasting funciona?	88
4.4.5. Extensión al caso del viajero	88
4.5. Dos monedas	93
4.5.1. Muestreo de Gibbs	99
4.6. Uso de JAGS	105
4.7. Uso de STAN	113
4.8. Ejercicios	117
5. Métodos lineares de regresión	119
5.1. Introducción	119
5.2. Regresión lineal	120
5.2.1. Forma matricial	121
5.2.2. Laboratorio	124
5.3. Propiedades estadísticas	130
5.3.1. Prueba t	132
5.3.2. Prueba F	132
5.3.3. Laboratorio	133
5.4. Medida de bondad de ajuste	135
5.4.1. Laboratorio	136
5.4.1.1. R^2	137
5.4.1.2. R^2 ajustado	137
5.4.1.3. <code>summary</code>	138
5.5. Predicción	139
5.5.1. Laboratorio	140

5.5.1.1. Ajuste de la regresión sin intervalos de confianza	141
5.5.1.2. Ajuste de la regresión con intervalos de confianza	141
5.5.1.3. Ajuste de la regresión con intervalos de confianza y predicción	142
5.6. Interacciones	146
5.6.1. Laboratorio	147
5.7. Hipótesis en regresión lineal	151
5.7.1. Hipotésis	152
5.7.2. Chequeos básicos de las hipótesis de regresión lineal	153
5.7.2.1. Independencia lineal, Errores con esperanza nula, Homocedasticidad	153
5.7.2.2. Independencia de los errores	156
5.7.2.3. Normalidad de los errores	160
5.7.2.4. Multicolinealidad	163
5.7.3. Otros chequeos importantes	165
5.7.3.1. Puntos extremos	165
5.7.3.2. Puntos de apalancamiento (leverage)	169
Distancia de Cook.	169
5.8. Ejercicios	181
6. Regresión Logística	183
6.1. Razón de proporción	186
6.2. Máxima verosimilitud	187
6.2.1. Residuos	188
6.3. Diagnósticos del modelo	189
6.3.1. Supuesto de linealidad	189
6.3.2. Valor de gran influencia	191
6.3.3. Multicolinealidad	195

6.4.	Predicción y poder de clasificación	195
6.4.1.	Curva ROC	199
6.5.	Ejercicios	202
7.	Métodos de selección de variables	205
7.1.	Selección del mejor subconjunto.	205
7.1.1.	Error de prueba	207
7.1.1.1.	Validación cruzada	207
	Validación cruzada «Leave-One-Out» (LOOCV)	207
	Validación cruzada k -veces	208
7.1.1.2.	Validación cruzada para clasificación	208
7.1.2.	Otras medidas de error de entrenamiento	209
7.1.3.	Selección de modelos hacia adelante (Forward Stepwise Selection)	210
7.1.4.	Selección de modelos hacia atrás (Backward Stepwise Selection)	211
7.2.	Métodos de regularización	211
7.2.1.	Regresión Ridge	211
7.2.1.1.	Estimación clásica por mínimos cuadrados . . .	212
7.2.1.2.	Ventajas	212
7.2.2.	Regresión Lasso	213
7.2.3.	Explicación gráfica	213
7.3.	Laboratorio	214
7.3.1.	Cross-Validation	214
7.3.1.1.	Leave-one-out Cross Validation (LOOCV) . .	214
7.3.1.2.	K-Fold Cross Validation	216
7.3.2.	Selección de variables	217
7.3.2.1.	Análisis exploratorio	217

7.3.2.2. Selección del mejor subconjunto	219
7.3.2.3. Regresión forward y backward	229
7.3.2.4. Regresión Ridge	234
7.3.3. Regresión Lasso	243
7.4. Ejercicios	245
8. Análisis en componentes principales	247
8.1. Representación gráfica	248
8.2. Primer componente principal	248
8.3. Segunda componente principal	250
8.4. Circulo de correlaciones	251
8.5. Volvamos a nuestro ejemplo	251
8.6. ¿Cuántos componentes usar?	252
8.7. Laboratorio	253
8.8. Ejercicios	256

Capítulo 1

Introducción

Capítulo 2

Estimación de densidades

2.1. Histograma

El histograma es una de las estructuras básicas en estadística. Básicamente con este objeto se puede visualizar la distribución de los datos sin tener conocimiento previo de los mismos.

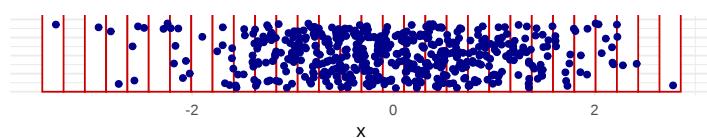
2.1.1. Construcción Estadística

Suponga que X_1, X_2, \dots, X_n proviene de una distribución desconocida.

- Seleccione un origen x_0 y divida la linea real en *segmentos*.

$$B_j = [x_0 + (j - 1)h, x_0 + jh) \quad j \in \mathbb{Z}$$

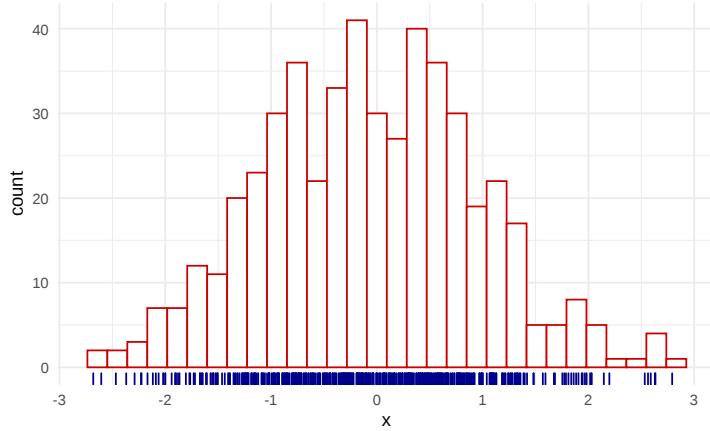
- Cuente cuántas observaciones caen en cada segmento. n_j .



- Cuente la frecuencia por el tamaño de muestra n y el ancho de banda h .

$$f_j = \frac{n_j}{nh}$$

- Dibuje el histograma.



Formalmente el histograma es el

$$\hat{f}_h(x) = \frac{1}{nh} \sum_{i=1}^n \sum_j I(X_i \in B_j) I(x \in B_j),$$

donde I es la indicadora.

2.1.2. Construcción probabilistica

Denote $m_j = jh - h/2$ el centro del segmento,

$$\begin{aligned} \mathbb{P}\left(X \in \left[m_j - \frac{h}{2}, m_j + \frac{h}{2}\right]\right) &= \int_{m_j - \frac{h}{2}}^{m_j + \frac{h}{2}} f(u) du \\ &\approx f(m_j)h \end{aligned}$$

Esto se puede aproximar como

$$\mathbb{P} \left(X \in \left[m_j - \frac{h}{2}, m_j + \frac{h}{2} \right) \right) \approx \frac{1}{n} \# \left\{ X \in \left[m_j - \frac{h}{2}, m_j + \frac{h}{2} \right) \right\}$$

Acomodando un poco la expresión

$$\hat{f}_h(m_j) = \frac{1}{nh} \# \left\{ X \in \left[m_j - \frac{h}{2}, m_j + \frac{h}{2} \right) \right\}$$

2.1.3. Propiedades estadísticas

Suponga que $x_0 = 0$ y que $x \in B_j$ fijo, entonces

$$\hat{f}_h(m_j) = \frac{1}{nh} \sum_{i=1}^n I(X_i \in B_j)$$

2.1.4. Sesgo

El cálculo del sesgo es el

$$\begin{aligned} \mathbb{E} [\hat{f}_h(m_j)] &= \frac{1}{nh} \sum_{i=1}^n \mathbb{E} [I(X_i \in B_j)] \\ &= \frac{1}{nh} n \mathbb{E} [I(X_i \in B_j)] \end{aligned}$$

$I(X_i \in B_j)$ es una indicadora con probabilidad de 1 de $\int_{(j-1)h}^{jh} f(u)du$ y 0 sino.

Entonces

$$\mathbb{E} [I(X_i \in B_j)] = \mathbb{P} (I(X_i \in B_j) = 1) = \int_{(j-1)h}^{jh} f(u)du.$$

Entonces,

$$\mathbb{E} [f_h(m_j)] = \frac{1}{h} \int_{(j-1)h}^{jh} f(u)du$$

$$\text{Sesgo}(\hat{f}_h(m_j)) = \frac{1}{h} \int_{(j-1)h}^{jh} f(u)du - f(x)$$

Esto se puede aproximar usando Taylor alrededor del centro $m_j = jh - h/2$ de B_j de modo que $f(u) - f(x) \approx f'(m_j)(u - x)$.

$$\text{Sesgo}(\hat{f}_h(m_j)) = \frac{1}{h} \int_{(j-1)h}^{jh} f(u)du - f(x) \approx f'(m_j)(m_j - x)$$

2.1.5. Varianza

Dado que todos los X_i son i.i.d., entonces

$$\begin{aligned} \text{Var}(\hat{f}_h(m_j)) &= \text{Var}\left(\frac{1}{nh} \sum_{i=1}^n I(X_i \in B_j)\right) \\ &= \frac{1}{n^2 h^2} n \text{Var}(I(X_i \in B_j)) \end{aligned}$$

La variable I es una bernoulli con parametro $\int_{(j-1)h}^{jh} f(u)du$ por lo tanto su varianza es el

$$\text{Var}(\hat{f}_h(x)) = \frac{1}{nh^2} \left(\int_{(j-1)h}^h f(u)du \right) \left(1 - \int_{(j-1)h}^h f(u)du \right)$$

Ejercicio 2.1. Usando un desarrollo de Taylor como en la parte anterior, pruebe que:

$$\text{Var}(\hat{f}_h(x)) \approx \frac{1}{nh} f(x)$$

2.1.6. Error cuadrático medio

El error cuadrático medio del histograma es el

$$\text{MSE}(\hat{f}_h(x)) = \text{E}\left[\left(\hat{f}_h(x) - f(x)\right)^2\right] = \text{Sesgo}^2(\hat{f}_h(x)) + \text{Var}(\hat{f}_h(x)).$$

Ejercicio 2.2. ¿Pueden probar la segunda igualdad de la expresión anterior?

Solución. Prueba segunda igualdad:

$$\begin{aligned} \text{Sesgo}^2(\hat{f}_h(x)) + \text{Var}(\hat{f}_h(x)) &= \\ [E(\hat{f}_h(x)) - f(x)]^2 + E[(E(\hat{f}_h(x)) - \hat{f}_h(x))^2] &= \\ E\left[[E(\hat{f}_h(x)) - f(x)]^2 + (E(\hat{f}_h(x)) - \hat{f}_h(x))^2\right] &\quad (*) \end{aligned}$$

Ahora note que:

$$\begin{aligned} E[(E(\hat{f}_h(x)) - f(x))(E(\hat{f}_h(x)) - \hat{f}_h(x))] &= \\ E\left[E(\hat{f}_h(x))^2\right] - E\left[E(\hat{f}_h(x)) \cdot \hat{f}_h(x)\right] - E\left[f(x) \cdot E(\hat{f}_h(x))\right] + \\ E\left[f(x) \cdot \hat{f}_h(x)\right] &= \\ E(\hat{f}_h(x))^2 - E(\hat{f}_h(x))^2 - E(\hat{f}_h(x)) \cdot E(f(x)) + E(f(x)) \cdot E(\hat{f}_h(x)) \\ &= 0 \end{aligned}$$

Entonces:

$$\begin{aligned} (*) &= E\left[[E(\hat{f}_h(x)) - f(x)]^2 - \right. \\ &\quad \left. 2(E(\hat{f}_h(x)) - f(x))(E(\hat{f}_h(x)) - \hat{f}_h(x)) + (E(\hat{f}_h(x)) - \hat{f}_h(x))^2\right] = \\ E\left[(E(\hat{f}_h(x)) - f(x) - E(\hat{f}_h(x)) + \hat{f}_h(x))^2\right] &= \\ E\left[(\hat{f}_h(x) - f(x))^2\right] \end{aligned}$$

□

Retomando los términos anteriores se tiene que

$$\begin{aligned} \text{MSE}(\hat{f}_h(x)) &= \frac{1}{nh} f(x) + f' \left\{ \left(j - \frac{1}{2}\right) h \right\}^2 \left\{ \left(j - \frac{1}{2}\right) h - x \right\}^2 \\ &\quad + o(h) + o\left(\frac{1}{nh}\right) \end{aligned}$$

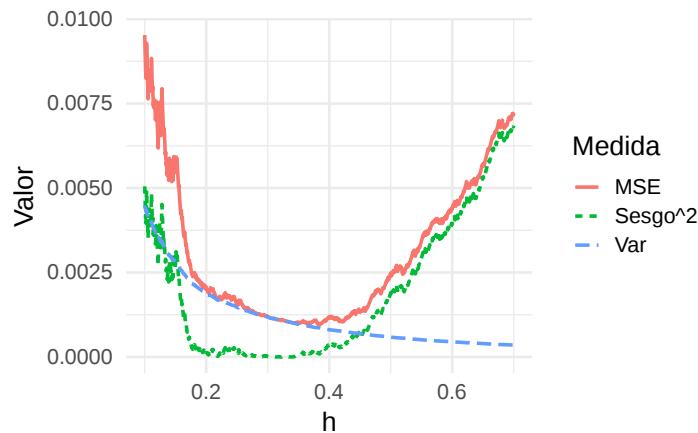
Nota: Si $h \rightarrow 0$ y $nh \rightarrow \infty$ entonces $\text{MSE}(\hat{f}_h(x)) \rightarrow 0$. Es decir, conforme usamos más observaciones, pero el ancho de banda de banda no decrece tan rápida, entonces el error cuadrático medio converge a 0.

Esto indica que si $\text{MSE}(\hat{f}_h(x)) \rightarrow 0$ (convergencia en \mathbb{L}^2) implica que $\hat{f}_h(x) \xrightarrow{\mathcal{P}} f(x)$, por lo tanto \hat{f}_h es consistente.

La fórmula anterior tiene la siguiente particularidad

- Si $h \rightarrow 0$, la varianza crece (converge a ∞) y el sesgo decrece (converge a $f'(0)x^2$).
- Si $h \rightarrow \infty$, la varianza decrece (hacia 0) y el sesgo crece (hacia ∞)

Note que la figura siguiente tiene esa propiedad.



2.1.7. Error cuadrático medio integrado

El problema con el $\text{MSE}(\hat{f}_h(x))$ es que depende completamente del punto escogido x .

La solución a esto es integrar el MSE.

$$\begin{aligned}
\text{MISE}(\hat{f}_h(x)) &= \mathbb{E} \left[\int_{-\infty}^{\infty} \left\{ \hat{f}_h(x) - f(x) \right\}^2 dx \right] \\
&= \int_{-\infty}^{\infty} \mathbb{E} \left[\left\{ \hat{f}_h(x) - f(x) \right\}^2 \right] dx \\
&= \int_{-\infty}^{\infty} \text{MSE}(\hat{f}_h(x)) dx
\end{aligned}$$

Además,

$$\begin{aligned}
\text{MISE}(\hat{f}_h(x)) &= \int_{-\infty}^{\infty} \frac{1}{nh} f(x) dx \\
&\quad + \int_{-\infty}^{\infty} \sum_j I(x \in B_j) \left\{ \left(j - \frac{1}{2} \right) h - x \right\}^2 \left[f' \left(\left\{ j - \frac{1}{2} \right\} h \right) \right]^2 dx \\
&= \frac{1}{nh} + \sum_j \left[f' \left(\left\{ j - \frac{1}{2} \right\} h \right) \right]^2 \int_{B_j} \left\{ \left(j - \frac{1}{2} \right) h - x \right\}^2 dx \\
&= \frac{1}{nh} + \frac{h^2}{12} \sum_j \left[f' \left(\left\{ j - \frac{1}{2} \right\} h \right) \right]^2 \\
&\approx \frac{1}{nh} + \frac{h^2}{12} \int \{f'(x)\}^2 dx \\
&= \frac{1}{nh} + \frac{h^2}{12} \|f'\|_2^2
\end{aligned}$$

2.1.8. Ancho de banda óptimo para el histograma

El MISE tiene el mismo comportamiento que el MSE. La figura siguiente presenta el comportamiento de la varianza, sesgo y MISE para nuestro ejemplo.

La mala elección del parámetro h causa que el histograma no capture toda la estructura de los datos.

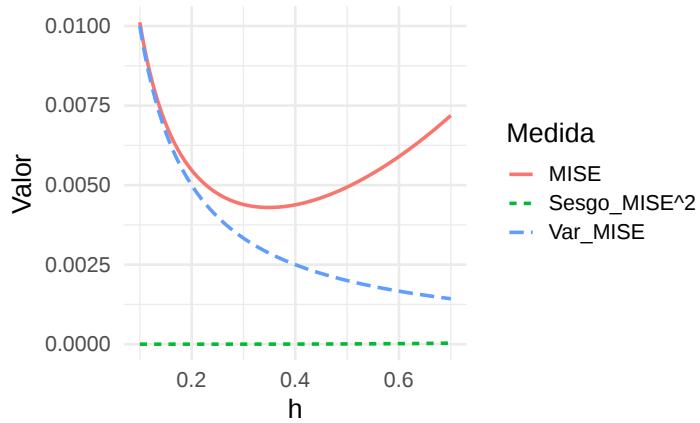
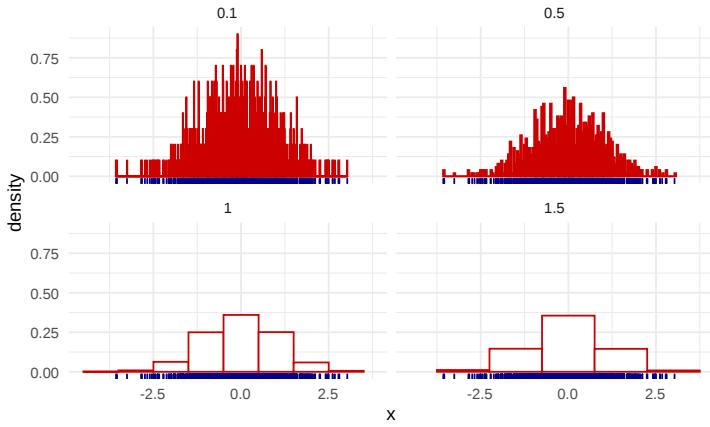


Figura 2.1:



En este caso se puede simplemente minimizar el MISE de la forma usual,

$$\frac{\partial \text{MISE}(f_h)}{\partial h} = -\frac{1}{nh^2} + \frac{1}{6}h\|f'\|_2^2 = 0$$

implica que

$$h_{opt} = \left(\frac{6}{n\|f'\|_2^2} \right)^{1/3} = O(n^{1/3}).$$

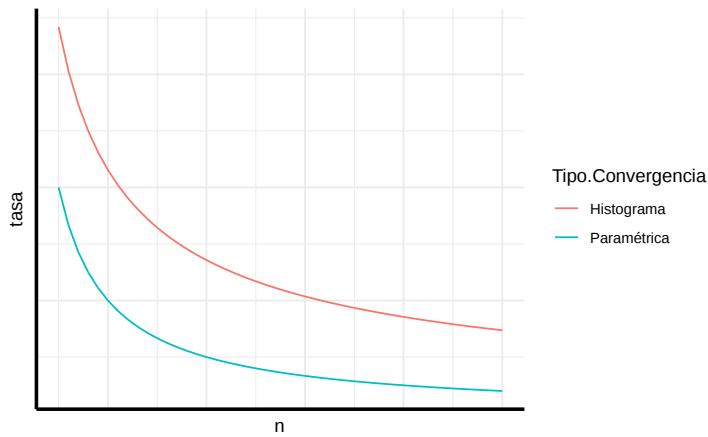
y que por lo tanto

$$\text{MISE}(\hat{f}_h) = \frac{1}{n} \left(\frac{n \|f'\|_2^2}{6} \right)^{1/3}$$

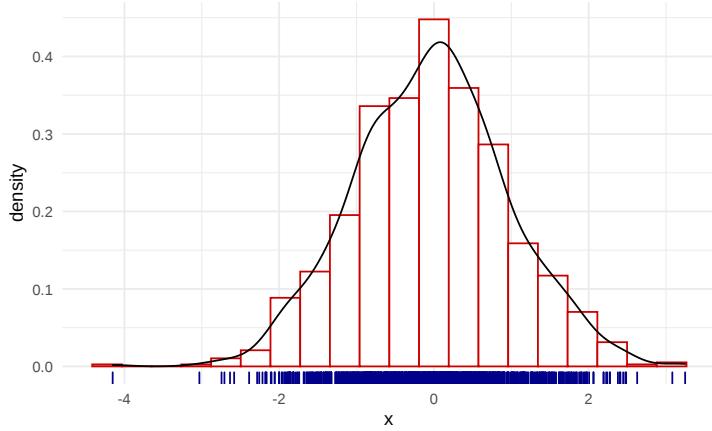
Nota: (Recuerde de Estadística I). Si $X_1, \dots, X_n \sim f_\theta$ i.i.d, con $\text{Var}(X) = \sigma^2$, recuerde que el estimador $\hat{\theta}$ de θ tiene la característica que

$$\text{MSE}(\theta) = \text{Var}(\hat{\theta}) + \text{Sesgo}^2(\hat{\theta}) = \frac{\sigma^2}{n}$$

Según la nota anterior la tasas de convergencia del histograma es más lenta que la de un estimador parámetrico considerando la misma cantidad de datos.



Finalmente, podemos encontrar el valor óptimo de esta datos dado por $h = \text{h_opt_MISE}$



2.2. Estimación No-paramétrica de densidad

2.2.1. Primera construcción

Sea X_1, \dots, X_n variables aleatorias i.i.d. con distribución f en \mathbb{R} .

La distribución de f es $F(x) = \int_{-\infty}^x f(t)dt$.

Considere la distribución empírica como

$$F_n(x) = \frac{1}{n} \sum_{i=1}^n I(X_i \leq x).$$

Por la ley de los grandes números tenemos que $\hat{F}_n(x) \xrightarrow{c.s.} F(x)$ para todo x en \mathbb{R} as $n \rightarrow \infty$. Entonces, $F_n(x)$ es consistente

para todo x in \mathbb{R} .

Nota: . ¿Podríamos derivar \hat{F}_n para encontrar el estimar \hat{f}_n ?

La respuesta es si (más o menos).

Suponga que $h > 0$ tenemos la aproximación

$$f(x) \approx \frac{F(x+h) - F(x-h)}{2h}.$$

Remplazando F por su estimador \hat{F}_n , defina

$$\hat{f}_n^R(x) = \frac{F_n(x+h) - F_n(x-h)}{2h},$$

donde $\hat{f}_n^R(x)$ es el estimador de *Rosenblatt*.

Podemos rescribirlo de la forma,

$$\hat{f}_n^R(x) = \frac{1}{2nh} \sum_{i=1}^n I(x-h < X_i \leq x+h) = \frac{1}{nh} \sum_{i=1}^n K_0\left(\frac{X_i - x}{h}\right)$$

con $K_0(u) = \frac{1}{2}I(-1 < u \leq 1)$, lo cuál es equivalente al caso del histograma.

2.2.2. Otra construcción

Con el histograma construimos una serie de segmentos fijo B_j y contábamos el número de datos que estaban **CONTENIDOS** en B_j

Nota: . ¿Qué pasaría si cambiamos la palabra **CONTENIDOS** por **ALREDOR DE «x»?**

Suponga que se tienen intervalos de longitud $2h$, es decir, intervalos de la forma $[x-h, x+h]$.

El histograma se escribe como

$$\hat{f}_h(x) = \frac{1}{2hn} \#\{X_i \in [x-h, x+h]\}.$$

Ahora tratemos de modificar ligeramente esta expresión notando dos cosas

1.

$$\frac{1}{2}I(|u| \leq 1)$$

$$\text{con } u = \frac{x-x_i}{h}$$

2.

$$\frac{1}{2} \# \{X_i \in [x-h, x+h]\} = \sum_{i=1}^n K\left(\frac{x-x_i}{h}\right) = \sum_{i=1}^n \frac{1}{2} I\left(\left|\frac{x-x_i}{h}\right| \leq 1\right)$$

\end{enumerate}

Finalmente se tiene que

$$\hat{f}_h(x) = \frac{1}{nh} \sum_{i=1}^n K\left(\frac{x-x_i}{h}\right)$$

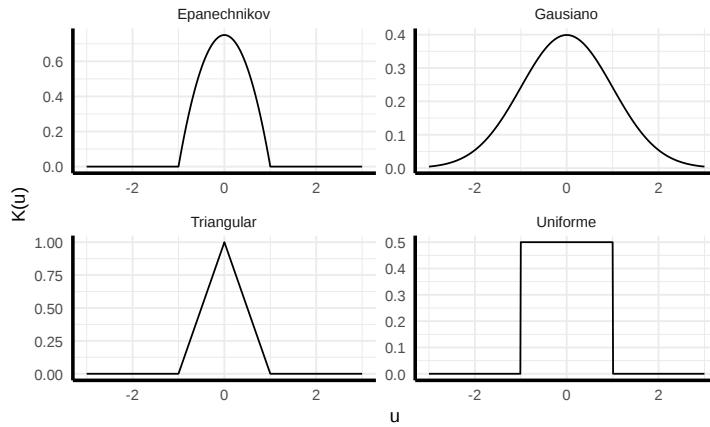
Nota: . ¿Qué pasaría si cambiaríamos la función K del histograma por una más general?

Esta función debería cumplir las siguientes características

- $K(u) \geq 0$.
- $\int_{-\infty}^{\infty} K(u)du = 1$.
- $\int_{-\infty}^{\infty} uK(u)du = 0$.
- $\int_{-\infty}^{\infty} u^2 K(u)du < \infty$.

Por ejemplo:

- **Uniforme:** $\frac{1}{2}I(|u| \leq 1)$.
- **Triangular:** $(1 - |u|)I(|u| \leq 1)$.
- **Epanechnikov:** $\frac{3}{4}(1 - u^2)I(|u| \leq 1)$.
- **Gausiano:** $\frac{1}{\sqrt{2\pi}} \exp\left(-\frac{1}{2}u^2\right)$.

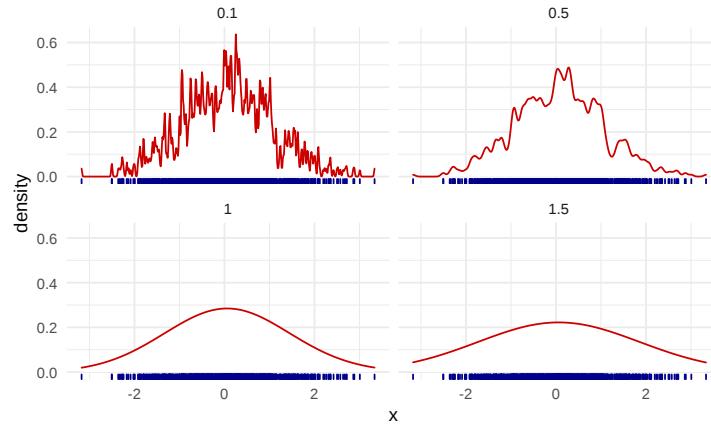


Entonces se tendría que la expresión general para un estimador por núcleos es

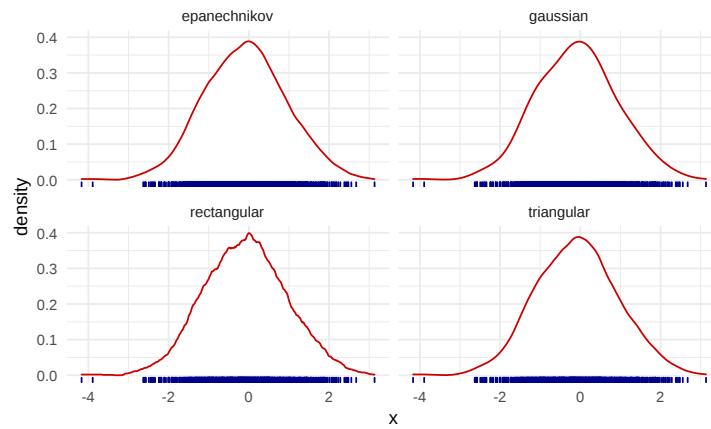
$$\hat{f}_h(x) = \frac{1}{nh} \sum_{i=1}^n K\left(\frac{x - x_i}{h}\right)$$

Nota: . ¿Qué pasaría si modificamos el ancho de banda h para un mismo kernel?

Nuevamente sería el ancho de banda ya que



Nota: . ¿Qué pasaría si modificamos el kernel para un mismo ancho de banda h ?

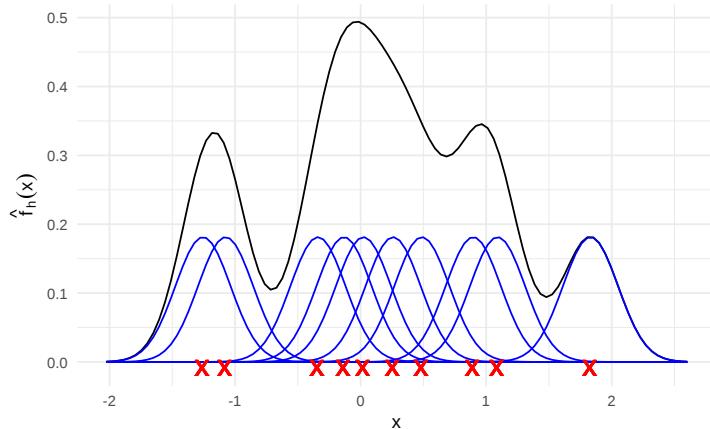


Recordemos nuevamente la fórmula

$$\hat{f}_h(x) = \frac{1}{nh} \sum_{i=1}^n K\left(\frac{x - X_i}{h}\right)$$

Cada sumando de esta expresión es una función por si misma. Si la integramos se obtiene que

$$\frac{1}{nh} \int K\left(\frac{x - X_i}{h}\right) dx = \frac{1}{nh} \int K(u) h du = \frac{1}{n} \int K(u) du = \frac{1}{n}$$



2.3. Propiedades Estadísticas

Nota: . ¿Podríamos imitar lo mismo que hicimos para el histograma?

Si. Las propiedades que vimos anteriormente son universales para estimadores.

Entonces:

$$\begin{aligned} \text{MSE}(\hat{f}_h(x)) &= \text{Var}(\hat{f}_h(x)) + \text{Sesgo}^2(\hat{f}_h(x)) \\ \text{MISE}(\hat{f}_h) &= \int \text{Var}(\hat{f}_h(x)) dx + \int \text{Sesgo}^2(\hat{f}_h(x)) dx \end{aligned}$$

donde

$$\text{Var}(\hat{f}_h(x)) = \mathbb{E} [\hat{f}_h(x) - \mathbb{E} \hat{f}_h(x)]^2 \text{ and } \text{Sesgo}(\hat{f}_h(x)) = \mathbb{E} [\hat{f}_h(x)] - f(x).$$

2.3.1. Varianza

$$\begin{aligned}
\text{Var}(\hat{f}_h(x)) &= \text{Var}\left(\frac{1}{n} \sum_{i=1}^n K\left(\frac{x-X_i}{h}\right)\right) \\
&= \frac{1}{n^2 h^2} \sum_{i=1}^n \text{Var}\left(K\left(\frac{x-X_i}{h}\right)\right) \\
&= \frac{1}{nh^2} \text{Var}\left(K\left(\frac{x-X}{h}\right)\right) \\
&= \frac{1}{nh^2} \left\{ \mathbb{E}\left[K^2\left(\frac{x-X}{h}\right)\right] - \left\{\mathbb{E}\left[K\left(\frac{x-X}{h}\right)\right]\right\}^2 \right\}.
\end{aligned}$$

Usando que:

$$\begin{aligned}
\mathbb{E}\left[K^2\left(\frac{x-X}{h}\right)\right] &= \int K^2\left(\frac{x-s}{h}\right) f(s) ds \\
&= h \int K^2(u) f(uh+x) du \\
&= h \int K^2(u) \{f(x) + o(1)\} du \\
&= h \left\{ \|K\|_2^2 f(x) + o(1) \right\}.
\end{aligned}$$

$$\begin{aligned}
\mathbb{E}\left[K\left(\frac{x-X}{h}\right)\right] &= \int K\left(\frac{x-s}{h}\right) f(s) ds \\
&= h \int K(u) f(uh+x) du \\
&= h \int K(u) \{f(x) + o(1)\} du \\
&= h \{f(x) + o(1)\}.
\end{aligned}$$

Por lo tanto se obtiene que

$$\text{Var}\left(\hat{f}_h(x)\right) = \frac{1}{nh} \|K\|_2^2 f(x) + o\left(\frac{1}{nh}\right), \text{ si } nh \rightarrow \infty.$$

2.3.2. Sesgo

Para el sesgo tenemos

$$\begin{aligned}
 \text{Sesgo}(\hat{f}_h(x)) &= \mathbb{E}[\hat{f}_h(x)] - f(x) \\
 &= \frac{1}{nh} \sum_{i=1}^n \mathbb{E}\left[K\left(\frac{x-X_i}{h}\right)\right] - f(x) \\
 &= \frac{1}{h} \mathbb{E}\left[K\left(\frac{x-X_1}{h}\right)\right] - f(x) \\
 &= \int \frac{1}{h} K\left(\frac{x-u}{h}\right) f(u) du - f(x)
 \end{aligned}$$

Ejercicio 2.3. Usando el cambio de variable $s = \frac{u-x}{h}$ y las propiedades del kernel pruebe que

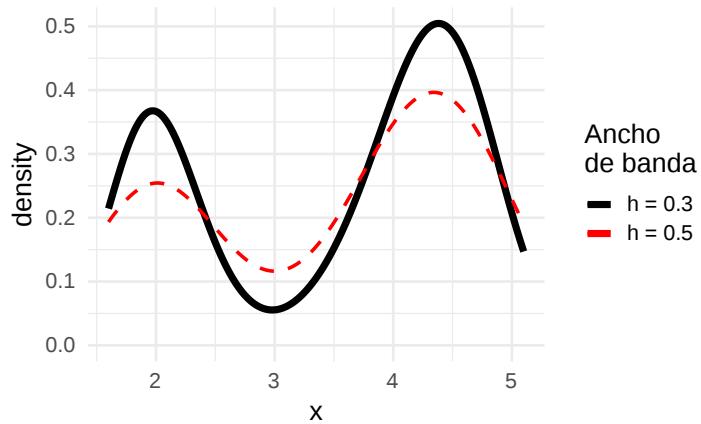
$$\text{Sesgo}(\hat{f}_h(x)) = \frac{h^2}{2} f'' \mu_2(K) + o(h^2), \text{ si } h \rightarrow 0$$

donde $\mu_2 = \int s^2 K(s) ds$.

***Nota:** En algunas pruebas más formales, se necesita además que f'' sea absolutamente continua y que $\int (f'''(x))^2 dx < \infty$.*

Solución.

$$\begin{aligned}
 \text{Sesgo}(\hat{f}_h(x)) &= \int \frac{1}{h} K\left(\frac{x-u}{h}\right) f(u) du - f(x) \\
 &= \frac{1}{h} \int h K(s) f(sh+x) ds - f(x) \\
 &= \int K(s) \left[f(x) + f'(x)(sh+x-x) \right. \\
 &\quad \left. + \frac{f''(x)}{2}(sh+x-x)^2 + o(h^2) \right] - f(x) \\
 &= \int K(s) f(x) ds + \int h f'(x) s K(s) ds \\
 &\quad + \int \frac{h^2}{2} f''(x) s^2 K(s) ds + o(h^2) - f(x) \\
 &= f(x) + 0 + \frac{h^2}{2} f''(x) \mu_2(K) + o(h^2) - f(x) \\
 &= \frac{h^2}{2} f''(x) \mu_2(K) + o(h^2)
 \end{aligned}$$



Nota: . Note como los cambios en el ancho de banda modifican la suavidad (sesgo) y el aplanamiento de la curva (varianza).

2.3.3. Error cuadrático medio y Error cuadrático medio integrado

El error cuadrático medio se escribe

$$\begin{aligned}\text{MSE}(\hat{f}_h(x)) &= \text{Sesgo}(\hat{f}_h(x))^2 + \text{Var}(\hat{f}_h(x)) \\ &= \frac{h^4}{4} (\mu_2(K) f''(x))^2 + \frac{1}{nh} \|K\|_2^2 f(x) + o(h^4) + o\left(\frac{1}{nh}\right).\end{aligned}$$

Y el error cuadrático medio integrado se escribe como,

$$\begin{aligned}\text{MISE}(\hat{f}_h) &= \int \text{MSE}(\hat{f}_h(x)) dx \\ &= \int \text{Sesgo}(\hat{f}_h(x))^2 + \text{Var}(\hat{f}_h(x)) dx \\ &= \frac{h^4}{4} \mu_2^2(K) \|f''(x)\|_2^2 + \frac{1}{nh} \|K\|_2^2 + o(h^4) + o\left(\frac{1}{nh}\right).\end{aligned}$$

2.3.4. Ancho de banda óptimo

Minimizando el MISE con respecto a h obtenemos

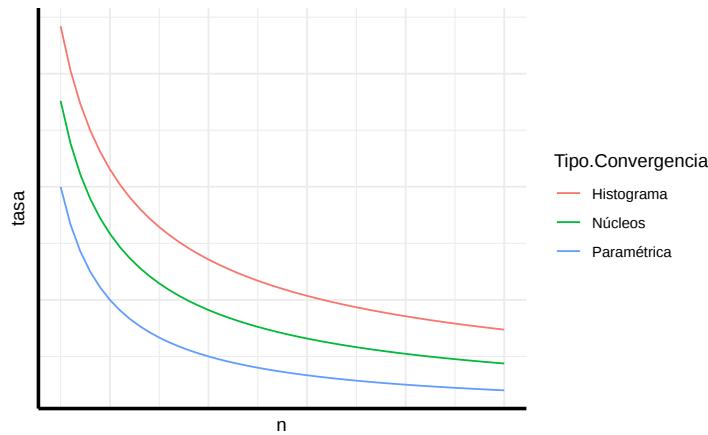
$$h_{opt} = \left(\frac{\|K\|_2^2}{\|f''\|_2^2 (\mu_2(K))^2 n} \right)^{1/5} = O(n^{-1/5}).$$

Nota: . De forma práctica, h_{opt} no es un estimador útil de h porque depende de $\|f''\|_2^2$ que es desconocido.

Más adelante veremos otra forma de encontrar este estimador.

Evaluando h_{opt} en el MISE tenemos que

$$\text{MISE}(\hat{f}_h) = \frac{5}{4} \left(\|K\|_2^2 \right)^{4/5} \left(\|f''\|_2^2 \mu_2(K) \right)^{2/5} n^{-4/5} = O(n^{-4/5}).$$



Nota: . Formalmente, es posible probar que si f es β veces continuamente diferenciable y $\int (f^{(\beta)})^2 < \infty$, entonces se tiene que

$$h_{opt} = O\left(n^{-\frac{1}{2\beta+1}}\right).$$

Por lo tanto se podría aproximar a una tasa paramétrica de convergencia si β es grande.

2.4. Escogiendo el ancho de banda

Nota: . La principal característica del ancho de banda

$$h_{opt} = \left(\frac{\|K\|_2^2}{\|f''\|_2^2 (\mu_2(K))^2 n} \right)^{1/5} = O\left(n^{-1/5}\right).$$

ES QUE ¡NO FUNCIONA!

Veremos dos métodos para determinar un h que funcione:

- Referencia normal.
- Validación cruzada.

2.4.1. Referencia normal

Nota: . Este método es más efectivo si se conoce que la verdadera distribución es bastante suave, unimodal y simétrica.

Más adelante veremos otro método para densidades más generales.

Asuma que f es normal distribuida y se utiliza un kernel K gausiano. Entonces se tiene que

$$\begin{aligned}\hat{h}_{rn} &= \left(\frac{\|K\|_2^2}{\|f''\|_2^2 (\mu_2(K))^2 n} \right)^{1/5} = O(n^{-1/5}) \\ &= 1,06\hat{\sigma}n^{-1/5}.\end{aligned}$$

donde

$$\hat{\sigma} = \sqrt{\frac{1}{n-1} \sum_{i=1}^n (x_i - \bar{x}^2)}$$

Ejercicio 2.4. Pruebe que la ecuación anterior es verdadera. Es decir, calcule $\|K\|_2^2$, $\|f''\|_2^2$ y $\mu_2(K)$

Nota: . Un problema con $\hat{h}_{rn} = 1,06\hat{\sigma}n^{-1/5}$ es su sensibilidad a los valores extremos.

Ejemplo 2.1. La varianza empírica de 1, 2, 3, 4, 5, es 2.5.

La varianza empírica de 1, 2, 3, 4, 5, 99, es 1538.

El rango intercuantil IQR se define como

$$\text{IQR}^X = Q_3^X - Q_1^X$$

donde Q_1^X y Q_3^X son el primer y tercer de un conjunto de datos X_1, \dots, X_n .

Con el supuesto que $X \sim \mathcal{N}(\mu, \sigma^2)$ entonces $Z = \frac{X - \mu}{\sigma} \sim \mathcal{N}(0, 1)$.

Entonces,

$$\begin{aligned}\text{IQR} &= Q_3^X - Q_1^X \\ &= (\mu + \sigma Q_3^Z) - (\mu + \sigma Q_1^Z) \\ &= \sigma (Q_3^Z - Q_1^Z) \\ &\approx \sigma (0,67 - (-0,67)) \\ &= 1,34\sigma.\end{aligned}$$

Por lo tanto $\hat{\sigma} = \frac{\widehat{\text{IQR}}^X}{1,34}$

Podemos sustituir la varianza empírica de la fórmula inicial y tenemos

$$\hat{h}_{rn} = 1,06 \frac{\widehat{\text{IQR}}^X}{1,34} n^{-\frac{1}{5}} \approx 0,79 \widehat{\text{IQR}}^X n^{-\frac{1}{5}}$$

Combinando ambos estimadores, podemos obtener,

$$\hat{h}_{rn} = 1,06 \min \left\{ \frac{\widehat{\text{IQR}}^X}{1,34}, \hat{\sigma} \right\} n^{-\frac{1}{5}}$$

2.4.2. Validación Cruzada

Defina el *error cuadrático integrado* como

$$\begin{aligned}\text{ISE}(\hat{f}_h) &= \int (\hat{f}_h(x) - f(x))^2 dx \\ &= \int \hat{f}_h^2(x) dx - 2 \int \hat{f}_h(x) f(x) dx + \int f^2(x) dx.\end{aligned}$$

Nota: . El MISE es el valor esperado del ISE.

Nuestro objetivo es minimizar el ISE con respecto a h .

Primero note que $\int f^2(x) dx$ NO DEPENDE de h . Podemos minimizar la expresión

$$\text{ISE}(\hat{f}_h) - \int f^2(x) dx = \int \hat{f}_h^2(x) dx - 2 \int \hat{f}_h(x) f(x) dx$$

Vamos a resolver esto en dos pasos partes

Integral $\int \hat{f}_h(x)f(x)dx$

El término $\int \hat{f}_h(x)f(x)dx$ es el valor esperado de $E[\hat{f}(X)]$. Su estimador es

$$E[\widehat{\hat{f}(X)}] = \frac{1}{n} \sum_{i=1}^n \hat{f}_h(X_i) = \frac{1}{n^2 h} \sum_{i=1}^n \sum_{j=1}^n K\left(\frac{X_j - X_i}{h}\right).$$

Nota: . El problema con esta expresión es que las observaciones que se usan para estimar la esperanza son las misma que se usan para estimar $\hat{f}_h(x)$ (Se utilizan doble).

La solución es remover la $i^{\text{ésima}}$ observación de \hat{f}_h para cada i .

Redefiniendo el estimador anterior tenemos $\int \hat{f}_h(x)f(x)dx$ como

$$\frac{1}{n} \sum_{i=1}^n \hat{f}_{h,-i}(X_i),$$

donde

$$\hat{f}_{h,-i}(x) = \frac{1}{(n-1)h} \sum_{\substack{j=1 \\ j \neq i}}^n K\left(\frac{x - X_i}{h}\right).$$

Integral $\int \hat{f}_h^2(x)dx$

Siguiendo con el término $\int \hat{f}_h^2(x)dx$ note que este se puede reescribir como

$$\begin{aligned} \int \hat{f}_h^2(x)dx &= \int \left(\frac{1}{nh} \sum_{i=1}^n K\left(\frac{x - X_i}{h}\right) \right)^2 dx \\ &= \frac{1}{n^2 h^2} \sum_{i=1}^n \sum_{i=1}^n \int K\left(\frac{x - X_i}{h}\right) K\left(\frac{x - X_j}{h}\right) dx \\ &= \frac{1}{n^2 h} \sum_{i=1}^n \sum_{i=1}^n \int K(u) K\left(\frac{X_i - X_j}{h} - u\right) du \\ &= \frac{1}{n^2 h} \sum_{i=1}^n \sum_{i=1}^n K * K\left(\frac{X_i - X_j}{h}\right). \end{aligned}$$

donde $K * K$ es la convolución de K consigo misma.

Finalmente tenemos la función,

$$\text{CV}(h) = \frac{1}{n^2 h} \sum_{i=1}^n \sum_{j=1}^n K * K \left(\frac{X_i - X_j}{h} \right) - \frac{2}{(n-1)h} \sum_{i=1}^n \sum_{\substack{j=1 \\ j \neq i}}^n K \left(\frac{X_i - X_j}{h} \right).$$

Nota: . Note que $\text{CV}(h)$ no depende de f o sus derivadas.

Nota: . Para efectos prácticos es mejor utilizar el criterio

$$CV(h) = \int \hat{f}_h^2(x) dx - \frac{2}{(n-1)h} \sum_{i=1}^n \sum_{\substack{j=1 \\ j \neq i}}^n K \left(\frac{X_i - X_j}{h} \right)$$

y luego calcular numéricamente la integral.

2.4.3. Intervalos de confianza para estimadores de densidad no paramétricos

Usando los resultados anteriores y asumiendo que $h = cn^{-\frac{1}{5}}$ entonces

$$n^{-\frac{2}{5}} \left\{ \hat{f}_h(x) - f(x) \right\} \xrightarrow{\mathcal{L}} \mathcal{N} \left(\underbrace{\frac{c^2}{2} f'' \mu_2(K)}_{b_x}, \underbrace{\frac{1}{c} f(x) \|K\|_2^2}_{v_x} \right).$$

Si $z_{1-\frac{\alpha}{2}}$ es el cuantil $1 - \frac{\alpha}{2}$ de una distribución normal estándar, entonces

$$\begin{aligned} 1 - \alpha &\approx \mathbb{P} \left(b_x - z_{1-\frac{\alpha}{2}} v_x \leq n^{2/5} \left\{ \hat{f}_h(x) - f(x) \right\} \leq b_x + z_{1-\frac{\alpha}{2}} v_x \right) \\ &= \mathbb{P} \left(\hat{f}_h(x) - n^{-2/5} \left\{ b_x + z_{1-\frac{\alpha}{2}} v_x \right\} \leq f(x) \leq \hat{f}_h(x) - n^{-2/5} \left\{ b_x - z_{1-\frac{\alpha}{2}} v_x \right\} \right) \end{aligned}$$

Esta expresión nos dice que con una probabilidad de $1 - \alpha$ se tiene que

$$\left[\hat{f}_h(x) - \frac{h^2}{2} f''(x) \mu_2(K) - z_{1-\frac{\alpha}{2}} \sqrt{\frac{f(x) \|K\|_2^2}{nh}}, \hat{f}_h(x) - \frac{h^2}{2} f''(x) \mu_2(K) + z_{1-\frac{\alpha}{2}} \sqrt{\frac{f(x) \|K\|_2^2}{nh}} \right]$$

Al igual que en los casos anteriores, este intervalo no es útil ya que depende de $f(x)$ y $f''(x)$.

Si h es pequeño relativamente a $n^{-\frac{1}{5}}$ entonces el segundo término $\frac{h^2}{2} f''(x) \mu_2(K)$ podría ser ignorado.

Podemos reemplazar $f(x)$ por su estimador $\hat{f}_h(x)$. Entonces tendríamos una intervalo aplicable a nuestro caso

$$\left[\hat{f}_h(x) - z_{1-\frac{\alpha}{2}} \sqrt{\frac{\hat{f}_h(x) \|K\|_2^2}{nh}}, \hat{f}_h(x) + z_{1-\frac{\alpha}{2}} \sqrt{\frac{\hat{f}_h(x) \|K\|_2^2}{nh}} \right]$$

Nota: . Este intervalo de confianza solo funciona en cada punto particular de $f(x)$.

Existe una versión más general para determinar la banda de confianza de toda la función. Por favor revisar la página 62 de (**Hardle2004**).

2.5. Laboratorio

Comenzaremos con una librería bastante básica llamada **KernSmooth**.

2.5.1. Efecto de distintos Kernels en la estimación

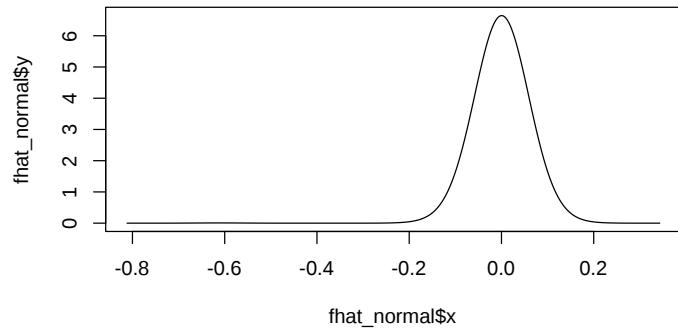
```
x <- read.csv("data/stockres.txt")
x <- unlist(x)
```

```
summary(x)
```

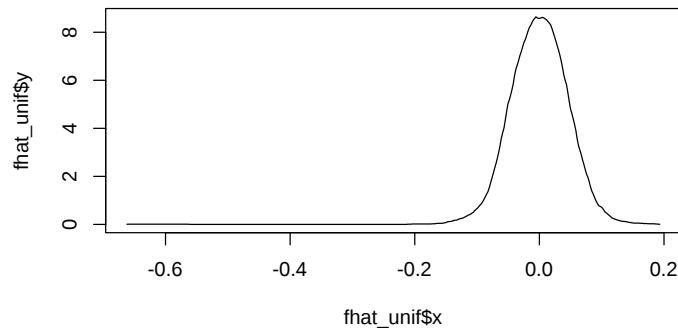
```
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## -0.6118200 -0.0204085 -0.0010632 -0.0004988  0.0215999  0.1432286
```

```
library(KernSmooth)
```

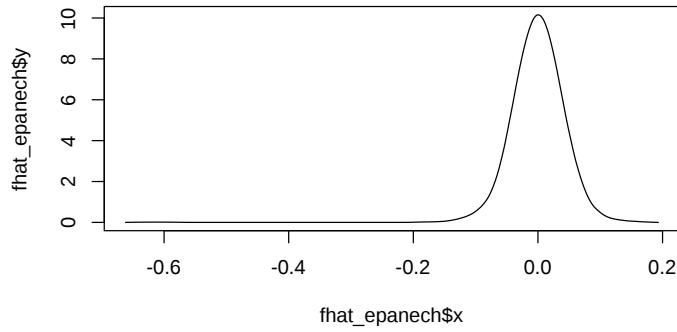
```
fhat_normal <- bkde(x, kernel = "normal", bandwidth = 0.05)
plot(fhat_normal, type = "l")
```



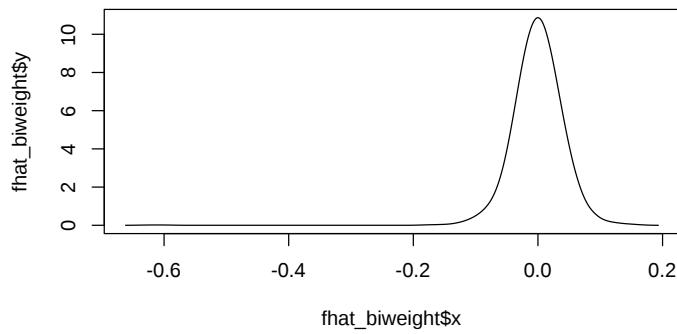
```
fhat_unif <- bkde(x, kernel = "box", bandwidth = 0.05)
plot(fhat_unif, type = "l")
```



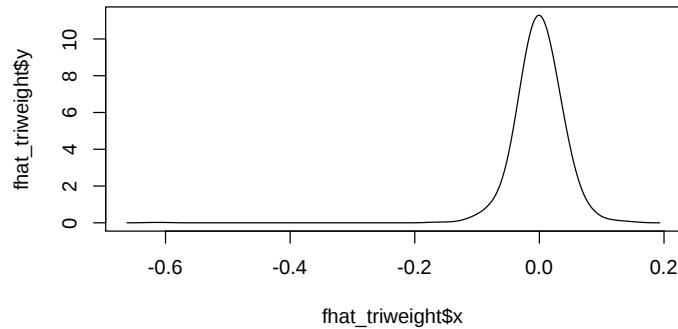
```
fhat_epanech <- bkde(x, kernel = "epanech", bandwidth = 0.05)
plot(fhat_epanech, type = "l")
```



```
fhat_biweight <- bkde(x, kernel = "biweight", bandwidth = 0.05)
plot(fhat_biweight, type = "l")
```



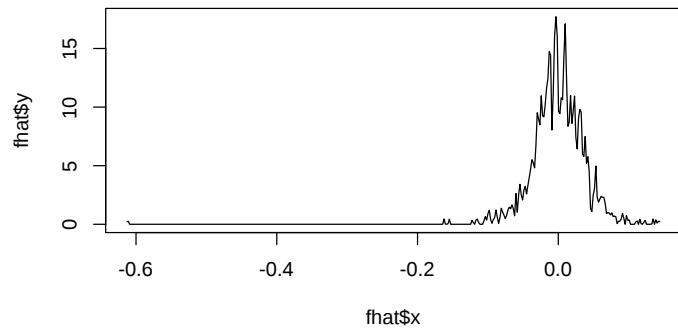
```
fhat_triweight <- bkde(x, kernel = "triweight", bandwidth = 0.05)
plot(fhat_triweight, type = "l")
```



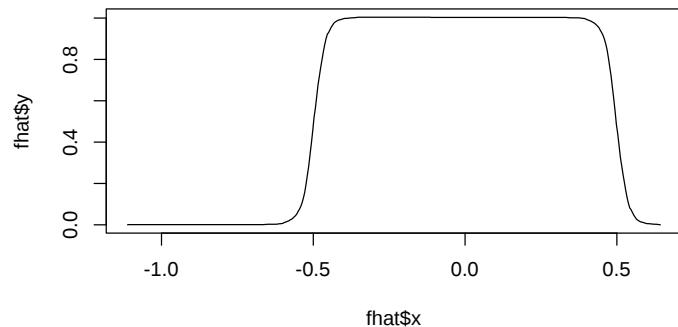
2.5.2. Efecto del ancho de banda en la estimación

** Kernel uniforme **

```
fhat <- bkde(x, kernel = "box", bandwidth = 0.001)
plot(fhat, type = "l")
```

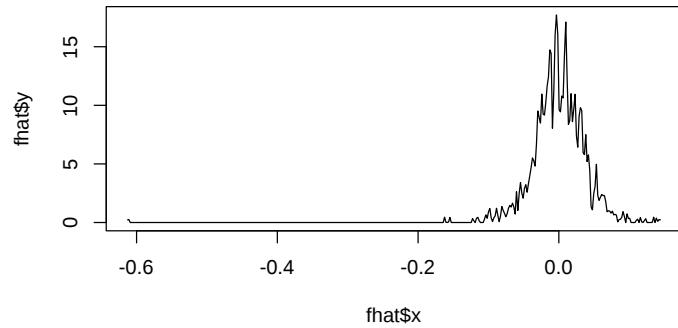


```
fhat <- bkde(x, kernel = "box", bandwidth = 0.5)
plot(fhat, type = "l")
```

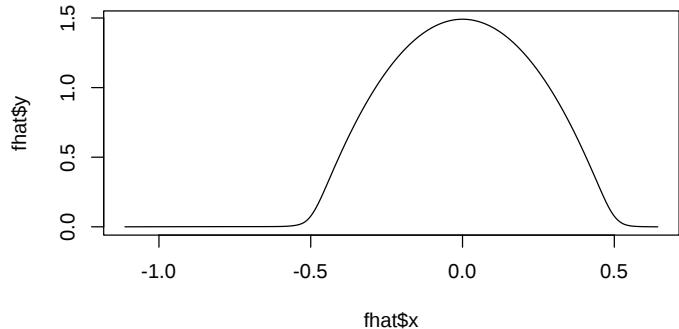


** Kernel Epanechnikov **

```
fhat <- bkde(x, kernel = "epa", bandwidth = 0.001)
plot(fhat, type = "l")
```



```
fhat <- bkde(x, kernel = "epa", bandwidth = 0.5)
plot(fhat, type = "l")
```



```

suppressMessages(library(tidyverse))
library(gganimate)

fani <- tibble()

for (b in seq(0.001, 0.02, length.out = 40)) {
  f <- bkde(x, kernel = "epa", bandwidth = b, gridsize = length(x))
  fani <- fani %>% bind_rows(tibble(xreal = sort(x),
    x = f$x, y = f$y, bw = b))
}

ggplot(data = fani) + geom_line(aes(x, y), color = "blue") +
  labs(title = paste0("Ancho de banda = {closest_state}")) +
  transition_states(bw) + view_follow() + theme_minimal(base_size = 20)

# anim_save('manual_figure/bandwidth-animation.gif')

```

Nota: . - Construya una variable llamada `u` que sea una secuencia de -0.15 a 0.15 con un paso de 0.01 - Asigne `x` a los datos `stockrel` y calcule su media y varianza. - Usando la función `dnorm` construya los valores de la distribución de los datos usando la media y varianza calculada anteriormente. Asigne a esta variable `f\Param`. - Defina un ancho de banda `h` en 0.02 - Construya un histograma para estos datos con ancho de banda `h`. Llame a esta variable `f\hist` - Usando el paquete `KernSmooth` y la función `bkde`, construya una función que calcule el estimador no paramétrico con un núcleo Epanechnikov

para un ancho de banda h . Llame a esta variable f_{epa} . - Dibuje en el mismo gráfico la estimación paramétrica y no paramétrica.

```
x <- read.csv("data/stockres.txt")
x <- unlist(x)
# Eliminar nombres de las columnas
names(x) <- NULL

u <- seq(-0.15, 0.15, by = 0.01)

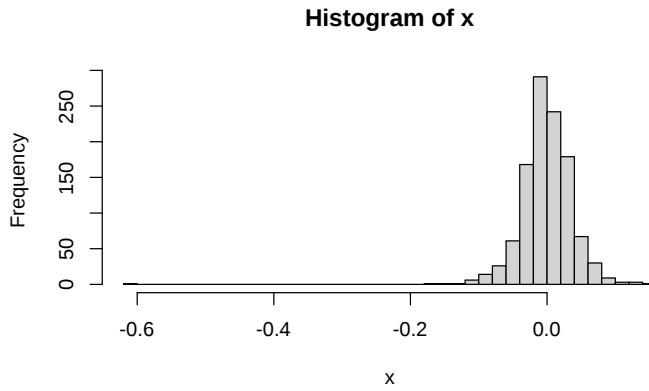
mu <- mean(x)
sigma <- sd(x)

f_param <- dnorm(u, mean = mu, sd = sigma)

h <- 0.02

n_bins <- floor(diff(range(x))/h)

f_hist <- hist(x, breaks = n_bins)
```

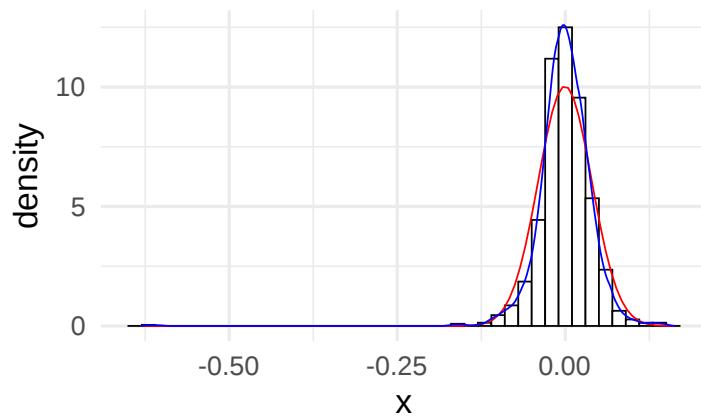


```
f_epa <- as.data.frame(bkde(x, kernel = "epa", bandwidth = h))

x_df <- data.frame(x)
```

```
library(ggplot2)

ggplot(x_df, aes(x)) + geom_histogram(aes(y = ..density..),
  binwidth = 0.02, col = "black", fill = "white") +
  stat_function(fun = dnorm, args = list(mean = mu,
    sd = sigma), color = "red") + geom_line(data = f_epa,
  aes(x, y), color = "blue") + theme_minimal(base_size = 20)
```



2.5.3. Ancho de banda óptimo

Usemos la regla de la normal o también conocida como Silverman. **Primero recuerde que en este caso se asume que $f(x)$ sigue una distribución normal.** En este caso, lo que se obtiene es que

$$\begin{aligned}\|f''\|_2^2 &= \sigma^{-5} \int \{\phi''\}^2 dx \\ &= \sigma^{-5} \frac{3}{8\sqrt{\pi}} \approx 0,212\sigma^{-5}\end{aligned}$$

donde ϕ es la densidad de una normal estándar.

El estimador para σ es

$$s = \sqrt{\frac{1}{n-1} \sum_{i=1}^n (x_i - \bar{x})^2}.$$

Y usando el cálculo realizado anteriormente, se obtiene que

$$h_{normal} = \left(\frac{4s^5}{3n} \right)^{1/5} \approx 1,06sn^{-1/5}.$$

Un estimador más robusto es

$$h_{normal} = 1,06 \min \left\{ s, \frac{IQR}{1,34} \right\} n^{-1/5}.$$

¿Por qué es $IQR/1,34$?

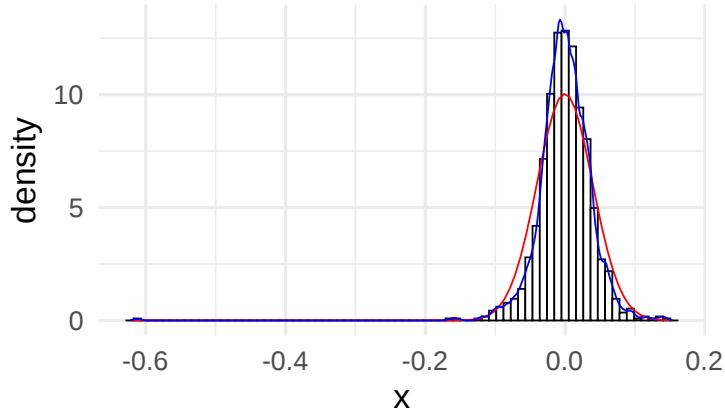
```
s <- sd(x)
n <- length(x)

h_normal <- 1.06 * s * n^(-1/5)

h <- h_normal

n_bins <- floor(diff(range(x))/h)
f_hist <- hist(x, breaks = n_bins, plot = FALSE)
f_epa <- as.data.frame(bkde(x, kernel = "epa", bandwidth = h))

ggplot(x_df, aes(x)) + geom_histogram(aes(y = ..density..),
  binwidth = h, col = "black", fill = "white") +
  stat_function(fun = dnorm, args = list(mean = mu,
    sd = sigma), color = "red") + geom_line(data = f_epa,
  aes(x, y), color = "blue") + theme_minimal(base_size = 20)
```



```

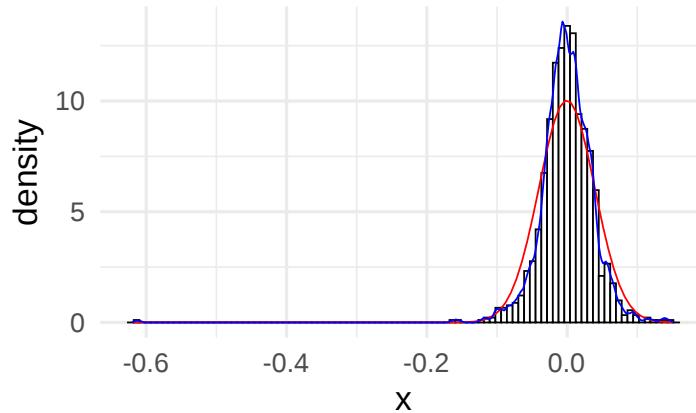
h_iqr <- 1.06 * min(s, IQR(x)/1.34) * n^(-1/5)

h <- h_iqr

n_bins <- floor(diff(range(x))/h)
f_hist <- hist(x, breaks = n_bins, plot = FALSE)
f_epa <- as.data.frame(bkde(x, kernel = "epa", bandwidth = h))

ggplot(x_df, aes(x)) + geom_histogram(aes(y = ..density..),
  binwidth = h, col = "black", fill = "white") +
  stat_function(fun = dnorm, args = list(mean = mu,
    sd = sigma), color = "red") + geom_line(data = f_epa,
  aes(x, y), color = "blue") + theme_minimal(base_size = 20)

```



Una librería más especializada es `np` (non-parametric).

```
library(np)

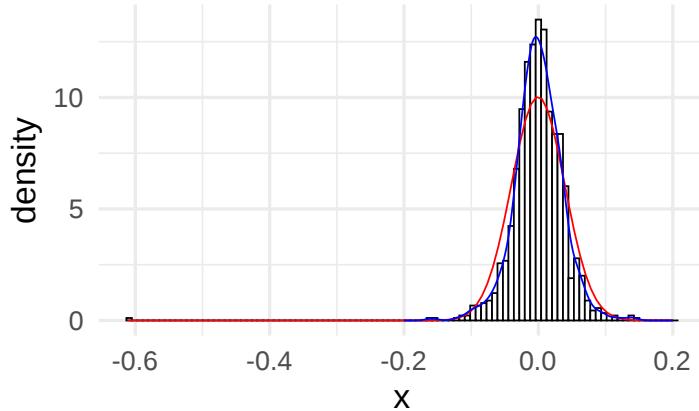
x.eval <- seq(-0.2, 0.2, length.out = 200)

h_normal_np <- npudensbw(dat = x, bwmethod = "normal-reference")

dens.ksum <- npksum(txdat = x, exdat = x.eval, bws = h_normal_np$bw)$ksum/(n *
  h_normal_np$bw[1])

dens.ksum.df <- data.frame(x = x.eval, y = dens.ksum)

ggplot(x_df, aes(x)) + geom_histogram(aes(y = ..density..),
  binwidth = h_normal_np$bw, col = "black", fill = "white") +
  stat_function(fun = dnorm, args = list(mean = mu,
    sd = sigma), color = "red") + geom_line(data = dens.ksum.df,
  aes(x, y), color = "blue") + theme_minimal(base_size = 20)
```



2.5.4. Validación cruzada

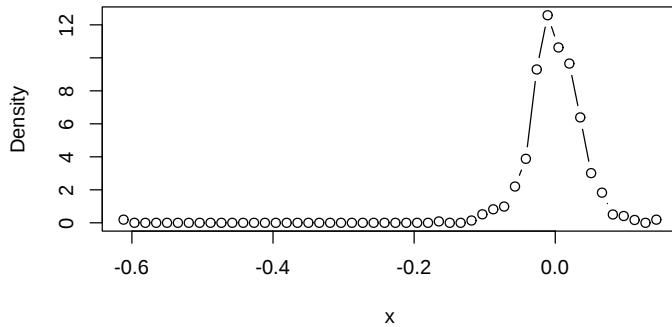
La forma que vimos en clase es la de validación cruzada por mínimos cuadrados “least-square cross validation” la cual se puede ejecutar con este comando.

```
h_cv_np_ls <- npudensbw(dat = x, bwmethod = "cv.ls",
                           ckertype = "epa", ckerorder = 2)
```

```
## Multistart 1 of 1 |Multistart 1 of 1 |Multistart 1 of 1 |Multistart 1 of 1 /Multis
```

```
dens.np <- npudens(h_cv_np_ls)

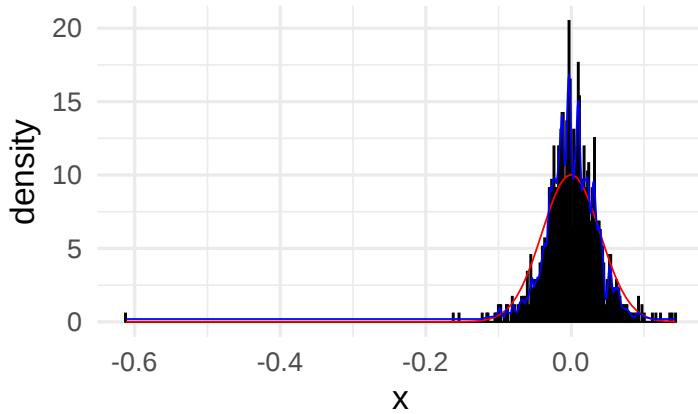
plot(dens.np, type = "b")
```



```

dens.np.df <- data.frame(x = dens.np$eval[, 1], y = dens.np$dens)

ggplot(x_df, aes(x)) + geom_histogram(aes(y = ..density..),
                                         binwidth = h_cv_np_ls$bw, col = "black", fill = "white") +
  stat_function(fun = dnorm, args = list(mean = mu,
                                         sd = sigma), color = "red") + geom_line(data = dens.np.df,
                                         aes(x, y), color = "blue") + theme_minimal(base_size = 20)
  
```



2.5.5. Temas adicionales

** Reducción del sesgo ** Como lo mencionamos en el texto, una forma de mejorar el sesgo en la estimación es suponer que la función de densidad es

más veces diferenciable.

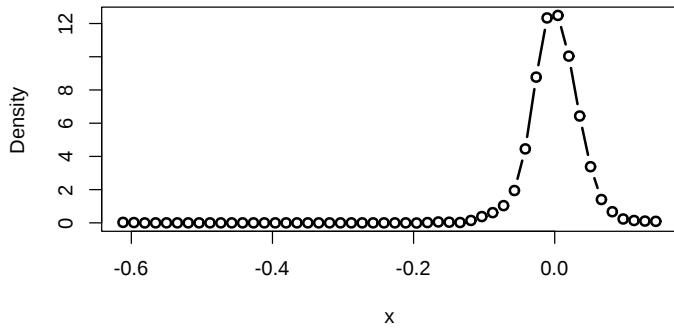
Esto se logra asumiendo que el Kernel es más veces diferenciable.

```
h_cv_np_ls <- npudensbw(dat = x, bwmethod = "cv.ls",
    ckertype = "epa", ckerorder = 4)
```

```
## Multistart 1 of 1 |Multistart 1 of 1 |Multistart 1 of 1 |Multistart 1 of 1 /Multis
```

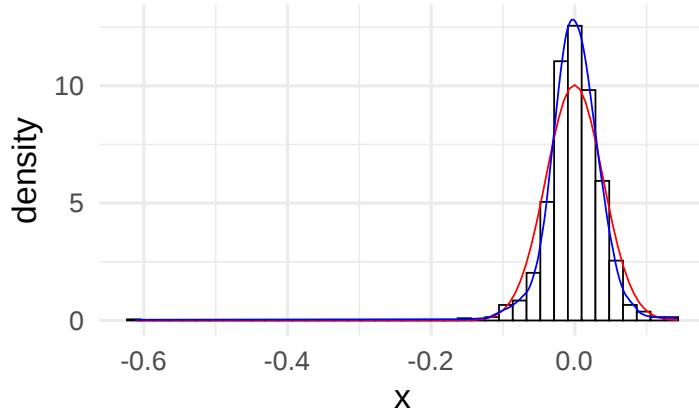
```
dens.np <- npudens(h_cv_np_ls)

plot(dens.np, type = "b", lwd = 2)
```



```
dens.np.df <- data.frame(x = dens.np$eval[, 1], y = dens.np$dens)

ggplot(x_df, aes(x)) + geom_histogram(aes(y = ..density..),
    binwidth = h_cv_np_ls$bw, col = "black", fill = "white") +
    stat_function(fun = dnorm, args = list(mean = mu,
        sd = sigma), color = "red") + geom_line(data = dens.np.df,
    aes(x, y), color = "blue") + theme_minimal(base_size = 20)
```



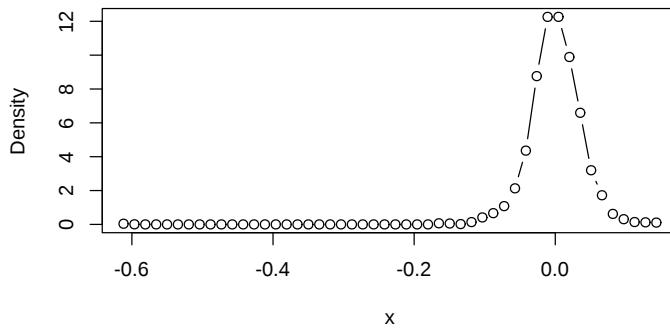
Otra forma de estimar el ancho de banda Otra forma de estimar ancho de bandas óptimos es usando máxima verosimilitud. Les dejo de tarea revisar la sección 1.1 del artículo de (Hall 1987) para entender su estructura.

```
h_cv_np_ml <- npudensbw(dat = x, bwmethod = "cv.ml",
                           ckertype = "epanechnikov")
```

```
## Multistart 1 of 1 |Multistart 1 of 1 |Multistart 1 of 1 |Multistart 1 of 1

dens.np <- npudens(h_cv_np_ml)

plot(dens.np, type = "b")
```

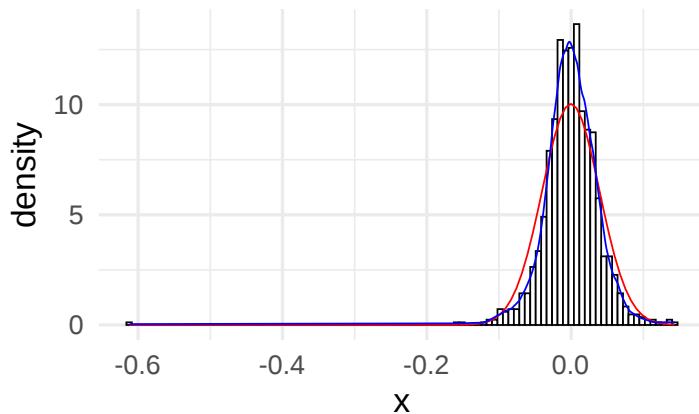


```

dens.np.df <- data.frame(x = dens.np$eval[, 1], y = dens.np$dens)

ggplot(x_df, aes(x)) + geom_histogram(aes(y = ..density..),
  binwidth = h_cv_np_ml$bw, col = "black", fill = "white") +
  stat_function(fun = dnorm, args = list(mean = mu,
    sd = sigma), color = "red") + geom_line(data = dens.np.df,
  aes(x, y), color = "blue") + theme_minimal(base_size = 20)

```



```

h_cv_np_ml <- npudensbw(dat = x, bwmethod = "cv.ml",
  ckertype = "epanechnikov", ckerorder = 4)

```

```

## Multistart 1 of 1 |Multistart 1 of 1 |Multistart 1 of 1 |Multistart 1 of 1 /Multis

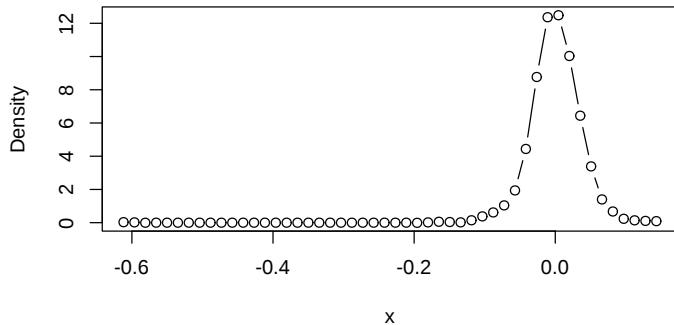
```

```

dens.np <- npudens(h_cv_np_ml)

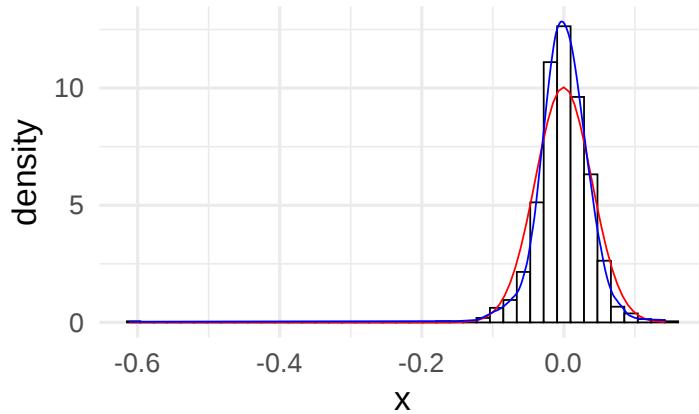
plot(dens.np, type = "b")

```



```
dens.np.df <- data.frame(x = dens.np$eval[, 1], y = dens.np$dens)

ggplot(x_df, aes(x)) + geom_histogram(aes(y = ..density..),
  binwidth = h_cv_np_ml$bw, col = "black", fill = "white") +
  stat_function(fun = dnorm, args = list(mean = mu,
    sd = sigma), color = "red") + geom_line(data = dens.np.df,
  aes(x, y), color = "blue") + theme_minimal(base_size = 20)
```



```
fani <- tibble()

for (b in seq(0.001, 0.05, length.out = 40)) {
```

```
f <- npudens(tdat = x, ckertype = "epanechnikov",
              bandwidth.compute = FALSE, bws = b)
fani <- fani %>% bind_rows(tibble(xreal = sort(x),
                                      x = f$eval$x, y = f$dens, bw = b))
}

ggplot(data = fani) + geom_line(aes(x, y), color = "blue") +
  labs(title = paste0("Ancho de banda = {closest_state}")) +
  theme_minimal(base_size = 20) + transition_states(bw) +
  view_follow()

# anim_save('manual_figure/bandwidth-animation-np.gif')
```

Ejercicio 2.5. Implementar el intervalo confianza visto en clase para estimadores de densidades por núcleos y visualizarlo de en ggplot.

Si se atreven: ¿Se podría hacer una versión animada de ese gráfico para visualizar el significado real de este el intervalo de confianza?

2.6. Ejercicios

Del libro de (**Härdle2004**) hagan los siguientes ejercicios

1. **Sección 2:** 1, 2, 3, 5, 7, 14
2. **Sección 3:** 4, 8, 10, 11, 16,

Capítulo 3

Jackknife y Bootstrap

Suponga que se quiere estimar un intervalo de confianza para la media μ desconocida de un conjunto de datos X_1, \dots, X_n que tiene distribución $\mathcal{N}(\mu, \sigma^2)$.

Primero se conoce que

$$\sqrt{n}(\hat{\mu} - \mu) \xrightarrow{\mathcal{L}} \mathcal{N}(0, \sigma^2),$$

y esto nos permite escribir el intervalo de confianza como

$$\left[\hat{\mu} - \hat{\sigma} z_{1-\frac{\alpha}{2}}, \hat{\mu} + \hat{\sigma} z_{1-\frac{\alpha}{2}} \right]$$

donde $z_{1-\frac{\alpha}{2}}$ es el cuantil $1 - \frac{\alpha}{2}$ de una normal estándar.

La expresión anterior es posible ya que el supuesto es que la distribución de $\hat{\theta}$ es normal.

Nota: . ¿Qué pasaría si este supuesto es falso o al menos no conocemos la distribución de $\hat{\theta}$?

¿Cómo podemos encontrar ese intervalo de confianza?

Nota: . Para una muestra fija, el estimador anterior $\hat{\mu}$ solamente un valor. No se conoce la distribución de $\hat{\mu}$. Lo único que se puede estimar son valores puntuales como la media, varianza, mediana, etc, pero no sabemos nada de su distribución.

3.1. Caso concreto

Suponga que tenemos la siguiente tabla de datos, que representa una muestra de tiempos y distancias de viajes en Atlanta.

Cargamos la base de la siguiente forma:

```
CommuteAtlanta <- read.csv2("data/CommuteAtlanta.csv")
```

City	Age	Distance	Time	Sex
Atlanta	19	10	15	M
Atlanta	55	45	60	M
Atlanta	48	12	45	M
Atlanta	45	4	10	F
Atlanta	48	15	30	F
Atlanta	43	33	60	M

Para este ejemplo tomaremos la variable `Time` que la llamaremos `x` para ser más breves. En este caso note que

```
x <- CommuteAtlanta$Time
```

La media es 29.11 y su varianza 429.2483968. Para efectos de lo que sigue, asignaremos la varianza a la variable T_n

```
Tn <- var(x)
```

A partir de estos dos valores, ¿Cuál sería un intervalo de confianza para la media?

Note que esta pregunta es difícil ya que no tenemos ningún tipo de información adicional.

Las dos técnicas que veremos a continuación nos permitirán extraer *información adicional* de la muestra.

Nota: . Para efectos de este capítulo, llamaremos $T_n = T(X_1, \dots, X_n)$ al estadístico formado por la muestra de los X_i 's.

3.2. Jackknife

Esta técnica fue propuesta por Quenouille 1949 y consiste en la siguiente observación.

Se puede probar que muchos de los estimadores tiene la propiedad que

$$\text{Sesgo}(T_n) = \frac{a}{n} + \frac{b}{n^2} + O\left(\frac{1}{n^3}\right) \quad (3.1)$$

para algún a and b .

Por ejemplo $\sigma^2 = \text{Var}(X_i)$ y sea $\hat{\sigma}_n^2 = n^{-1} \sum_{i=1}^n (X_i - \bar{X})^2$. Entonces,

$$\mathbb{E}(\hat{\sigma}_n^2) = \frac{n-1}{n} \sigma^2$$

por lo tanto

$$\text{Sesgo} = -\frac{\sigma^2}{n}$$

Por lo tanto en este caso $a = -\sigma^2$ y $b = 0$.

Defina $T_{(-i)}$ como el estimador T_n pero eliminando el i -ésimo término.

Es claro que en este contexto, se tiene que

$$\text{Sesgo}(T_{(-i)}) = \frac{a}{n-1} + \frac{b}{(n-1)^2} + O\left(\frac{1}{(n-1)^3}\right) \quad (3.2)$$

Ejercicio 3.1. Una forma fácil de construir los $T_{(-i)}$ es primero replicando la matriz de datos múltiple veces usando el producto de kronecker

```
n <- length(x)
jackdf <- kronecker(matrix(1, 1, n), x)
```

15	15	15	15	15	15	15	15	15	15	15
60	60	60	60	60	60	60	60	60	60	60
45	45	45	45	45	45	45	45	45	45	45
10	10	10	10	10	10	10	10	10	10	10
30	30	30	30	30	30	30	30	30	30	30
60	60	60	60	60	60	60	60	60	60	60
45	45	45	45	45	45	45	45	45	45	45
10	10	10	10	10	10	10	10	10	10	10
25	25	25	25	25	25	25	25	25	25	25
15	15	15	15	15	15	15	15	15	15	15

Y luego se elimina la diagonal

```
diag(jackdf) <- NA
```

NA	15	15	15	15	15	15	15	15	15	15
60	NA	60	60	60	60	60	60	60	60	60
45	45	NA	45	45	45	45	45	45	45	45
10	10	10	NA	10	10	10	10	10	10	10
30	30	30	30	NA	30	30	30	30	30	30
60	60	60	60	60	NA	60	60	60	60	60
45	45	45	45	45	45	NA	45	45	45	45
10	10	10	10	10	10	10	NA	10	10	10
25	25	25	25	25	25	25	25	NA	25	25
15	15	15	15	15	15	15	15	15	15	NA

Cada columna contiene toda la muestra excepto el i -ésimo elemento. Solo basta estimar la media de cada columna:

```
T_i <- apply(jackdf, 2, var, na.rm = TRUE)
```

x
429.7098
428.1905
429.6023
429.3756
430.1087
428.1905
429.6023
429.3756
430.0764
429.7098

Definamos el sesgo *jackife* como

$$b_{jack} = (n - 1)(\bar{T}_n - T_n)$$

donde

$$\bar{T}_n = \frac{1}{n} \sum_{i=1}^n T_{(-i)}$$

Ejercicio 3.2. En nuestro caso tendríamos lo siguiente:

```
(bjack <- (n - 1) * (mean(T_i) - Tn))
```

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

Es decir, que los T_i generan estimadores de T_n que contienen el mismo sesgo.

Observe que b_{jack} tiene la siguiente propiedad

$$\begin{aligned}
\mathbb{E}(b_{\text{jack}}) &= (n-1) \left(\mathbb{E}[\bar{T}_n] - \mathbb{E}[T_n] \right) \\
&= (n-1) \left(\mathbb{E}[\bar{T}_n] - \theta + \theta - \mathbb{E}[T_n] \right) \\
&= (n-1) \left(\text{Sesgo}(\bar{T}_n) - \text{Sesgo}(T_n) \right) \\
&= (n-1) \left[\left(\frac{1}{n-1} - \frac{1}{n} \right) a + \left(\frac{1}{(n-1)^2} - \frac{1}{n^2} \right) b + O\left(\frac{1}{n^3}\right) \right] \\
&= \frac{a}{n} + \frac{(2n-1)b}{n^2(n-1)} + O\left(\frac{1}{n^2}\right) \\
&= \text{Sesgo}(T_n) + O\left(\frac{1}{n^2}\right)
\end{aligned}$$

Nota: . Es decir, en general, el estimador b_{jack} aproxima correctamente $\text{Sesgo}(T_n)$ hasta con un error del n^{-2} .

Podemos usar los T_{-i} para generar muestras adicionales para estimar el parámetro θ .

En este caso defina el siguiente estimador:

$$\tilde{T}_i = nT_n - (n-1)T_{(-i)}.$$

Nota: . A \tilde{T}_i se le llaman **pseudo-valores** y representa el aporte o peso que tiene la variable X_i para estimar T_n .

Ejercicio 3.3. Usado un cálculo similar para el b_{jack} pruebe que

$$\text{Sesgo}(T_{\text{jack}}) = -\frac{b}{n(n-1)} + O\left(\frac{1}{n^2}\right) = O\left(\frac{1}{n^2}\right).$$

¿Qué conclusión se obtiene de este cálculo?

Ejercicio 3.4. Los pseudo-valores se estiman de forma directa como,

```

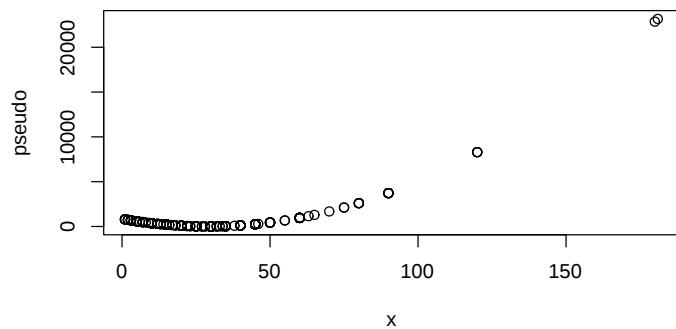
pseudo <- n * Tn - (n - 1) * T_i
pseudo[1:10]

```

```
## [1] 199.02972209 957.16225222 252.64417993 365.79679037 -0.06666345
## [6] 957.16225222 252.64417993 365.79679037 16.09799519 199.02972209
```

Lo importante acá es notar la similitud que tiene con los datos reales,

```
plot(x = x, y = pseudo)
```



Con estos pseudo-valores, es posible estimar la media y la varianza de T_n con sus respectivos estimadores:

$$T_{\text{jack}} = \frac{1}{n} \sum_{i=1}^n \tilde{T}_i$$

donde

$$v_{\text{jack}} = \frac{\sum_{i=1}^n \left(\tilde{T}_i - \frac{1}{n} \sum_{i=1}^n \tilde{T}_i \right)^2}{n(n-1)}.$$

Nota: . Sin embargo, se puede demostrar fácilmente que se pueden usar pseudovalores para construir una prueba normal de hipótesis. Dado que cada pseudovalor es independiente e idénticamente distribuido (iid), se deduce que su promedio se ajusta a una distribución normal a medida que el tamaño de la muestra aumenta. El promedio de los pseudovalores es solo T_{jack} y el valor

esperado de ese promedio, debido a la construcción a la imparcialidad del estimador, es el parámetro bajo investigación, θ . Por lo tanto, tenemos que

$$\frac{\sqrt{n} (T_{jack} - \theta)}{\sqrt{v_{jack}}} \rightarrow N(0, 1).$$

Ejercicio 3.5.

```
(Tjack <- mean(pseudo))
```

```
## [1] 429.2484
```

```
(Vjack <- var(pseudo, na.rm = TRUE))
```

```
## [1] 2701991
```

```
(sdjack <- sqrt(Vjack))
```

```
## [1] 1643.774
```

```
(z <- qnorm(1 - 0.05/2))
```

```
## [1] 1.959964
```

```
c(Tjack - z * sdjack/sqrt(n), Tjack + z * sdjack/sqrt(n))
```

```
## [1] 285.1679 573.3289
```

3.3. Bootstrap

Este método es un poco más sencillo de implementar que Jackknife y es igualmente de eficaz propuesto por Efron 1979.

Primero recordemos que estamos estimando una estadístico a partir de una muestra de modo que $T_n = g(X_1, \dots, X_n)$ donde g es cualquier función (media, varianza, quantiles, etc).

Supongamos que conocemos la distribución real de los X 's, llamada $F(x)$. Si uno quisiera estimar la varianza de X basta con hacer

$$\text{Var}_F(T_n) = \frac{\sigma^2}{n} = \frac{\int x^2 dF(x) - (\int x dF(x))^2}{n}$$

donde $\sigma^2 = \text{Var}(X)$ y el subíndice F es solo para indicar la dependencia con la distribución real.

Ahora dado que no tenemos la distribución real $F(x)$, una opción es encontrar un estimador de esta llamado \hat{F}_n .

La técnica de bootstrap se basa en extraer muchas muestras iid de la distribución \hat{F}_n de modo que se pueda conocer su varianza.

En simple pasos la técnica es

1. Seleccione $X_1^*, \dots, X_n^* \sim \hat{F}_n$
2. Estime $T_n^* = g(X_1^*, \dots, X_n^*)$
3. Repita los Pasos 1 y 2, B veces para obtener $T_{n,1}^*, \dots, T_{n,B}^*$
4. Estime

$$v_{\text{boot}} = \frac{1}{B} \sum_{b=1}^B \left(T_{n,b}^* - \frac{1}{B} \sum_{r=1}^B T_{n,r}^* \right)^2$$

Por la ley de los grandes números tenemos que

$$v_{\text{boot}} \xrightarrow{\text{a.s.}} \mathbb{V}_{\hat{F}_n}(T_n), \quad \text{si } B \rightarrow \infty. \quad (3.3)$$

además llamaremos,

$$\widehat{\text{se}}_{\text{boot}} = \sqrt{v_{\text{boot}}}$$

En pocas palabras lo que tenemos es que

$$\begin{array}{lll} \text{Mundo Real: } F & \implies X_1, \dots, X_n \implies & T_n = g(X_1, \dots, X_n) \\ \text{Mundo Bootstrap: } \hat{F}_n & \implies X_1^*, \dots, X_n^* \implies & T_n^* = g(X_1^*, \dots, X_n^*) \end{array}$$

En términos de convergencia lo que se tiene es que

$$\text{Var}_F(T_n) \xrightarrow{O(1/\sqrt{n})} \text{Var}_{\hat{F}_n}(T_n) \xrightarrow{O(1/\sqrt{B})} v_{boot}$$

Nota: . ¿Cómo extraemos una muestra de \hat{F}_n ?

Recuerden que \hat{F}_n asigna la probabilidad de $\frac{1}{n}$ a cada valor usado para construirla.

Por lo tanto, todos los puntos originales X_1, \dots, X_n tienen probabilidad $\frac{1}{n}$ de ser escogidos, que resulta ser equivalente a un muestreo con remplazo n -veces.

Así que basta cambiar el punto 1. del algoritmo mencionando anteriormente con

1. Seleccione una muestra con remplazo X_1^*, \dots, X_n^* de X_1, \dots, X_n .

Ejercicio 3.6. En este ejemplo podemos tomar $B = 1000$ y construir esa cantidad de veces nuestro estimador.

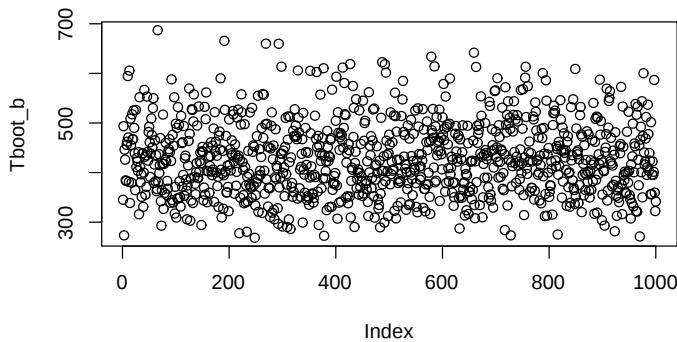
```
B <- 1000
Tboot_b <- NULL

for (b in 1:B) {
  xb <- sample(x, size = n, replace = TRUE)
  Tboot_b[b] <- var(xb)
}

Tboot_b[1:10]

## [1] 345.1819 493.5279 273.3998 446.3071 426.0340 384.2662 383.2132 455.813
```

```
plot(Tboot_b)
```



Por supuesto podemos encontrar los estadísticos usuales para esta nueva muestra

```
(Tboot <- mean(Tboot_b))
```

```
## [1] 428.066
```

```
(Vboot <- var(Tboot_b))
```

```
## [1] 5504.701
```

```
(sdboot <- sqrt(Vboot))
```

```
## [1] 74.19367
```

```
### Intervalos de confianza
```

```
\subsubsection{Intervalo Normal}
```

Este es el más sencillo y se escribe como

```
\begin{equation}
T_{\{n\}} \pm z_{\{\alpha / 2\}} \widehat{\mathbf{Se}}_{\{\mathbf{boot}\}}
\end{equation}
```

```
\BeginKnitrBlock{remark}<div class="remark">\iffalse{} <span class="remark"><math>z_{\{\alpha / 2\}} \widehat{\mathbf{Se}}_{\{\mathbf{boot}\}}</math></span></div>
\BeginKnitrBlock{exercise}<div class="exercise"><span class="exercise" id="exr`r`">c(Tn - z * sdboot, Tn + z * sdboot)
</span></div>
```

```
## [1] 283.8315 574.6653
```

```
\subsubsection{Intervalo pivotal}
```

Sea $\theta = T(F)$ y $\widehat{\theta}_n = \widehat{F}^{-1}(\widehat{F}_n)$

Sea $H(r)$ la función de distribución del pivote:

```
\[
H(r) = \mathbb{P}_{\widehat{F}}(R_n \leq r).
\]
```

Además considere $C_n^* = (a, b)$ donde

```
\[
a = \widehat{\theta}_n - H^{-1}(1 - \frac{\alpha}{2}) \quad \text{y} \quad
b = \widehat{\theta}_n - H^{-1}(1 - \frac{\alpha}{2})
```

Se sigue que

```
\begin{aligned}
& \mathbb{P}(a \leq \theta \leq b) \\
&= \mathbb{P}(\widehat{\theta}_n - H^{-1}(1 - \frac{\alpha}{2}) \leq \theta \leq \widehat{\theta}_n - H^{-1}(1 - \frac{\alpha}{2})) \\
&= H(\widehat{\theta}_n - H^{-1}(1 - \frac{\alpha}{2})) - H(\widehat{\theta}_n - H^{-1}(1 - \frac{\alpha}{2})) \\
&= H(H^{-1}(1 - \frac{\alpha}{2})) - H(H^{-1}(1 - \frac{\alpha}{2})) \\
&= 1 - \frac{\alpha}{2} - (1 - \frac{\alpha}{2}) \\
&= \frac{\alpha}{2}
\end{aligned}
```

```

&=1-\frac{\alpha}{2}-\frac{\alpha}{2}=1-\alpha
\end{align*}
\BeginKnitrBlock{remark}<div class="remark">\iffalse{} <span class="remark"><em>Nota:</em>
El problema es que este intervalo depende de  $(H)$  desconocido.
</div>\EndKnitrBlock{remark}

```

Para resolver este problema, se puede construir una versión `_bootstrap_` de (H) usando

```

\[
\widehat{H}(r)=\frac{1}{B} \sum_{b=1}^B I(\left(R_n, b\right)^{*} \leq r)
\]
donde  $\left(R_n, b\right)^{*}=\widehat{\theta}_n, b-\widehat{\theta}_n$ .

```

Sea $(r_{\beta})^{*}$ el cuantil muestral de tamaño β de $(R_n, 1)^{*}$

```

\BeginKnitrBlock{remark}<div class="remark">\iffalse{} <span class="remark"><em>Nota:</em>
\begin{equation*}
r_{\beta}^{*}=\widehat{\theta}_n-\widehat{\theta}_n
\end{equation*}</div>\EndKnitrBlock{remark}

```

Con estas observaciones

It follows that an approximate $(1-\alpha)$ confidence interval is $(C_n)=(\widehat{H}(r_{\beta}))^{*}$

```

\begin{aligned}
\widehat{a} &= \widehat{\theta}_n-\widehat{H}^{-1}\left(1-\frac{\alpha}{2}\right) \\
&= \widehat{\theta}_n-r_{1-\alpha / 2}^{*} \\
&= \widehat{\theta}_n-\widehat{\theta}_{1-\alpha / 2}^{*}+\widehat{\theta}_n \\
&=2 \widehat{\theta}_n-\widehat{\theta}_{1-\alpha / 2}^{*}-\widehat{\theta}_{\alpha / 2}^{*} \\
\widehat{b} &=\widehat{\theta}_n-\widehat{H}^{-1}\left(\frac{\alpha}{2}\right) \\
&=\widehat{\theta}_n-r_{\alpha / 2}^{*} \\
&= \widehat{\theta}_n-\widehat{\theta}_{\alpha / 2}^{*}+\widehat{\theta}_n \\
&=2 \widehat{\theta}_n-\widehat{\theta}_{\alpha / 2}^{*}-\widehat{\theta}_{\alpha / 2}^{*} \\
\end{aligned}

```

```
\BeginKnitrBlock{remark}<div class="remark">\iffalse{} <span class="remark"><math>\widehat{\theta}_n - \widehat{\theta}_{((1-\alpha/2)B)}^*</math>
\[
C_n = \left(2 \widehat{\theta}_n - \widehat{\theta}_{((1-\alpha/2)B)}^*\right)^*

```

Una mejora del intervalo anterior sería normalizar los estimadores previamente

```
\[
Z_n = \frac{T_n - \theta}{\widehat{\text{se}}_{\text{boot}}}^*.
\]
Como  $\theta$  es desconocido, entonces la versión a estimar es
\[
Z_n, b^* = \frac{T_n, b^* - T_n}{\widehat{\text{se}}_b}^*
\]
donde  $\widehat{\text{se}}_b^*$  es un estimador del error estándar de
```

```
\BeginKnitrBlock{remark}<div class="remark">\iffalse{} <span class="remark"><math>\widehat{\text{se}}_{\text{boot}}^* = \sqrt{\frac{1}{B} \sum_{b=1}^B (\widehat{\theta}_n - \widehat{\theta}_{((1-\alpha/2)B)}^*)^2}</math>
Con esto se puede obtener cantidades  $(Z_n, 1^*, \dots, Z_n, B^*)$  que
```

Sea $(z_{\alpha}^*, z_{1-\alpha/2}^*)$ del α cuantíl de $(Z_n, 1^*, \dots, Z_n, B^*)$

Define el intervalo

```
\begin{equation*}
```

```
C_n = \left(T_n - z_{1-\alpha/2}^* \widehat{\text{se}}_{\text{boot}}^*\right)^*

```

Justificado por el siguiente cálculo:

```
\begin{aligned}
& \mathbb{P}(\theta \in C_n) \leq \mathbb{P}(T_n - z_{1-\alpha/2}^* \leq \frac{\theta - \bar{\theta}}{\sqrt{Z_n}} \leq z_{\alpha/2}^*) \\
& \leq \mathbb{P}(z_{\alpha/2}^* \leq \frac{T_n - \theta}{\sqrt{Z_n}} \leq z_{1-\alpha/2}^*) \leq Z_n \leq z_{1-\alpha/2}^* \approx 1-\alpha
\end{aligned}
```

\BeginKnitrBlock{exercise}<div class="exercise">

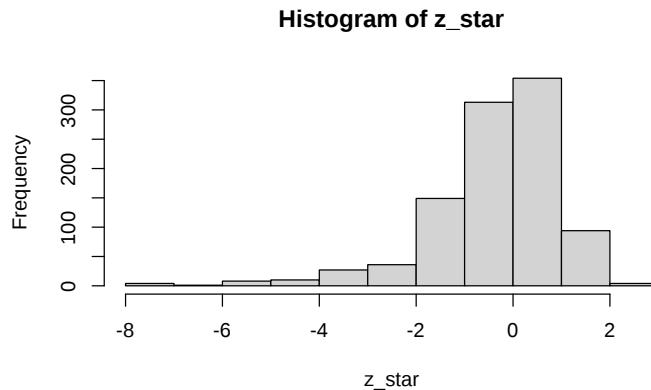
Note que para este caso tenemos que hacer bootstrap para cada estimador bootstrap calculate

```
```r
B <- 1000
Tboot_b <- NULL
Tboot_bm <- NULL
sdboot_b <- NULL

for (b in 1:B) {
 xb <- sample(x, size = n, replace = TRUE)
 Tboot_b[b] <- var(xb)
 for (m in 1:B) {
 xbm <- sample(xb, size = n, replace = TRUE)
 Tboot_bm[m] <- var(xbm)
 }
 sdboot_b[b] <- sd(Tboot_bm)
}

z_star <- (Tboot_b - Tn)/sdboot_b

hist(z_star)
```



```
c(Tn - quantile(z_star, 1 - 0.05/2) * sdboot, Tn -
 quantile(z_star, 0.05/2) * sdboot)
```

```
97.5% 2.5%
317.7259 707.0044
```

```
Resumiendo
```

Resumiendo todos los métodos de cálculo de intervalos obtenemos

```
```r
knitr::kable(data.frame(Metodo = c("Jackknife", "Bootstrap Normal",
  "Bootstrap Pivotal", "Bootstrap Pivotal Estudentizado"),
  Inferior = c(Tjack - z * sdjack/sqrt(n), Tn - z *
    sdboot, 2 * Tn - quantile(Tboot_b, 1 - 0.05/2),
  Tn - quantile(z_star, 1 - 0.05/2) * sdboot),
  Superior = c(Tjack + z * sdjack/sqrt(n), Tn + z *
    sdboot, 2 * Tn - quantile(Tboot_b, 0.05/2),
  Tn - quantile(z_star, 0.05/2) * sdboot)))
```

Metodo	Inferior	Superior
Jackknife	285.1679	573.3289
Bootstrap Normal	283.8315	574.6653
Bootstrap Pivotal	271.2827	551.4989
Bootstrap Pivotal Estudentizado	317.7259	707.0044

3.4. Ejercicios

1. Repita los ejercicios anteriores para calcular intervalos de confianza para la distancia promedio y la varianza del desplazamiento de las personas. Use los métodos de Jackknife y Bootstrap (con todos sus intervalos de confianza). Dada que la distancia es una medida que puede ser influenciada por distancias muy cortas o muy largas, se puede calcular el logaritmo de esta variable para eliminar la escala de la distancias.
2. Verifique que esta última variable se podría estimar paramétricamente con una distribución normal. Repita los cálculos anteriores tomando como cuantiles los de una normal con media 0 y varianza 1.
3. Compare los intervalos calculados y comente los resultados.
4. Del libro (Wasserman 2006) **Sección 3:** 2, 3, 7, 9, 11.

Capítulo 4

Estimación de densidades con Bayes

4.1. Introducción a la estimación Bayesiana

4.1.1. Preliminares

Recordemos que tenemos $f(\theta)$ la previa, $L(\theta)$ la verosimilitud de los datos y $f(\theta| \text{data})$ la posterior ajustada a los datos.

$$f(\theta| \text{data}) \propto f(\theta)L(\theta)$$

Además para el caso de la binomial tenemos que

$$f(y|\theta) = \theta^y(1-\theta)^{(1-y)}$$

y la distribución beta se escribe de la forma

$$\begin{aligned} f(\theta|a,b) &= \text{beta}(\theta|a,b) \\ &= \theta^{(a-1)}(1-\theta)^{(b-1)}/B(a,b) \end{aligned}$$

donde

$$B(a, b) = \int_0^1 \theta^{(a-1)} (1-\theta)^{(b-1)} d\theta.$$

Los valores de a y b controlan la forma de esta distribución

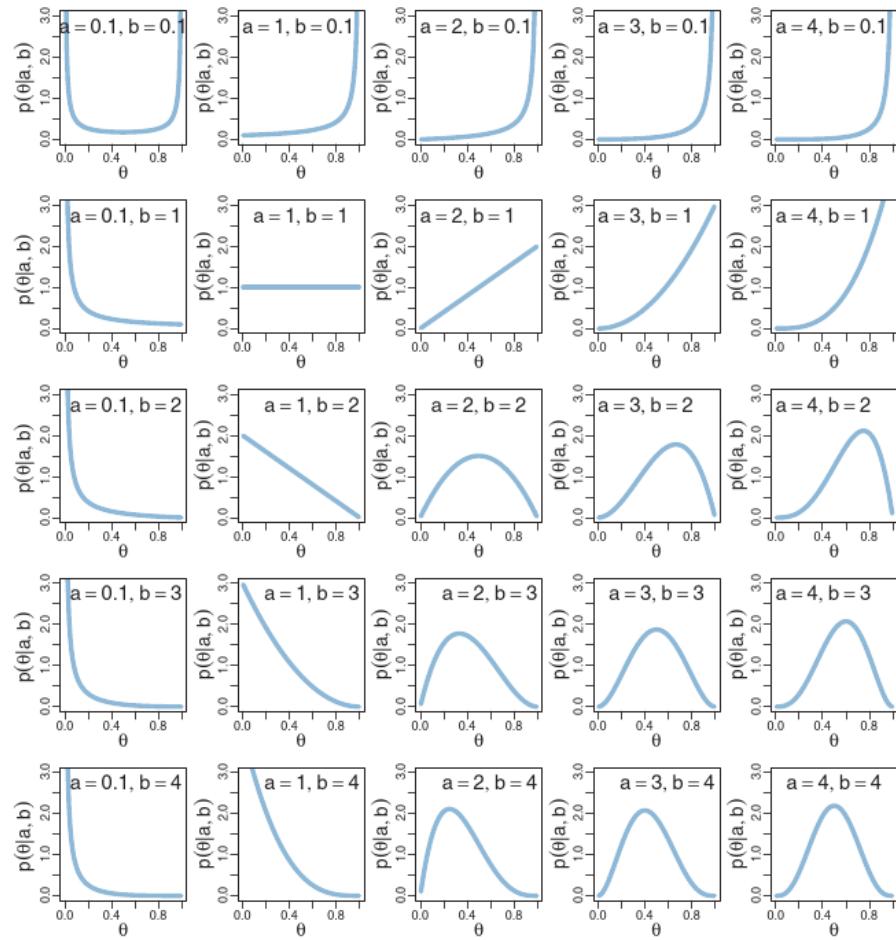


Figura 4.1: Tomado de Kruschke (2014)

Una forma alternativa es $\mu = a/(a+b)$ es la media, $\kappa = a+b$ es la concentración y $\omega = (a-1)/(a+b-2)$ es la moda de la distribución Beta, entonces se cumple que

$$\begin{aligned} a &= \mu\kappa \quad \text{y} \quad b = (1 - \mu)\kappa \\ a &= \omega(\kappa - 2) + 1 \quad \text{y} \quad b = (1 - \omega)(\kappa - 2) + 1 \quad \text{para } \kappa > 2 \end{aligned}$$

Es decir, es posible estimar a y b de κ , μ y ω

De acuerdo la combinación de estas dos distribuciones forma una familia conjugada de modo que

$$\begin{aligned} f(\theta|z, N) &= f(z, N|\theta)f(\theta)/f(z, N) \\ &= \theta^z(1 - \theta)^{(N-z)} \frac{\theta^{(a-1)}(1 - \theta)^{(b-1)}}{B(a, b)} / p(z, N) \\ &= \theta^z(1 - \theta)^{(N-z)} \theta^{(a-1)}(1 - \theta)^{(b-1)} / [B(a, b)p(z, N)] \\ &= \theta^{((z+a)-1)}(1 - \theta)^{((N-z+b)-1)} / [B(a, b)p(z, N)] \\ &= \theta^{((z+a)-1)}(1 - \theta)^{((N-z+b)-1)} / B(z + a, N - z + b) \end{aligned}$$

4.1.2. Ejemplo sencillo

Suponga que se hace una encuesta a 27 estudiantes y se encuentra que 11 dicen que duermen más de 8 horas diarias y el resto no. Nuestro objetivo es encontrar inferencias sobre la proporción p de estudiantes que duermen al menos 8 horas diarias. El modelo más adecuado es

$$f(x|p) \propto p^s(1 - p)^f$$

donde s es la cantidad de estudiantes que duermen más de 8 horas y f los que duermen menos de 8 horas.

Una primera aproximación para la previa es usar una distribución discreta. En este caso, el investigador asigna una probabilidad a cierta cantidad de horas de sueño, según su experiencia. Así, por ejemplo:

```
(p <- seq(0.05, 0.95, by = 0.1))
```

```

## [1] 0.05 0.15 0.25 0.35 0.45 0.55 0.65 0.75 0.85 0.95

(prior <- c(1, 5.2, 8, 7.2, 4.6, 2.1, 0.7, 0.1, 0,
          0))

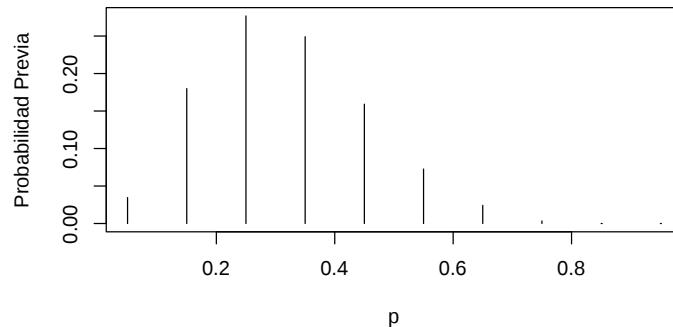
## [1] 1.0 5.2 8.0 7.2 4.6 2.1 0.7 0.1 0.0 0.0

(prior <- prior/sum(prior))

## [1] 0.034602076 0.179930796 0.276816609 0.249134948 0.159169550 0.07266436
## [7] 0.024221453 0.003460208 0.000000000 0.000000000

plot(p, prior, type = "h", ylab = "Probabilidad Previa")

```



El paquete LearnBayes tiene la función `pdisc` que estima la distribución posterior para una previa discreta binomial. Recuerde que el valor 11 representa la cantidad de estudiantes con más de 8 horas de sueño y 16 lo que no duermen esa cantidad.

```

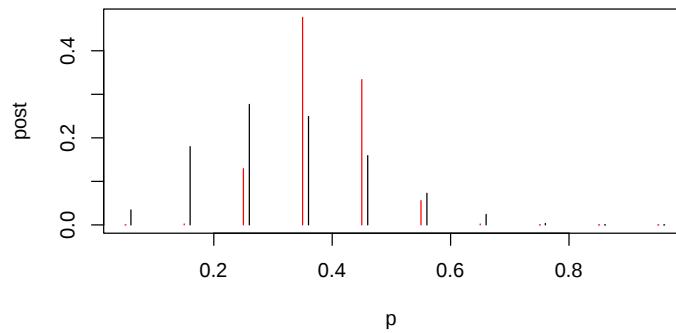
library(LearnBayes)
data <- c(11, 16)
post <- pdisc(p, prior, data)
round(cbind(p, prior, post), 2)

```

```
##          p prior post
## [1,] 0.05  0.03 0.00
## [2,] 0.15  0.18 0.00
## [3,] 0.25  0.28 0.13
## [4,] 0.35  0.25 0.48
## [5,] 0.45  0.16 0.33
## [6,] 0.55  0.07 0.06
## [7,] 0.65  0.02 0.00
## [8,] 0.75  0.00 0.00
## [9,] 0.85  0.00 0.00
## [10,] 0.95  0.00 0.00
```

Y podemos ver la diferencia entre la previa (negro) y la posterior (roja),

```
plot(p, post, type = "h", col = "red")
lines(p + 0.01, prior, type = "h")
```



¿Qué se puede deducir de estos resultados?

4.1.3. Datos reales

Continuemos el ejercicio pero esta vez usando datos reales.

Carguemos los datos `studdendata` del paquete `LearnBayes`. Esta base son preguntas que se le hicieron a un grupo de estudiantes de Bowling Green State University. Para mayor información use `?studentdata`.

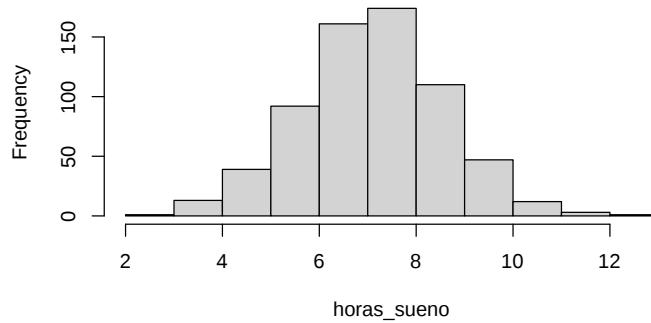
```
data("studentdata")
```

Como solo se tiene la hora de dormir y la hora de despertarse, se debe tomar la diferencia.

```
horas_sueno <- studentdata$WakeUp - studentdata$ToSleep
horas_sueno <- na.omit(horas_sueno)
summary(horas_sueno)
```

	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
##	2.500	6.500	7.500	7.385	8.500	12.500

```
hist(horas_sueno, main = "")
```



Ahora supongamos que se tiene quiere ajustar una previa continua a este modelo. Para esto usaremos una distribución Beta con parámetros a y b , de la forma

$$f(p|\alpha, \beta) \propto p^{1-a}(1-p)^{1-b}.$$

El ajuste de los parámetros de la Beta depende mucho de la información previa que se tenga del modelo. Una forma fácil de estimarlo es a través de cuantiles con los cuales se puede reescribir estos parámetros. En particular, suponga que se cree que el 50 % de las observaciones la proporción será menor que 0.3 y el 90 % será menor que 0.5.

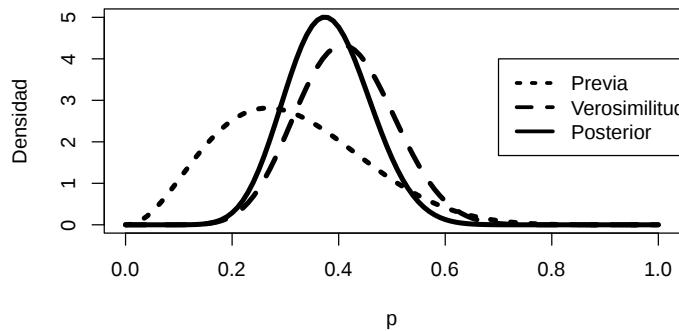
Para esto ajustaremos los siguientes parámetros

```
quantile2 <- list(p = 0.9, x = 0.5)
quantile1 <- list(p = 0.5, x = 0.3)
ab <- beta.select(quantile1, quantile2)

a <- ab[1]
b <- ab[2]
s <- 11
f <- 16
```

En este caso se obtendrá la distribución posterior Beta con parámetros $\alpha + s$ y $\beta + f$,

```
curve(dbeta(x, a + s, b + f), from = 0, to = 1, xlab = "p",
      ylab = "Densidad", lty = 1, lwd = 4)
curve(dbeta(x, s + 1, f + 1), add = TRUE, lty = 2,
      lwd = 4)
curve(dbeta(x, a, b), add = TRUE, lty = 3, lwd = 4)
legend(0.7, 4, c("Previa", "Verosimilitud", "Posterior"),
       lty = c(3, 2, 1), lwd = c(3, 3, 3))
```



En particular, si estamos interesados en $\mathbb{P}(p \geq 0.5 | \text{data})$ se puede estimar con

```
1 - pbeta(0.5, a + s, b + f)
```

```
## [1] 0.0690226
```

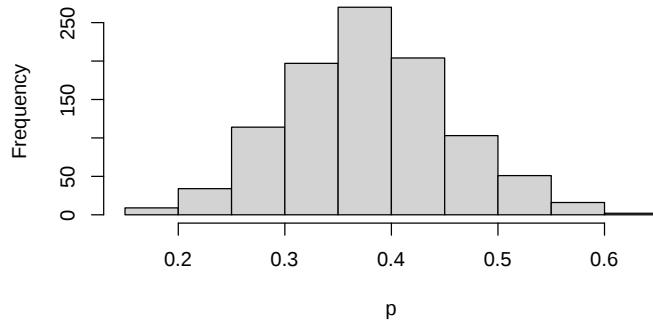
y el intervalo de confianza correspondiente a esta distribución sería

```
qbeta(c(0.05, 0.95), a + s, b + f)
```

```
## [1] 0.2555267 0.5133608
```

Otra opción para estimar este intervalo es simular 1000 veces la distribución beta y observar su comportamiento en los cuantiles

```
ps <- rbeta(1000, a + s, b + f)
hist(ps, xlab = "p", main = "")
```



La probabilidad que este valor sea mayor que 0.5 es

```
sum(ps >= 0.5) / 1000
```

```
## [1] 0.069
```

```
quantile(ps, c(0.05, 0.95))
```

```
##      5%      95%
## 0.2520157 0.5196835
```

4.2. Previa de histograma

El caso anterior funciona perfecto dada la combinación Binomial-Beta.

¿Qué pasaría si nuestra previa no está basada beta, sino que quisiéramos extraerla directamente de los datos?

El método que usaremos será el siguiente:

- Elija una cuadrícula de valores de p sobre un intervalo que cubra la densidad posterior.
- Calcule el producto de la probabilidad $L(p)$ y el $f(p)$ sobre esa grilla.

- Normalice dividiendo cada producto por la suma de los productos. En esto paso, estamos aproximando la densidad posterior por una probabilidad discreta Distribución en la grilla.
- Usando el comando `sample` de R, tome una muestra aleatoria con reemplazo de la distribución discreta.

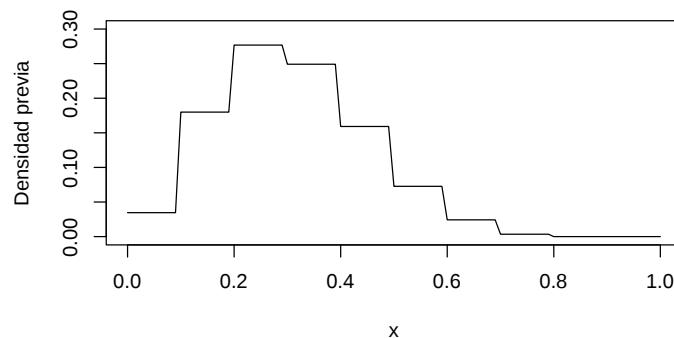
El resultado nos debe arrojar una muestra de la distribución posterior sobre la grilla

Suponga nuevamente que tenemos las mismas previas dadas al inicio del capítulo

```
midpt <- seq(0.05, 0.95, by = 0.1)
prior <- c(1, 5.2, 8, 7.2, 4.6, 2.1, 0.7, 0.1, 0, 0)
prior <- prior/sum(prior)
```

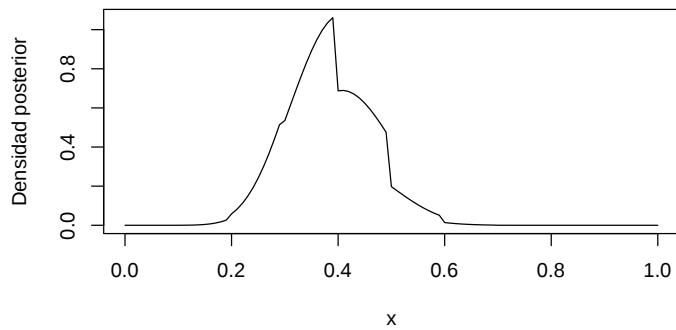
Con la función `histprior` construye los valores de p sobre una grilla.

```
curve(histprior(x, midpt, prior), from = 0, to = 1,
      ylab = "Densidad previa", ylim = c(0, 0.3))
```



Luego recordando que nuestra posterior es $\text{beta}(s + 1, f + 1)$ tenemos que

```
curve(histprior(x, midpt, prior) * dbeta(x, s + 1,
f + 1), from = 0, to = 1, ylab = "Densidad posterior")
```

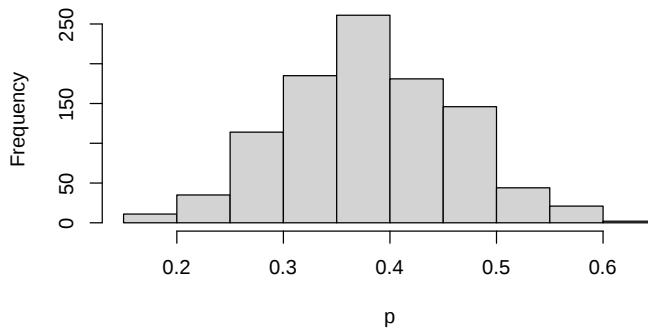


Para conseguir la distribución posterior, solo debemos de construirla para una secuencia ordenada de valores p

```
p = seq(0, 1, length = 1000)
post = histprior(p, midpt, prior) * dbeta(p, s + 1,
f + 1)
post = post/sum(post)
```

Finalmente basta con tomar el muestreo de la posterior

```
ps <- sample(p, replace = TRUE, prob = post)
hist(ps, xlab = "p", main = "")
```



4.3. Métodos Monte Carlo

4.4. Una moneda

El tratamiento clásico de la estimación de parámetros bayesiana nos dice que si tenemos una densidad previa y la “combinamos” con la verosimilitud de los datos, estos nos dará una densidad con más información. Se podría repetir el proceso varias veces para tratar de ajustar mejor la densidad posterior.

Sin embargo, se podría usar potencia de los métodos Monte Carlo para que esta búsqueda sea muy efectiva para encontrar los parámetros adecuados.

4.4.1. Ejemplo del viajero

Suponga que tenemos un viajero que quiere estar en 7 lugares distintos (suponga que están en línea recta) y la probabilidad de pasar a un lugar a otro se decide tirando una moneda no sesgada (50 % a la derecha y 50 % a la izquierda).

Este caso sería una simple caminata aleatoria sin ningún interés en particular.

Suponga además, que el viajero quiere estar más tiempo donde haya una mayor cantidad de personas P pero siguiendo ese patrón aleatorio. Entonces la forma de describir su decisión de moverse sería:

- Tira la moneda y decide si va a la izquierda o la derecha.
 1. Si el lugar nuevo tiene **MÁS** personas que el actual salta a ese lugar.
 2. Si el lugar nuevo tiene **MENOS** personas entonces el viajero tiene que decidir si se queda o se mueve. | calcula la probabilidad de moverse como $p_{moverse} = P_{nuevo}/P_{actual}$.

Tira un número aleatorio entre 0 y 1 r

 1. Si $p_{moverse} > r$ entonces se mueve.
 2. Sino, se queda donde está.

```
P <- 1:7

pos_actual <- sample(P, 1)
pos_nueva <- pos_actual

n_pasos <- 50000
trayectoria <- numeric(n_pasos)

trayectoria[1] <- pos_actual

for (k in 2:n_pasos) {
  # Tira la moneda para decidir

  moneda <- rbinom(1, 1, 0.5)
  # moneda es 0 o 1
  pos_nueva <- pos_actual
  if (moneda == 1 & (pos_actual + 1) <= 7) {
    pos_nueva = pos_actual + 1
  } else if (moneda == 0 & (pos_actual - 1) >= 1) {
    pos_nueva <- pos_actual - 1
  }

  p_moverse <- min(pos_nueva/pos_actual, 1)

  hay_movimiento <- 1 - p_moverse <= runif(1)
```

```

if (hay_movimiento) {
    pos_actual <- pos_nueva
}

trayectoria[k] <- pos_nueva
}

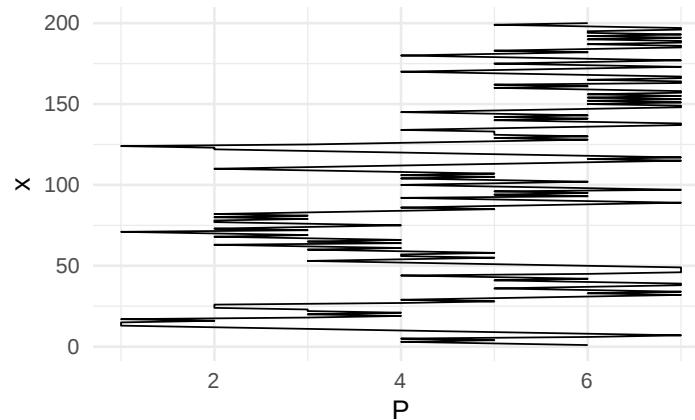
```

```

df <- data.frame(x = 1:n_pasos, P = trayectoria)

ggplot(df[1:200, ]) + geom_line(aes(x, P)) + coord_flip() +
    theme_minimal(base_size = 16)

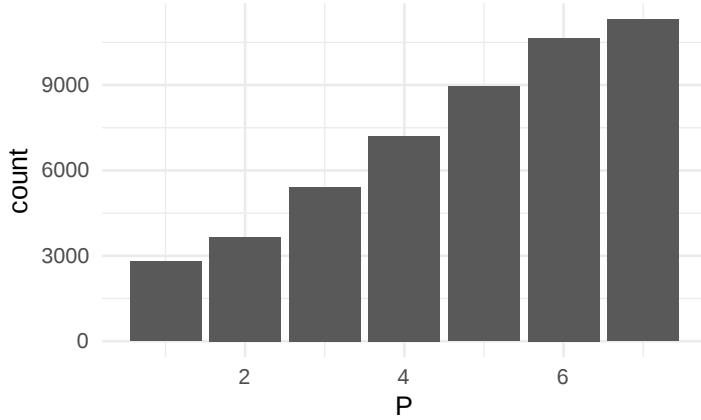
```



```

ggplot(df) + geom_histogram(aes(P), stat = "count") +
    theme_minimal(base_size = 16)

```



```
mean(trayectoria)
```

```
## [1] 4.86008
```

```
sd(trayectoria)
```

```
## [1] 1.798412
```

4.4.2. Cadenas de Markov

Recuerde que estamos buscando el «camino» que el viajero tomará para pasar la mayor parte del tiempo de en los lugares más poblados (con mayor θ).

$$\theta_1 \curvearrowright \theta_2 \curvearrowright \dots \curvearrowright \theta_{50000}.$$

Denotamos $\theta_1 \rightarrow \theta_2$ si el viajero pasó de θ_1 hacia θ_2 .

Entonces

- $\mathbb{P}(\theta \rightarrow \theta + 1) = 0,5 \min\left(\frac{P(\theta+1)}{P(\theta)}, 1\right)$
- $\mathbb{P}(\theta + 1 \rightarrow \theta) = 0,5 \min\left(\frac{P(\theta)}{P(\theta+1)}, 1\right)$

Entonces la razón entre estas dos probabilidades es

$$\begin{aligned} \frac{\mathbb{P}(\theta \rightarrow \theta + 1)}{\mathbb{P}(\theta + 1 \rightarrow \theta)} &= \frac{0,5 \min(P(\theta + 1)/P(\theta), 1)}{0,5 \min(P(\theta)/P(\theta + 1), 1)} \\ &= \begin{cases} \frac{P(\theta+1)}{P(\theta)} & \text{si } P(\theta + 1) > P(\theta) \\ \frac{P(\theta+1)}{P(\theta)} & \text{si } P(\theta + 1) < P(\theta) \end{cases} \\ &= \frac{P(\theta + 1)}{P(\theta)}. \end{aligned}$$

Es decir que la razón de las probabilidades es equivalente a la razón entre las proporción de las poblaciones. Por lo tanto la mayoría de las veces se estará en los lugares con mayor población.

Esta cadena se puede escribir usando una matriz de transición de la forma

$$T = \begin{pmatrix} \ddots & \mathbb{P}(\theta - 2 \rightarrow \theta - 1) & 0 & 0 & 0 \\ \ddots & \mathbb{P}(\theta - 1 \rightarrow \theta - 1) & \mathbb{P}(\theta - 1 \rightarrow \theta) & 0 & 0 \\ 0 & \mathbb{P}(\theta \rightarrow \theta - 1) & \mathbb{P}(\theta \rightarrow \theta) & \mathbb{P}(\theta \rightarrow \theta + 1) & 0 \\ 0 & 0 & \mathbb{P}(\theta + 1 \rightarrow \theta) & \mathbb{P}(\theta + 1 \rightarrow \theta + 1) & \ddots \\ 0 & 0 & 0 & \mathbb{P}(\theta + 2 \rightarrow \theta + 1) & \ddots \end{pmatrix}$$

La matriz T tiene las propiedades

1. Existencia de una única distribución estacionaria (llamada f más adelante).
2. Es ergódica, i.e., es aperiódica y positiva recurrente. Recuerde que una cadena de markov es ergódica si siempre se puede pasar de un estado a otro (no necesariamente en 1 paso). Otra forma de verlo es que la para alguna potencia de T todos los sus elementos serán positivos estrictos.

4.4.3. El algoritmo de Metropolis-Hastings

El ejemplo anterior era bastante sencillo pero demuestra que se puede encontrar el mejor estimador posible simplemente ejecutando una y otra vez maximizando la estadía en los lugares más poblados.

En este ejemplo la función a maximizar es la cantidad de personas $P(\theta) = \theta$, pero en general nuestro objetivo será maximizar la distribución posterior $f(\theta | \text{datos})$.

En palabras simples el algoritmo de Metropoli Hasting es

1. Simule un valor θ^* de una densidad de propuesta $p(\theta^* | \theta^{t-1})$
2. Estime la razón

$$R = \frac{f(\theta^*) L(\theta^{t-1} | \theta^*)}{f(\theta^{t-1}) L(\theta^* | \theta^{t-1})}$$

3. Estima la probabilidad de aceptación $p_{\text{moverse}} = \min\{R, 1\}$.
4. Tome θ^t tal que $\theta^t = \theta^*$ con probabilidad p_{moverse} ; en otro caso $\theta^t = \theta^{t-1}$

El algoritmo de Metropolis-Hastings se puede construir de muchas formas, dependiendo de la densidad de proposición

Si esta es independiente de las elecciones anteriores entonces,

$$L(\theta^* | \theta^{t-1}) = L(\theta^*)$$

Otras formas es escoger

$$L(\theta^* | \theta^{t-1}) = h(\theta^* - \theta^{t-1})$$

donde h es simétrica alrededor del origen. En este tipo de cadenas, la razón R tiene la forma

$$R = \frac{f(\theta^*)}{f(\theta^{t-1})}$$

Una última opción es tomar

$$\theta^* = \theta^{t-1} + Z$$

donde Z es una normal centrada con cierta estructura de varianza.

4.4.4. ¿Por qué el algoritmo de Metropolis Hasting funciona?

$$\mathbb{P}(\theta_* | \theta^{(t)}) = L(\theta_* | \theta^{(t)}) \cdot \min \left\{ 1, \frac{f(\theta_*) L(\theta^{(t)} | \theta_*)}{f(\theta^{(t)}) L(\theta_* | \theta^{(t)})} \right\} \quad (4.1)$$

Si se comienza en $f(\theta^{(t)})$ entonces

$$\begin{aligned} & f(\theta^{(t)}) \mathbb{P}(\theta_* | \theta^{(t)}) \\ &= f(\theta^{(t)}) L(\theta_* | \theta^{(t)}) \min \left\{ 1, \frac{f(\theta_*) L(\theta^{(t)} | \theta_*)}{f(\theta^{(t)}) L(\theta_* | \theta^{(t)})} \right\} \\ &= \min \left\{ f(\theta^{(t)}) L(\theta_* | \theta^{(t)}), f(\theta_*) L(\theta^{(t)} | \theta_*) \right\} \\ &= f(\theta_*) L(\theta^{(t)} | \theta_*) \min \left\{ \frac{f(\theta^{(t)}) L(\theta_* | \theta^{(t)})}{f(\theta_*) L(\theta^{(t)} | \theta_*)}, 1 \right\} \\ &= f(\theta_*) \mathbb{P}(\theta^{(t)} | \theta_*) \end{aligned} \quad (4.2)$$

Asumiendo que existe una cantidad finita de estados $\theta_1, \dots, \theta_M$, entonces.

$$\begin{aligned} f(\theta_j) &= \underbrace{\sum_{i=1}^M f(\theta_i) \mathbb{P}(\theta_j | \theta_i)}_{\text{Probabilidad total}} = \sum_{i=1}^M f(\theta_j) \mathbb{P}(\theta_i | \theta_j) \\ & f(\boldsymbol{\theta})^\top T = f(\boldsymbol{\theta}) \end{aligned} \quad (4.3)$$

Cual indica que no importa donde empecemos siempre llegaremos a la densidad estacionaria f .

https://www.ece.iastate.edu/~namrata/EE527_Spring08/l4c.pdf#page=32

4.4.5. Extensión al caso del viajero

Retomemos el ejemplo del viajero. Supongamos que ahora existen una cantidad infinita de lugares a los que puede ir y que la población de cada isla es proporcional a la densidad posterior. Además, el viajero podría saltar a

cualquier isla que quisiera y su probabilidad de salto cae de forma continua en el intervalo $[0, 1]$.

Para hacer este ejemplo concreto, el viajero no conoce cuál es su probabilidad de salto θ pero sabe que ha tirado la moneda N veces y observado z éxitos. Por lo tanto tendremos una verosimilitud de $L(z, N|\theta) = \theta^z(1 - \theta)^{(N-z)}$.

La previa será dada por $f(\theta) = \text{beta}(\theta|a, b)$.

Los saltos serán gobernados por una normal centrada con media σ de modo que $\Delta\theta \sim \mathcal{N}(0, \sigma^2)$.

Entonces el algoritmo de Metropolis Hasting se puede reformular como

1. Simule un valor de salto $\Delta\theta \sim \mathcal{N}(0, \sigma^2)$ y denote $\theta^t = \theta^t + \Delta\theta$.
2. Probabilidad de aceptación p_{moverse}

$$\begin{aligned} p_{\text{moverse}} &= \min \left(1, \frac{P(\theta_*)}{P(\theta_{t-1})} \right) \\ &= \min \left(1, \frac{p(D|\theta_*) p(\theta_*)}{p(D|\theta_{t-1}) p(\theta_{t-1})} \right) \\ &= \min \left(1, \frac{\text{Bernoulli}(z, N|\theta_*) \text{beta}(\theta_*|a, b)}{\text{Bernoulli}(z, N|\theta_{t-1}) \text{beta}(\theta_{t-1}|a, b)} \right) \\ &= \min \left(1, \frac{\theta_*^z (1 - \theta_*)^{(N-z)} \theta_* (1 - \theta_*)^{(b-1)} / B(a, b)}{\theta_{t-1}^z (1 - \theta_{t-1})^{(N-z)} \theta_{t-1}^{(a-1)} (1 - \theta_{t-1})^{(b-1)} / B(a, b)} \right) \end{aligned}$$

3. Tome θ_t tal que $\theta_t = \theta_*$ con probabilidad p_{moverse} ; en otro caso $\theta_t = \theta_{t-1}$

En el ejemplo del viajero queremos ver la probabilidad θ de que salte al siguiente destino. Tomemos $\sigma = 0,2$ y supongamos que se ha visto que el viajero de $N = 20$ y $z = 14$ éxitos. Por cuestiones de practicidad se tomará $\theta_0 = 0,1$.

```
# Carga de datos observados
datos_observados <- c(rep(0, 6), rep(1, 14))
```

```

# Función de verosimilitud Binomial
verosimilitud <- function(theta, data) {
  z <- sum(data)
  N <- length(data)
  pDatosDadoTheta <- thetaz * (1 - theta)(N - z)
  # Es para asegurarse que los datos caigan en [0,1].
  pDatosDadoTheta[theta > 1 | theta < 0] <- 0
  return(pDatosDadoTheta)
}

# densidad previa
previa <- function(theta) {
  pTheta <- dbeta(theta, 1, 1)
  # Es para asegurarse que los datos caigan en [0,1].
  pTheta[theta > 1 | theta < 0] <- 0
  return(pTheta)
}

# densidad posterior
posterior <- function(theta, data) {
  posterior <- verosimilitud(theta, data) * previa(theta)
  return(posterior)
}

n_pasos <- 50000

trayectoria <- rep(0, n_pasos)

# Valor inicial
trayectoria[1] <- 0.01

n_aceptados <- 0
n_rechazados <- 0

sigma <- 0.2

```

```

for (t in 2:(n_pasos - 1)) {
  pos_actual <- trayectoria[t]

  salto_propuesto <- rnorm(1, mean = 0, sd = sigma)

  proba_aceptacion <- min(1, posterior(pos_actual +
    salto_propuesto, datos_observados)/posterior(pos_actual,
    datos_observados))

  # Aceptamos el salto?
  if (runif(1) < proba_aceptacion) {
    # Aceptados
    trayectoria[t + 1] <- pos_actual + salto_propuesto
    n_aceptados <- n_aceptados + 1
  } else {
    # Rechazos
    trayectoria[t + 1] <- pos_actual
    n_rechazados <- n_rechazados + 1
  }
}

```

Obtenemos una tasa de aceptación del 49.4 y tasa de rechazo del 50.59

Podemos desechar los primeros 500 pasos (por ejemplo) del proceso ya que estos son de «calentamiento». De esta forma podremos estimar la media y la varianza de las trayectoria.

```
mean(trayectoria[500:n_pasos])
```

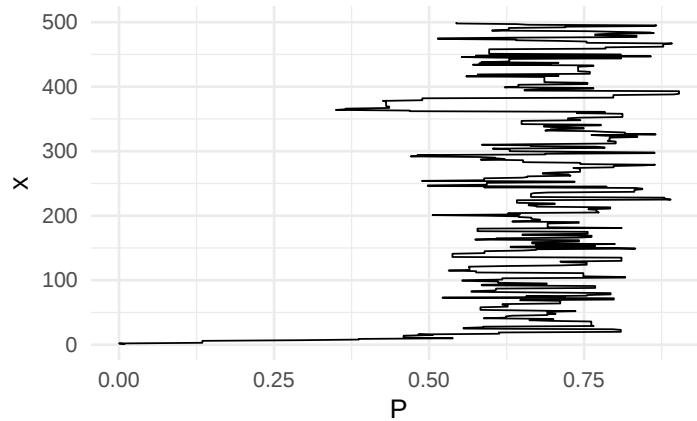
```
## [1] 0.6808914
```

```
sd(trayectoria[500:n_pasos])
```

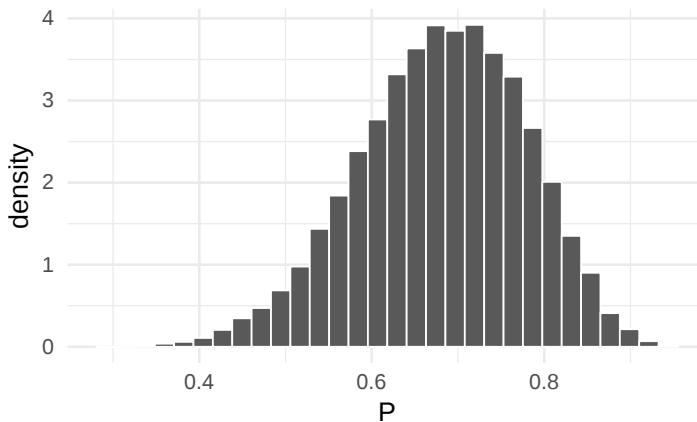
```
## [1] 0.09721105
```

```
df <- data.frame(x = 1:n_pasos, P = trayectoria)

ggplot(df[1:500, ]) + geom_line(aes(x, P), size = 0.5) +
  coord_flip() + theme_minimal(base_size = 16)
```



```
ggplot(df[500:n_pasos, ]) + geom_histogram(aes(P, y = ..density..),
  color = "white") + theme_minimal(base_size = 16)
```



4.5. Dos monedas

Un problema con el algoritmo de Metropolis-Hastings (M-H) es que solo funciona para la estimación de un solo parámetro.

El muestreo de Gibbs está pensado en el caso de la estimación de muchos parámetros de forma bastante ordenada.

Supongamos que tenemos dos monedas y queremos ver la proporción de escudos generados entre las dos monedas:

Tenemos:

- Parámetros: θ_1 y θ_2 .
- Datos: N_1 tiradas de la moneda 1 y N_2 tiradas de la moneda 2. (cada una tuvo z_1 y z_2 éxitos).

3. Verosimilitud: Bernoulli.

$$y_1^i \sim \text{Bernoulli}(\theta_1) \quad y_2^i \sim \text{Bernoulli}(\theta_2)$$

4. Previa: Beta independiente para cada θ .

$$\theta_1 \sim \text{Beta}(a_1, b_1) \quad \theta_2 \sim \text{Beta}(a_2, b_2)$$

La distribución posterior se puede escribir como

$$\begin{aligned} f(\theta_1, \theta_2 | D) &= f(D | \theta_1, \theta_2) \frac{f(\theta_1, \theta_2)}{f(D)} \\ &= \theta_1^{z_1} (1 - \theta_1)^{N_1 - z_1} \theta_2^{z_2} (1 - \theta_2)^{N_2 - z_2} \frac{f(\theta_1, \theta_2)}{f(D)} \\ &= \frac{\theta_1^{z_1} (1 - \theta_1)^{N_1 - z_1} \theta_2^{z_2} (1 - \theta_2)^{N_2 - z_2} \theta_1^{a_1 - 1} (1 - \theta_1)^{b_1 - 1} \theta_2^{a_2 - 1} (1 - \theta_2)^{b_2 - 1}}{f(D) B(a_1, b_1) B(a_2, b_2)} \\ &= \frac{\theta_1^{z_1 + a_1 - 1} (1 - \theta_1)^{N_1 - z_1 + b_1 - 1} \theta_2^{z_2 + a_2 - 1} (1 - \theta_2)^{N_2 - z_2 + b_2 - 1}}{f(D) B(a_1, b_1) B(a_2, b_2)} \end{aligned}$$

Entonces la distribución posterior de (θ_1, θ_2) son dos distribuciones independientes Betas: Beta $(z_1 + a, N_1 - z_1 + b_1)$ y Beta $(z_2 + a, N_2 - z_2 + b_2)$

Tratemos de encontrar los parámetros para la distribución posterior usando un algoritmo de Metropolis-Hasting. [Función tomada de Kruschke-Notes](#)

```
metro_2coins <- function(  
  z1, n1,                      # z = successes, n = trials  
  z2, n2,                      # z = successes, n = trials  
  size = c(0.1, 0.1),           # sds of jump distribution  
  start = c(0.5, 0.5),          # value of thetas to start at  
  num_steps = 5e4,              # number of steps to run the algorithm  
  prior1 = dbeta,               # function describing prior  
  prior2 = dbeta,               # function describing prior  
  args1 = list(),              # additional args for prior1  
  args2 = list()               # additional args for prior2  
) {  
  
  theta1            <- rep(NA, num_steps)    # trick to pre-allocate memory  
  theta2            <- rep(NA, num_steps)    # trick to pre-allocate memory  
  proposed_theta1   <- rep(NA, num_steps)    # trick to pre-allocate memory  
  proposed_theta2   <- rep(NA, num_steps)    # trick to pre-allocate memory  
  move              <- rep(NA, num_steps)    # trick to pre-allocate memory  
  theta1[1]         <- start[1]  
  theta2[1]         <- start[2]  
  
  size1 <- size[1]  
  size2 <- size[2]  
  
  for (i in 1:(num_steps-1)) {  
    # head to new "island"  
    proposed_theta1[i + 1] <- rnorm(1, theta1[i], size1)  
    proposed_theta2[i + 1] <- rnorm(1, theta2[i], size2)  
  
    if (proposed_theta1[i + 1] <= 0 ||  
        proposed_theta1[i + 1] >= 1 ||  
        proposed_theta2[i + 1] <= 0 ||  
        proposed_theta2[i + 1] >= 1) {  
      proposed_posterior <- 0    # because prior is 0  
    } else {  
      current_prior <-
```

```

do.call(prior1, c(list(theta1[i]), args1)) *
do.call(prior2, c(list(theta2[i]), args2))
current_likelihood <-
  dbinom(z1, n1, theta1[i]) *
  dbinom(z2, n2, theta2[i])
current_posterior <- current_prior * current_likelihood

proposed_prior <-
  do.call(prior1, c(list(proposed_theta1[i+1]), args1)) *
  do.call(prior2, c(list(proposed_theta2[i+1]), args2))
proposed_likelihood <-
  dbinom(z1, n1, proposed_theta1[i+1]) *
  dbinom(z2, n2, proposed_theta2[i+1])
proposed_posterior <- proposed_prior * proposed_likelihood
}

prob_move <- proposed_posterior / current_posterior

# sometimes we "sail back"
if (runif(1) > prob_move) { # sail back
  move[i + 1] <- FALSE
  theta1[i + 1] <- theta1[i]
  theta2[i + 1] <- theta2[i]
} else { # stay
  move[i + 1] <- TRUE
  theta1[i + 1] <- proposed_theta1[i + 1]
  theta2[i + 1] <- proposed_theta2[i + 1]
}
}

tibble(
  step = 1:num_steps,
  theta1 = theta1,
  theta2 = theta2,
  proposed_theta1 = proposed_theta1,
  proposed_theta2 = proposed_theta2,
  move = move,
  size1 = size1,

```

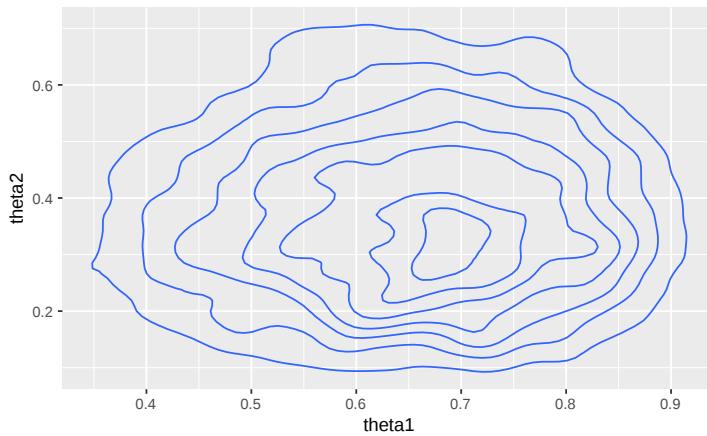
```

    size2 = size2
)
}

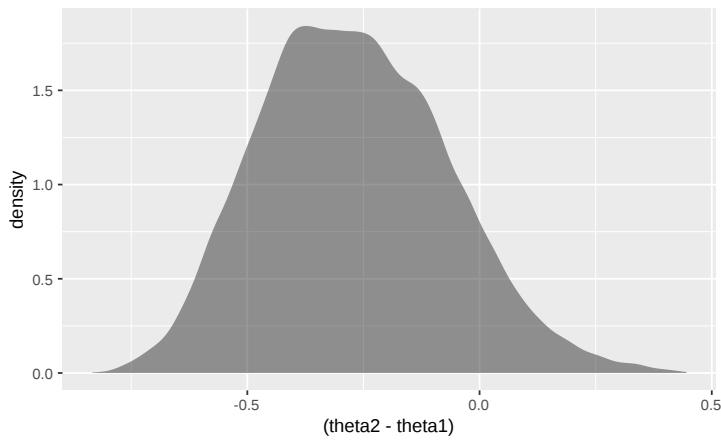
Metro_2coinsA <- metro_2coins(z1 = 6, n1 = 8, z2 = 2,
                                n2 = 7, size = c(0.02, 0.02), args1 = list(shape1 = 2,
                                shape2 = 2), args2 = list(shape1 = 2, shape2 = 2))

Metro_2coinsA %>% gf_density2d(theta2 ~ theta1)

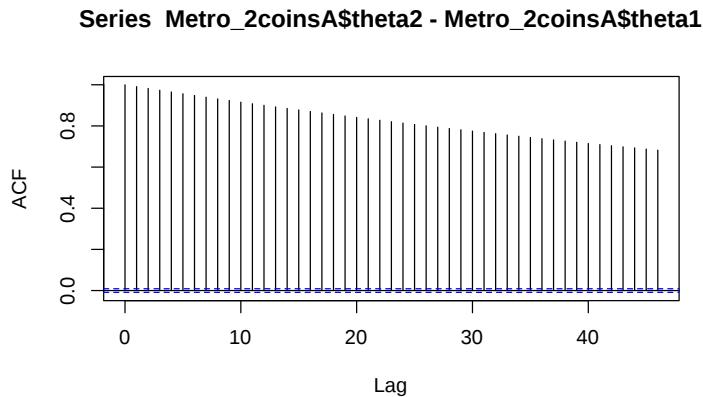
```



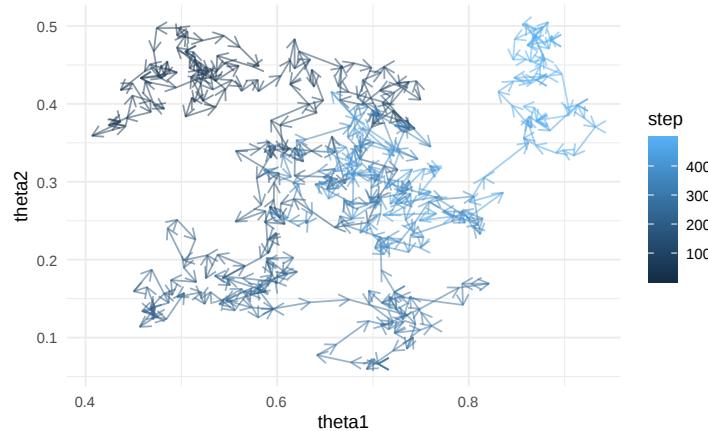
```
Metro_2coinsA %>% gf_density(~(theta2 - theta1))
```



```
acf(Metro_2coinsA$theta2 - Metro_2coinsA$theta1)
```



```
Metro_2coinsA %>% filter(step < 500) %>% gf_path(theta2 ~ theta1, color = ~step, alpha = 0.5, arrow = arrow(type = "open", angle = 30, length = unit(0.1, "inches"))) + theme_minimal()
```



```
library(gganimate)
Metro_2coinsAplot <- Metro_2coinsA %>% filter(step < 500) %>% gf_path(theta2 ~ theta1, color = ~step, alpha = 0.5, arrow = arrow(type = "open", angle = 30)) +
```

```

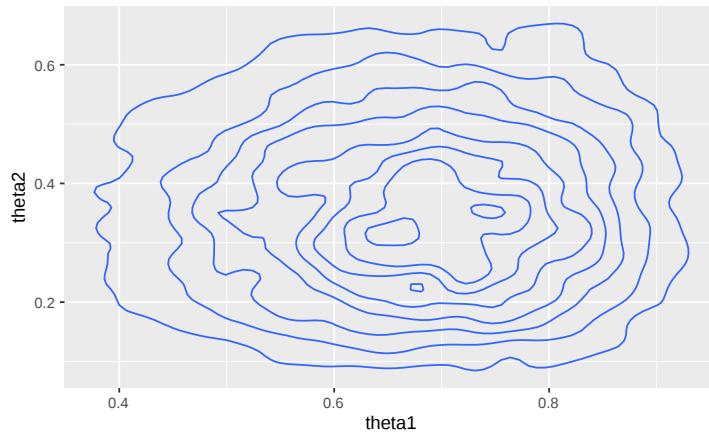
theme_minimal() + transition_reveal(step)

animate(Metro_2coinsAplot, fps = 1)

Metro_2coinsB <- metro_2coins(z1 = 6, n1 = 8, z2 = 2,
                                n2 = 7, size = c(0.2, 0.2), args1 = list(shape1 = 2,
                                shape2 = 2), args2 = list(shape1 = 2, shape2 = 2))

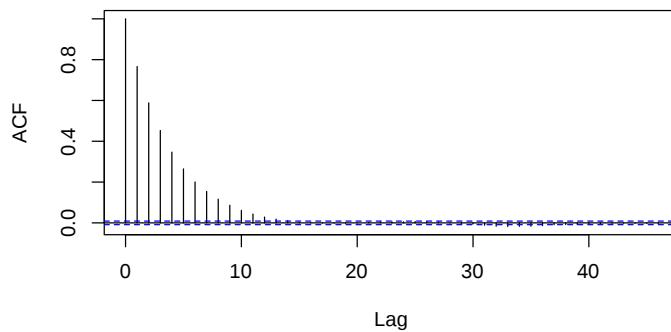
Metro_2coinsB %>% gf_density2d(theta2 ~ theta1)

```

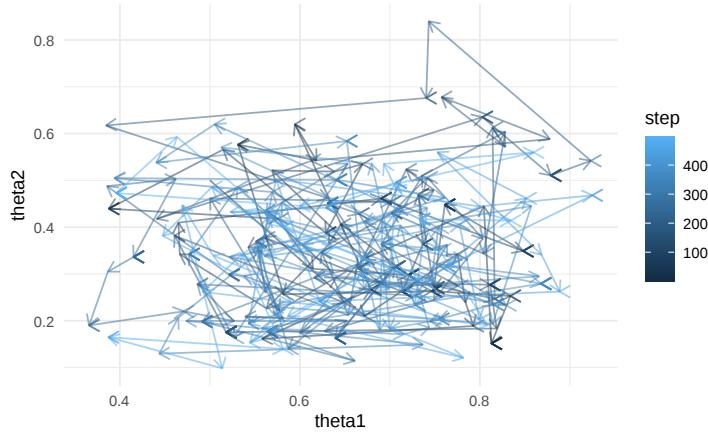


```
acf(Metro_2coinsB$theta2 - Metro_2coinsB$theta1)
```

Series Metro_2coinsB\$theta2 - Metro_2coinsB\$theta1



```
Metro_2coinsB %>% filter(step < 500) %>% gf_path(theta2 ~
  theta1, color = ~step, alpha = 0.5, arrow = arrow(type = "open",
  angle = 30, length = unit(0.1, "inches")))) + theme_minimal()
```



```
Metro_2coinsBplot <- Metro_2coinsA %>% filter(step <
  500) %>% gf_path(theta2 ~ theta1, color = ~step,
  alpha = 0.5, arrow = arrow(type = "open", angle = 30)) +
  theme_minimal() + transition_reveal(step)

animate(Metro_2coinsBplot, fps = 1)
```

4.5.1. Muestreo de Gibbs

Para este ejemplo (θ_1, θ_2) , entonces la forma de escoger la posteriores en cada paso sería de la forma:

1. Tome al azar un $\boldsymbol{\theta}^0 = (\theta_1^0, \theta_2^0)$.
2. Escoja θ_1^1 a partir de la distribución $f(\theta_1 | \theta_1 = \theta_1^0, \theta_2 = \theta_2^0, \mathbf{X})$.
3. Escoja θ_2^1 a partir de la distribución $f(\theta_2 | \theta_1 = \theta_1^1, \theta_2 = \theta_2^0, \mathbf{X})$.

Esto completa un ciclo del muestreo. Cada ciclo genera nuevos $\boldsymbol{\theta}^i = (\theta_1^i, \theta_2^i)$ hasta que el proceso converja.

Nota: . En realidad el muestreo de Gibbs se basa en el algoritmo de M-H, con la diferencia que la elección de los parámetros se escogen teniendo en cuenta los datos y fijando los otros parámetros. Es decir,

$$\begin{aligned} & [\theta_1 | \theta_2, \dots, \theta_M, \text{datos}] \\ & [\theta_2 | \theta_1, \theta_3, \dots, \theta_M, \text{datos}] \\ & [\theta_M | \theta_1, \dots, \theta_{M-1}, \text{datos}] \end{aligned}$$

El tratamiento teórico puede ser consultado https://www.ece.iastate.edu/~namrata/EE527_Spring08/l4c.pdf#page=16

$$\begin{aligned} f(\theta_1 | \theta_2, D) &= \frac{f(\theta_1, \theta_2 | D)}{f(\theta_2 | D)} \\ &= \frac{f(\theta_1, \theta_2 | D)}{\int f(\theta_1, \theta_2 | D) d\theta_1} \\ &= \frac{\text{dbeta}(\theta_1, z_1 + a_1, N_1 - z_1 + b_1) \cdot \text{dbeta}(\theta_2, z_2 + a_2, N_2 - z_2 + b_2)}{\int \text{dbeta}(\theta_1, z_1 + a_1, N_1 - z_1 + b_1) \cdot \text{dbeta}(\theta_2 | z_2 + a_2, N_2 - z_2 + b_2) d\theta_1} \\ &= \frac{\text{dbeta}(\theta_1, z_1 + a_1, N_1 - z_1 + b_1) \cdot \text{dbeta}(\theta_2, z_2 + a_2, N_2 - z_2 + b_2)}{\text{dbeta}(\theta_2 | z_2 + a_2, N_2 - z_2 + b_2) \int \text{dbeta}(\theta_1, z_1 + a_1, N_1 - z_1 + b_1) d\theta_1} \\ &= \frac{\text{dbeta}(\theta_1, z_1 + a_1, N_1 - z_1 + b_1)}{\int \text{dbeta}(\theta_1, z_1 + a_1, N_1 - z_1 + b_1) d\theta_1} \\ &= \text{dbeta}(\theta_1, z_1 + a_1, N_1 - z_1 + b_1) \end{aligned}$$

```
gibbs_2coins <- function(
  z1, n1,                      # z = successes, n = trials
  z2, n2,                      # z = successes, n = trials
  start = c(0.5, 0.5),          # value of thetas to start at
  num_steps = 1e4,               # number of steps to run the algorithm
  a1, b1,                      # params for prior for theta1
  a2, b2                       # params for prior for theta2
) {

  theta1 <- rep(NA, num_steps)  # trick to pre-allocate memory
  theta2 <- rep(NA, num_steps)  # trick to pre-allocate memory
```

```

theta1[1]           <- start[1]
theta2[1]           <- start[2]

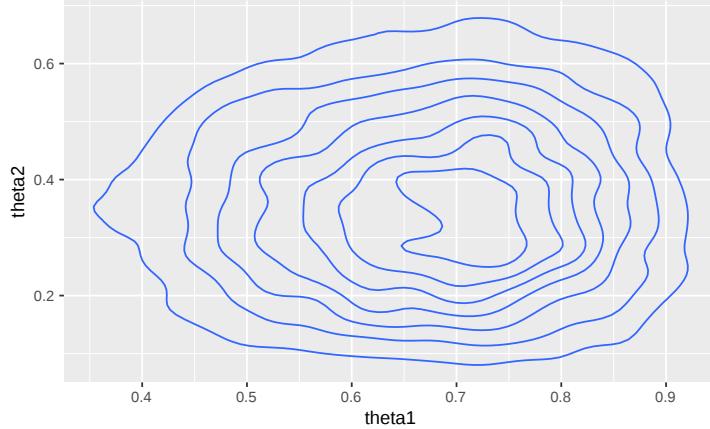
for (i in 1:(num_steps-1)) {
  if (i %% 2 == 1) { # update theta1
    theta1[i+1] <- rbeta(1, z1 + a1, n1 - z1 + b1)
    theta2[i+1] <- theta2[i]
  } else {           # update theta2
    theta1[i+1] <- theta1[i]
    theta2[i+1] <- rbeta(1, z2 + a2, n2 - z2 + b2)
  }
}

tibble(
  step = 1:num_steps,
  theta1 = theta1,
  theta2 = theta2,
)
}

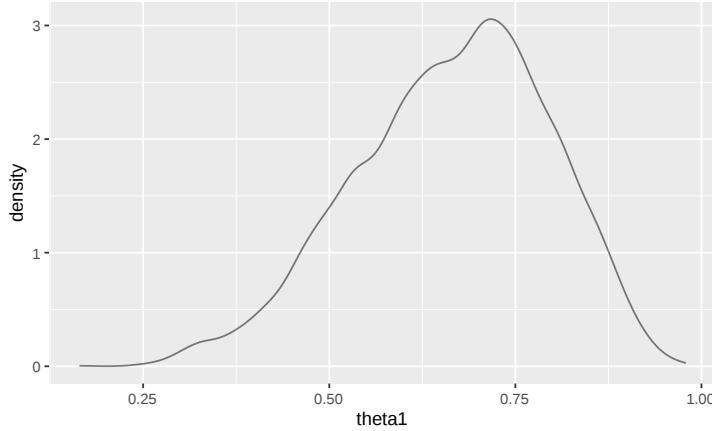
```

```
Gibbs <- gibbs_2coins(z1 = 6, n1 = 8, z2 = 2, n2 = 7,
                      a1 = 2, b1 = 2, a2 = 2, b2 = 2)
```

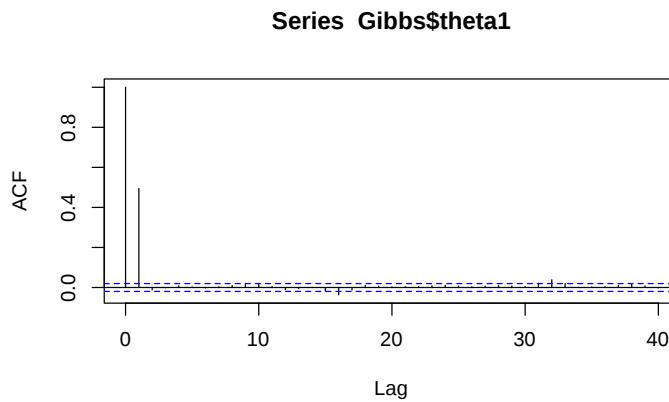
```
Gibbs %>% gf_density2d(theta2 ~ theta1)
```



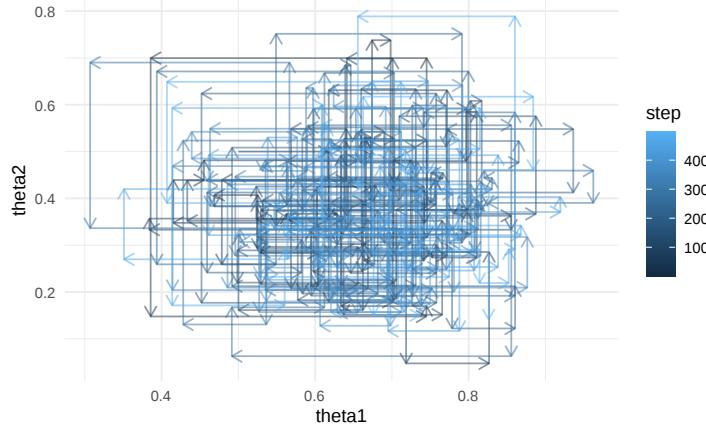
```
Gibbs %>% gf_dens(~theta1)
```



```
acf(Gibbs$theta1)
```



```
Gibbs %>% filter(step < 500) %>% gf_path(theta2 ~ theta1,
  color = ~step, alpha = 0.5, arrow = arrow(type = "open",
  angle = 30, length = unit(0.1, "inches"))) +
  theme_minimal()
```

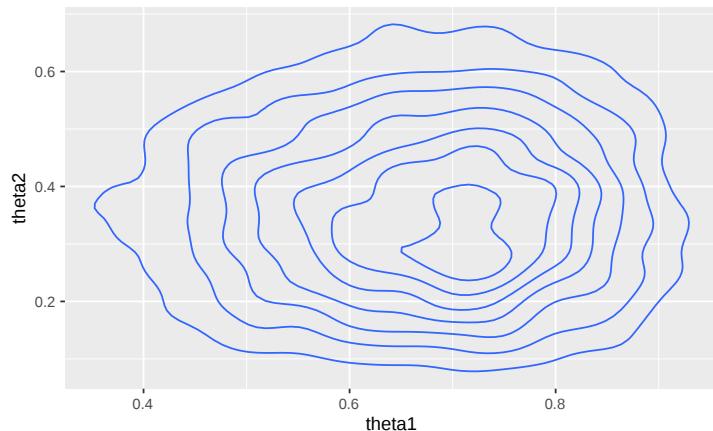


```
Gibbsplot <- Gibbs %>% filter(step < 500) %>% gf_path(theta2 ~
  theta1, color = ~step, alpha = 0.5, arrow = arrow(type = "open",
  angle = 30)) + theme_minimal() + transition_reveal(step)

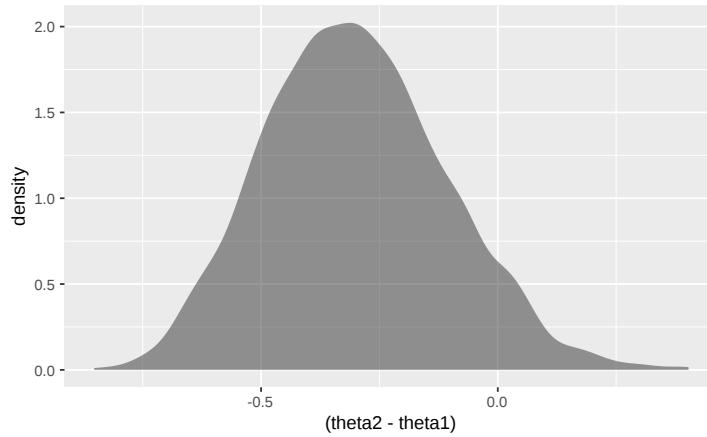
animate(Gibbsplot, fps = 1)
```

Ciclos completos

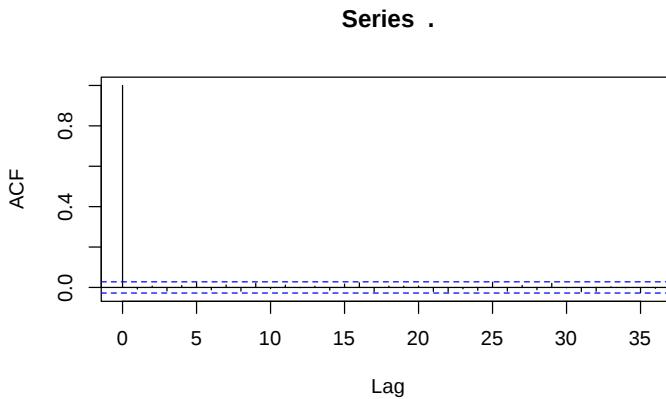
```
Gibbs %>% filter(step%%2 == 0) %>% gf_density2d(theta2 ~
  theta1)
```



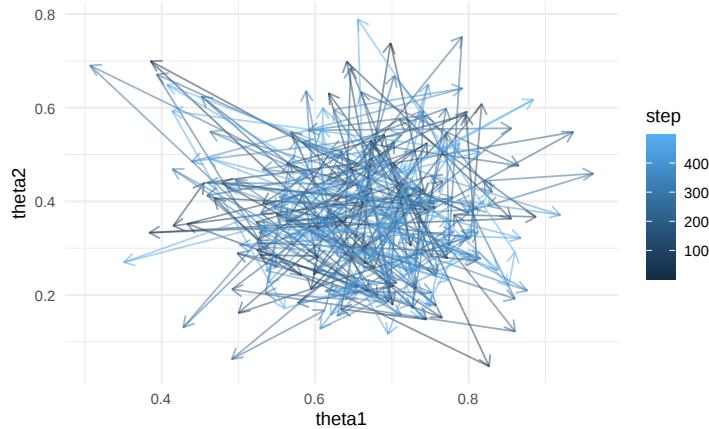
```
Gibbs %>% filter(step%%2 == 0) %>% gf_density(~(theta2 - theta1))
```



```
Gibbs %>% filter(step%%2 == 0) %>% mutate(difference = theta2 - theta1) %>% pull(difference) %>% acf()
```



```
Gibbs %>% filter(step < 500, step%%2 == 0) %>% gf_path(theta2 ~ theta1, color = ~step, alpha = 0.5, arrow = arrow(type = "open", angle = 30, length = unit(0.1, "inches")))) + theme_minimal()
```



```
Gibbsplot2 <- Gibbs %>% filter(step < 500, step%%2 == 0) %>% gf_path(theta2 ~ theta1, color = ~step, alpha = 0.5, arrow = arrow(type = "open", angle = 30)) + theme_minimal() + transition_reveal(step)

animate(Gibbsplot2, fps = 1)
```

4.6. Uso de JAGS

El paquete que usaremos en esta sección es `R2jags` y `coda`. Los cargamos con las instrucciones

```
remotes::install_github("rpruim/CalvinBayes")
```

```
library(R2jags)
library(coda)
library(CalvinBayes)
```

```
bernoulli <- read.csv(file = "data/bernoulli.csv",
sep = "")
```

```
tibble::glimpse(bernoulli)
```

```

## Rows: 50
## Columns: 1
## $ y <int> 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0,
##   mean(bernoulli$y)

## [1] 0.3

sd(bernoulli$y)

## [1] 0.46291

```

En el lenguaje usual de JAGS, el modelo debe escribirse de la forma:

```

model
{
  for (i in 1:N) {
    y[i] ~ dbern(theta)
  }
  theta ~ dbeta(1, 1)
}

```

donde `dbern` y `dbeta` son las densidades de una bernoulli y beta respectivamente. En este lenguaje no existen versiones vectorizadas de las funciones por lo que todo debe llenarse usando `for`'s. Una revisión completa de este lenguaje la pueden econtrar en su manual de uso ¹

El paquete `R2jags` tiene la capacidad que en lugar de usar este tipo de sintaxis, se pueda usar el lenguaje natural para escribir el modelo. Note el uso de `function` en este caso.

```

bern_model <- function() {
  for (i in 1:N) {
    y[i] ~ dbern(theta)
  }
  theta ~ dbeta(1, 1)
}

```

¹http://web.sgh.waw.pl/~atoroj/ekonometria_bayesowska/jags_user_manual.pdf

```
bern_jags <- jags(data = list(y = bernoulli$y, N = nrow(bernoulli)),
  model.file = bern_model, parameters.to.save = c("theta"))

## Compiling model graph
## Resolving undeclared variables
## Allocating nodes
## Graph information:
##   Observed stochastic nodes: 50
##   Unobserved stochastic nodes: 1
##   Total graph size: 53
##
## Initializing model
```

Veamos el resultado

```
bern_jags
```

```
## Inference for Bugs model at "/tmp/RtmpEajct8/model3d86204b61d.txt", fit using jags
## 3 chains, each with 2000 iterations (first 1000 discarded)
## n.sims = 3000 iterations saved
##          mu.vect sd.vect 2.5%   25%   50%   75% 97.5% Rhat n.eff
## theta     0.307  0.064 0.189  0.264  0.305  0.349  0.438 1.001  3000
## deviance 62.069  1.368 61.087 61.189 61.530 62.403 65.725 1.001  3000
##
## For each parameter, n.eff is a crude measure of effective sample size,
## and Rhat is the potential scale reduction factor (at convergence, Rhat=1).
##
## DIC info (using the rule, pD = var(deviance)/2)
## pD = 0.9 and DIC = 63.0
## DIC is an estimate of expected predictive error (lower deviance is better).
```

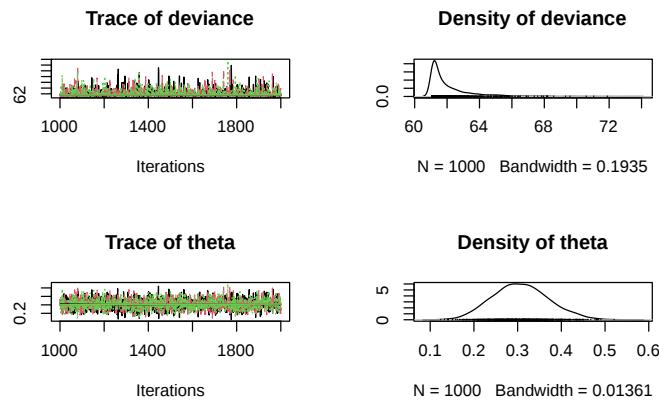
```
head(posterior(bern_jags))
```

```
## Error in verosimilitud(theta, data): argument "data" is missing, with no default
```

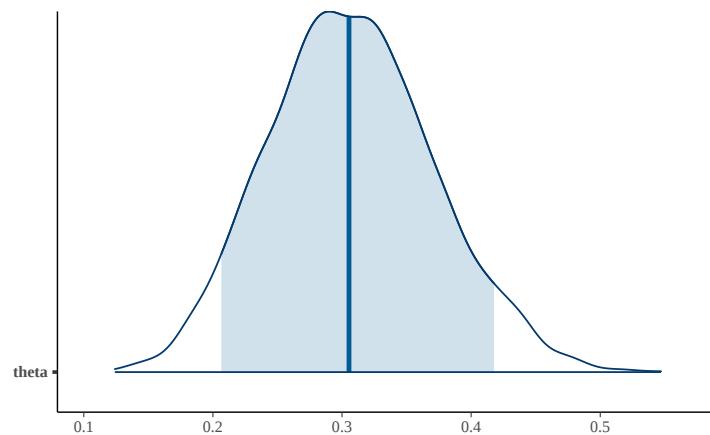
```
gf_dhistogram(~theta, data = posterior(bern_jags),
  bins = 50) %>% gf_dens(~theta, size = 1.5, alpha = 0.8) %>%
  gf_dist("beta", shape1 = 16, shape2 = 36, color = "red")
```

```
## Error in verosimilitud(theta, data): argument "data" is missing, with no de
```

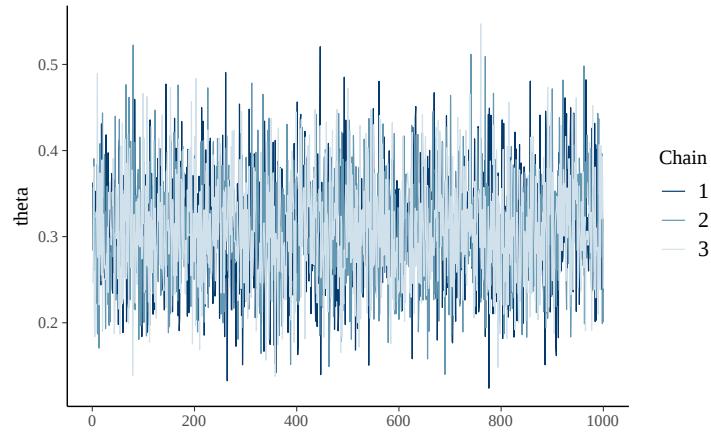
```
bern_mcmc <- as.mcmc(bern_jags)
plot(bern_mcmc)
```



```
library(bayesplot)
mcmc_areas(bern_mcmc, pars = c("theta"), prob = 0.9)
```



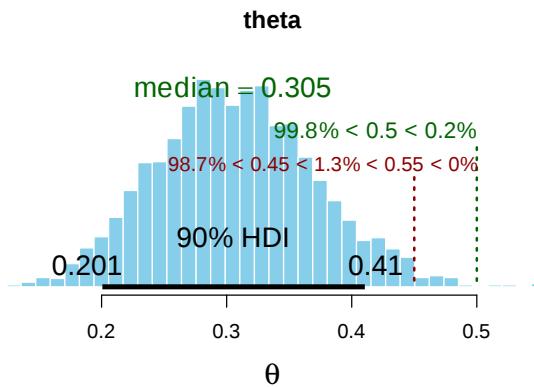
```
mcmc_trace(bern_mcmc, pars = "theta")
```



```
mcmc_trace(bern_mcmc, pars = "theta") %>% gf_facet_grid(Chain ~ .) %>% gf_refine(scale_color_viridis_d())
```

```
## Error: At least one layer must contain all facetting variables: `Chain`.  
## * Plot is missing `Chain`  
## * Layer 1 is missing `Chain`
```

```
plot_post(bern_mcmc[, "theta"], main = "theta", xlab = expression(theta),
  cenTend = "median", compVal = 0.5, ROPE = c(0.45,
  0.55), credMass = 0.9, quietly = TRUE)
```



```
twobernoulli <- read.csv("data/2bernoulli.csv")

knitr::kable(twobernoulli)
```

y	s
1	Reginald
0	Reginald
1	Reginald
0	Reginald
0	Tony
0	Tony
1	Tony
0	Tony
0	Tony
1	Tony
0	Tony

```

Target <- twobernoulli %>% rename(hit = y, subject = s)

Target %>% group_by(subject) %>% summarise(prop_0 = sum(1 -
  hit)/n(), prop_1 = sum(hit)/n(), attempts = n())

## # A tibble: 2 x 4
##   subject  prop_0  prop_1 attempts
##   <chr>     <dbl>    <dbl>     <int>
## 1 Reginald  0.25    0.75      8
## 2 Tony      0.714   0.286      7

bern2_model <- function() {
  for (i in 1:Nobs) {
    # each response is Bernoulli with the appropriate
    # theta
    hit[i] ~ dbern(theta[subject[i]])
  }
  for (s in 1:Nsub) {
    theta[s] ~ dbeta(2, 2)  # prior for each theta
  }
}

TargetList <- list(Nobs = nrow(Target), Nsub = 2, hit = Target$hit,
  subject = as.numeric(as.factor(Target$subject)))
TargetList

## $Nobs
## [1] 15
##
## $Nsub
## [1] 2
##
## $hit
##  [1] 1 0 1 1 1 1 0 0 0 1 0 0 1 0
##
## $subject
##  [1] 1 1 1 1 1 1 1 2 2 2 2 2 2 2

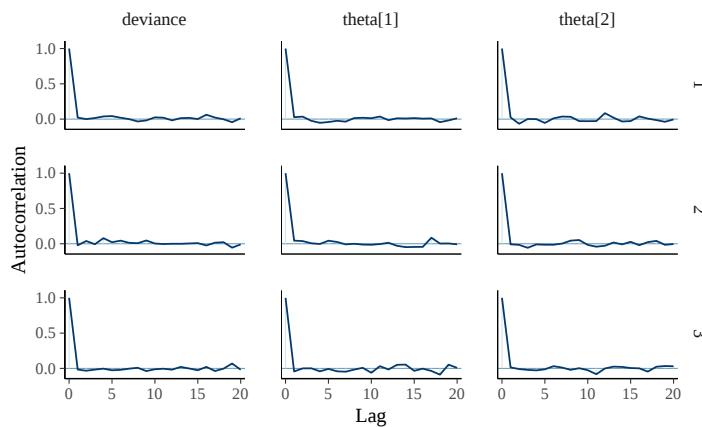
```

```
bern2_jags <- jags(data = TargetList, model = bern2_model,
parameters.to.save = "theta")
```

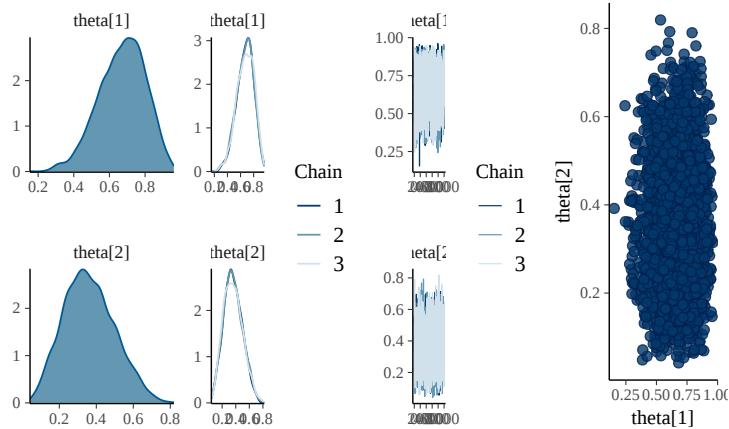
```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 15
##    Unobserved stochastic nodes: 2
##    Total graph size: 35
##
## Initializing model
```

```
bern2_mcmc <- as.mcmc(bern2_jags)

mcmc_acf(bern2_mcmc)
```



```
mcmc_combo(bern2_mcmc, combo = c("dens", "dens_overlay",
"trace", "scatter"), pars = c("theta[1]", "theta[2]"))
```



4.7. Uso de STAN

STAN² es otro tipo de lenguaje para definir modelos bayesiano. El lenguaje es un poco más sencillo, pero es particularmente útil para modelos bastante complejos.

STAN no usa el muestreo de Gibbs, sino en el método de Monte-Carlo Hamiltoniano. En el artículo (Hoffman y Gelman 2014) se propone el método NUTS para mejorar el muestreo de Gibbs.

En este curso no nos referiremos a este procedimiento, pero si veremos un poco de la sintaxis del lenguaje STAN.

```

data {
  int<lower=0> N;                      // number of trials
  int<lower=0, upper=1> y[N];    // success on trial n
}

parameters {
  real<lower=0, upper=1> theta; // chance of success
}

model {

```

²<https://mc-stan.org/>

```

theta ~ uniform(0, 1);           // prior
y ~ bernoulli(theta);          // likelihood
}

library(rstan)

fit <- stan(model_code = bern_stan$model_code, data = list(y = bernoulli$y,
  N = nrow(bernoulli)), iter = 5000)

## 
## SAMPLING FOR MODEL '4584de91ce47196187979d2da8a67926' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 1.5e-05 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take
## Chain 1: Adjust your expectations accordingly!
## Chain 1:
## Chain 1:
## Chain 1: Iteration: 1 / 5000 [  0%] (Warmup)
## Chain 1: Iteration: 500 / 5000 [ 10%] (Warmup)
## Chain 1: Iteration: 1000 / 5000 [ 20%] (Warmup)
## Chain 1: Iteration: 1500 / 5000 [ 30%] (Warmup)
## Chain 1: Iteration: 2000 / 5000 [ 40%] (Warmup)
## Chain 1: Iteration: 2500 / 5000 [ 50%] (Warmup)
## Chain 1: Iteration: 2501 / 5000 [ 50%] (Sampling)
## Chain 1: Iteration: 3000 / 5000 [ 60%] (Sampling)
## Chain 1: Iteration: 3500 / 5000 [ 70%] (Sampling)
## Chain 1: Iteration: 4000 / 5000 [ 80%] (Sampling)
## Chain 1: Iteration: 4500 / 5000 [ 90%] (Sampling)
## Chain 1: Iteration: 5000 / 5000 [100%] (Sampling)
## Chain 1:
## Chain 1: Elapsed Time: 0.016561 seconds (Warm-up)
## Chain 1:                      0.015825 seconds (Sampling)
## Chain 1:                      0.032386 seconds (Total)
## Chain 1:
## 
## SAMPLING FOR MODEL '4584de91ce47196187979d2da8a67926' NOW (CHAIN 2).
## Chain 2:

```

```
## Chain 2: Gradient evaluation took 5e-06 seconds
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0.05 s
## Chain 2: Adjust your expectations accordingly!
## Chain 2:
## Chain 2:
## Chain 2: Iteration: 1 / 5000 [ 0%] (Warmup)
## Chain 2: Iteration: 500 / 5000 [ 10%] (Warmup)
## Chain 2: Iteration: 1000 / 5000 [ 20%] (Warmup)
## Chain 2: Iteration: 1500 / 5000 [ 30%] (Warmup)
## Chain 2: Iteration: 2000 / 5000 [ 40%] (Warmup)
## Chain 2: Iteration: 2500 / 5000 [ 50%] (Warmup)
## Chain 2: Iteration: 2501 / 5000 [ 50%] (Sampling)
## Chain 2: Iteration: 3000 / 5000 [ 60%] (Sampling)
## Chain 2: Iteration: 3500 / 5000 [ 70%] (Sampling)
## Chain 2: Iteration: 4000 / 5000 [ 80%] (Sampling)
## Chain 2: Iteration: 4500 / 5000 [ 90%] (Sampling)
## Chain 2: Iteration: 5000 / 5000 [100%] (Sampling)
## Chain 2:
## Chain 2: Elapsed Time: 0.015872 seconds (Warm-up)
## Chain 2:           0.019947 seconds (Sampling)
## Chain 2:           0.035819 seconds (Total)
## Chain 2:
## 
## SAMPLING FOR MODEL '4584de91ce47196187979d2da8a67926' NOW (CHAIN 3).
## Chain 3:
## Chain 3: Gradient evaluation took 8e-06 seconds
## Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0.08 s
## Chain 3: Adjust your expectations accordingly!
## Chain 3:
## Chain 3:
## Chain 3: Iteration: 1 / 5000 [ 0%] (Warmup)
## Chain 3: Iteration: 500 / 5000 [ 10%] (Warmup)
## Chain 3: Iteration: 1000 / 5000 [ 20%] (Warmup)
## Chain 3: Iteration: 1500 / 5000 [ 30%] (Warmup)
## Chain 3: Iteration: 2000 / 5000 [ 40%] (Warmup)
## Chain 3: Iteration: 2500 / 5000 [ 50%] (Warmup)
## Chain 3: Iteration: 2501 / 5000 [ 50%] (Sampling)
## Chain 3: Iteration: 3000 / 5000 [ 60%] (Sampling)
```

```

## Chain 3: Iteration: 3500 / 5000 [ 70%] (Sampling)
## Chain 3: Iteration: 4000 / 5000 [ 80%] (Sampling)
## Chain 3: Iteration: 4500 / 5000 [ 90%] (Sampling)
## Chain 3: Iteration: 5000 / 5000 [100%] (Sampling)
## Chain 3:
## Chain 3: Elapsed Time: 0.020457 seconds (Warm-up)
## Chain 3: 0.017149 seconds (Sampling)
## Chain 3: 0.037606 seconds (Total)
## Chain 3:
## 
## SAMPLING FOR MODEL '4584de91ce47196187979d2da8a67926' NOW (CHAIN 4).
## Chain 4:
## Chain 4: Gradient evaluation took 4e-06 seconds
## Chain 4: 1000 transitions using 10 leapfrog steps per transition would take
## Chain 4: Adjust your expectations accordingly!
## Chain 4:
## Chain 4:
## Chain 4: Iteration: 1 / 5000 [ 0%] (Warmup)
## Chain 4: Iteration: 500 / 5000 [ 10%] (Warmup)
## Chain 4: Iteration: 1000 / 5000 [ 20%] (Warmup)
## Chain 4: Iteration: 1500 / 5000 [ 30%] (Warmup)
## Chain 4: Iteration: 2000 / 5000 [ 40%] (Warmup)
## Chain 4: Iteration: 2500 / 5000 [ 50%] (Warmup)
## Chain 4: Iteration: 2501 / 5000 [ 50%] (Sampling)
## Chain 4: Iteration: 3000 / 5000 [ 60%] (Sampling)
## Chain 4: Iteration: 3500 / 5000 [ 70%] (Sampling)
## Chain 4: Iteration: 4000 / 5000 [ 80%] (Sampling)
## Chain 4: Iteration: 4500 / 5000 [ 90%] (Sampling)
## Chain 4: Iteration: 5000 / 5000 [100%] (Sampling)
## Chain 4:
## Chain 4: Elapsed Time: 0.01831 seconds (Warm-up)
## Chain 4: 0.015764 seconds (Sampling)
## Chain 4: 0.034074 seconds (Total)
## Chain 4:

print(fit, probs = c(0.1, 0.9))

## Inference for Stan model: 4584de91ce47196187979d2da8a67926.

```

```

## 4 chains, each with iter=5000; warmup=2500; thin=1;
## post-warmup draws per chain=2500, total post-warmup draws=10000.
##
##           mean se_mean    sd   10%   90% n_eff Rhat
## theta    0.31    0.00 0.06   0.23   0.39  3136     1
## lp__ -32.60    0.01 0.72 -33.49 -32.11  4750     1
##
## Samples were drawn using NUTS(diag_e) at Wed Jun 17 15:30:58 2020.
## For each parameter, n_eff is a crude measure of effective sample size,
## and Rhat is the potential scale reduction factor on split chains (at
## convergence, Rhat=1).

theta_draws <- extract(fit)$theta

mean(theta_draws)

## [1] 0.3074515

quantile(theta_draws, probs = c(0.1, 0.9))

##           10%      90%
## 0.2282770 0.3911384

shinystan::launch_shinystan(fit)

```

Ejercicio 4.1. Replique los resultados anteriores pero para el caso de 2 monedas y comente los resultados.

4.8. Ejercicios

1. Del libro (Albert y col. 2009)
 - **Sección 3:** 3, 7.
 - **Sección 6:** 1, 3.
2. Del libro (Kruschke 2014)
 - **Sección 6:** 2.
 - **Sección 7:** 2.

Capítulo 5

Métodos lineares de regresión

NOTA: Para los siguientes capítulos nos basaremos en los libros (Hastie, Tibshirani y Friedman 2009) y (James y col. 2013).

5.1. Introducción

Supongamos que tenemos p variables de entrada que mezcladas con alguna relación desconocida y que provocan una respuesta Y de salida.

$$Y = f(X_1, \dots, X_p) + \varepsilon \quad (5.1)$$

Aquí f es desconocida, las variables X 's son las variables de entrada y ε es el error cometido por hacer esta aproximación.

Hay dos motivos para estimar f

1. **Predicción:** Si se estima f con \hat{f} entonces

$$\hat{Y} = \hat{f}(X_1, \dots, X_p).$$

Y si tuvieramos valores nuevos de los X 's entonces podríamos estimar el valor que el corresponde a Y .

Aquí lo importante es que los resultados sean preciso:

- a. **Error reducible:** Error de \hat{f} alrededor de f .
- b. **Error irreducible:** Error propio de las observaciones (muestreo).

$$\begin{aligned}\mathbb{E} [\hat{Y} - Y] &= \mathbb{E} \left[(f(X_1, \dots, X_p) + \varepsilon - \hat{f}(X_1, \dots, X_p))^2 \right] \\ &= \underbrace{\left(f(X_1, \dots, X_p) - \hat{f}(X_1, \dots, X_p) \right)^2}_{\text{Reducible}} + \underbrace{\text{Var}(\varepsilon)}_{\text{irreducible}}.\end{aligned}$$

2. **Inferencia:** Entender la relación entre X y Y .

- ¿Cuál es la relación entre las variables predictoras y la respuesta?
- ¿Cuáles son más importantes?
- ¿El modelo es correcto?

5.2. Regresión lineal

El caso más sencillo es cuando esta relación es lineal y se describe de la siguiente forma

$$Y = \beta_0 + \beta_1 X_1 + \cdots + \beta_k X_k + \varepsilon.$$

Aquí los valores β 's son constantes a estimar, las variables X 's son las variables de entrada y ε es el error cometido por hacer esta aproximación.

Los X 's pueden ser

1. Cuantitativos o Transformaciones.
2. Cualitativos.

En el caso de ser cualitativos existe un truco para incluirlos dentro de la regresión

Ejemplo 5.1. Se tiene la variable G codificada con Casado (1), Soltero (2), Divorciado (3) y Unión Libre (4). Si queremos meter esta variable en una regresión debemos tomarla de la forma

$$X_j = \mathbf{1}_{\{G=j+1\}}$$

que resulta en la matriz

$$\begin{matrix} X_1 & X_2 & X_3 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{matrix}$$

Existen otras formas de codificar este tipo de variables, pero esa es la más común.

5.2.1. Forma matricial

Podemos escribir la regresión de la forma

$$\mathbf{Y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\varepsilon}$$

donde

$$\mathbf{Y} = \begin{pmatrix} Y_1 \\ \vdots \\ Y_n \end{pmatrix}_{n \times 1} \quad \mathbf{Y} = \begin{pmatrix} 1 & X_{1,1} & \cdots & X_{p,1} \\ \vdots & \vdots & \cdots & \vdots \\ 1 & X_{1,n} & \cdots & X_{p,n} \end{pmatrix}_{n \times (p+1)} \\ \boldsymbol{\varepsilon} = \begin{pmatrix} \varepsilon_1 \\ \vdots \\ \varepsilon_n \end{pmatrix}_{n \times 1} \quad \boldsymbol{\beta} = \begin{pmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_p \end{pmatrix}_{(p+1) \times 1}$$

Suponemos que $\mathbb{E}[\varepsilon_i] = 0$ y $\text{Var}(\varepsilon_i) = \sigma^2$

La forma de resolver este problema es por mínimos cuadrados. Es decir, buscamos el $\hat{\beta}$ que cumpla lo siguiente:

$$\hat{\beta} = \operatorname{argmin}_{\beta} (\mathbf{Y} - \mathbf{X}\beta)^{\top} (\mathbf{Y} - \mathbf{X}\beta) \quad (5.2)$$

$$= \operatorname{argmin}_{\beta} \sum_{i=1}^n \left(Y_i - \beta_0 - \sum_{j=1}^p X_{j,i} \beta_j \right) \quad (5.3)$$

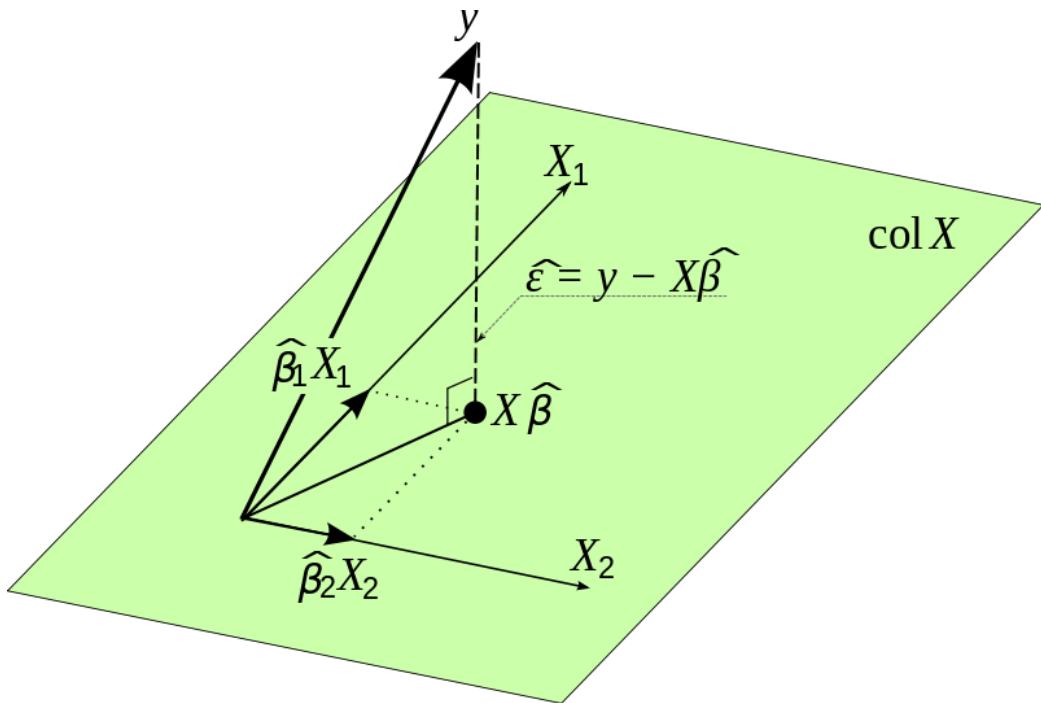


Figura 5.1: Tomado de https://www.wikiwand.com/en/Ordinary_least_squares

Suponga que γ es un vector cualquiera en \mathbb{R}^{p+1} y tenemos a $V = \{\mathbf{X}\gamma, \gamma \in \mathbb{R}^{p+1}\}$.

$$\mathbf{X}\beta = \operatorname{Proy}_V \mathbf{Y}$$

Entonces dado que

$$\mathbf{Y} - \mathbf{X}\boldsymbol{\beta} \perp V\mathbf{Y} - \mathbf{X}\boldsymbol{\beta}, \forall \boldsymbol{\gamma} \in \mathbb{R}^{p+1}.$$

$$\begin{aligned} & \langle \mathbf{X}\boldsymbol{\gamma}, \mathbf{Y} - \mathbf{X}\boldsymbol{\beta} \rangle = 0 \\ & \boldsymbol{\gamma}^\top \mathbf{X}^\top (\mathbf{Y} - \mathbf{X}\boldsymbol{\beta}) = 0 \\ & \boldsymbol{\gamma}^\top \mathbf{X}^\top \mathbf{Y} = \boldsymbol{\gamma}^\top \mathbf{X}^\top \mathbf{X}\boldsymbol{\beta} \\ & \mathbf{X}^\top \mathbf{Y} = \mathbf{X}^\top \mathbf{X}\boldsymbol{\beta} \\ & \boldsymbol{\beta} = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{Y} \end{aligned}$$

Donde $\mathbf{X}^\top \mathbf{X}$ debe ser invertible. Si no es así, se puede construir su inversa generalizada pero no garantiza la unicidad de los β 's. Es decir, puede existir $\hat{\boldsymbol{\beta}} \neq \tilde{\boldsymbol{\beta}}$ tal que $\mathbf{X}\hat{\boldsymbol{\beta}} = \mathbf{X}\tilde{\boldsymbol{\beta}}$

En el caso de predicción tenemos que

$$\begin{aligned} \hat{\mathbf{Y}} &= \mathbf{X}\boldsymbol{\beta} \\ &= \mathbf{X}(\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{Y} \\ &= H\mathbf{Y} \end{aligned}$$

Donde H es la matriz «techo» o «hat». Es la proyección de \mathbf{Y} al espacio de las columnas de \mathbf{X} .

Ejercicio 5.1. Suponga que tenemos la regresión simple

$$Y = \beta_0 + \beta_1 X_1 + \varepsilon.$$

Muestre que β_0 y β_1 son

Para el caso de la regresión simple tenemos que

$$\begin{aligned} \hat{\beta}_1 &= \frac{\sum_{i=1}^n (X_i - \bar{X})(Y_i - \bar{Y})}{\sum_{i=1}^n (X_i - \bar{x})^2} \\ \hat{\beta}_0 &= \bar{Y} - \hat{\beta}_1 \bar{X} \end{aligned}$$

usando los siguiente métodos:

1. El método de proyecciones.
2. Minimizando el criterio de mínimos cuadrados. Ecuación (5.3).

5.2.2. Laboratorio

Usemos la base `mtcars` para los siguientes ejemplos. Toda la información de esta base se encuentra en `?mtcars`.

```
mtcars <- within(mtcars, {
  vs <- factor(vs, labels = c("V-Shape", "Straight-Line"))
  am <- factor(am, labels = c("automatic", "manual"))
  cyl <- factor(cyl)
  gear <- factor(gear)
  carb <- factor(carb)
})

head(mtcars)

##          mpg cyl disp  hp drat    wt  qsec      vs      am
## Mazda RX4     21.0   6 160 110 3.90 2.620 16.46 V-Shape manual
## Mazda RX4 Wag 21.0   6 160 110 3.90 2.875 17.02 V-Shape manual
## Datsun 710    22.8   4 108  93 3.85 2.320 18.61 Straight-Line manual
## Hornet 4 Drive 21.4   6 258 110 3.08 3.215 19.44 Straight-Line automatic
## Hornet Sportabout 18.7   8 360 175 3.15 3.440 17.02 V-Shape automatic
## Valiant       18.1   6 225 105 2.76 3.460 20.22 Straight-Line automatic
##              gear carb
## Mazda RX4      4     4
## Mazda RX4 Wag   4     4
## Datsun 710     4     1
## Hornet 4 Drive  3     1
## Hornet Sportabout 3     2
## Valiant        3     1

summary(mtcars)

##      mpg          cyl          disp         hp         drat
##  Min.   : 12.00   Min.   : 4.00   Min.   : 70.00   Min.   : 50.00   Min.   : 2.770
##  1st Qu.: 15.43   1st Qu.: 6.00   1st Qu.: 120.00   1st Qu.: 95.00   1st Qu.: 3.435
##  Median : 19.20   Median : 8.00   Median : 141.00   Median : 123.00   Median : 3.605
##  Mean   : 20.09   Mean   : 7.18   Mean   : 132.90   Mean   : 102.60   Mean   : 3.704
##  3rd Qu.: 22.80   3rd Qu.: 8.00   3rd Qu.: 164.00   3rd Qu.: 141.00   3rd Qu.: 3.905
##  Max.   : 33.90   Max.   :10.00   Max.   :360.00   Max.   :264.00   Max.   :4.930
##      wt          qsec
##  Min.   : 1.580   Min.   : 10.40
##  1st Qu.: 2.385   1st Qu.: 15.40
##  Median : 3.080   Median : 17.80
##  Mean   : 3.436   Mean   : 17.86
##  3rd Qu.: 3.620   3rd Qu.: 19.20
##  Max.   : 5.420   Max.   :22.80
```

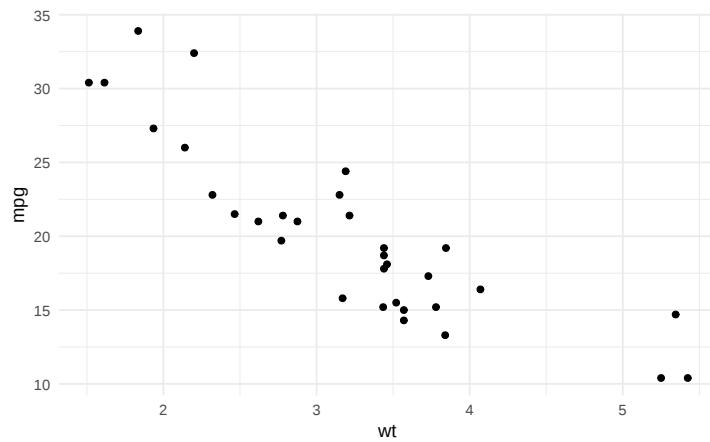
```

##   Min. :10.40    4:11    Min.   : 71.1    Min.   : 52.0    Min.   :2.760
## 1st Qu.:15.43    6: 7    1st Qu.:120.8   1st Qu.: 96.5   1st Qu.:3.080
## Median :19.20    8:14    Median :196.3    Median :123.0    Median :3.695
## Mean   :20.09          Mean   :230.7    Mean   :146.7    Mean   :3.597
## 3rd Qu.:22.80          3rd Qu.:326.0   3rd Qu.:180.0    3rd Qu.:3.920
## Max.   :33.90          Max.   :472.0    Max.   :335.0    Max.   :4.930
##           wt            qsec           vs            am      gear
##   Min.   :1.513      Min.   :14.50    V-Shape     :18    automatic:19  3:15
## 1st Qu.:2.581      1st Qu.:16.89   Straight-Line:14  manual   :13  4:12
## Median :3.325      Median :17.71
## Mean   :3.217      Mean   :17.85
## 3rd Qu.:3.610      3rd Qu.:18.90
## Max.   :5.424      Max.   :22.90
## carb
## 1: 7
## 2:10
## 3: 3
## 4:10
## 6: 1
## 8: 1

```

Observemos las relaciones generales de las variables de esta base de datos

```
ggplot(mtcars) + geom_point(aes(wt, mpg)) + theme_minimal()
```



El objetivo es tratar la eficiencia del automóvil `mpg` con respecto a su peso `wt`.

Usaremos una regresión lineal para encontrar los coeficientes.

Primero hay que construir la matriz de diseño

```
X <- mtcars$wt
head(X)

## [1] 2.620 2.875 2.320 3.215 3.440 3.460

Y <- mtcars$mpg
head(Y)

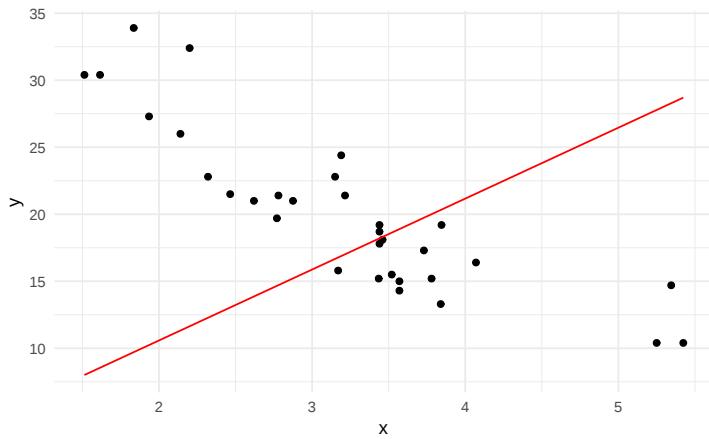
## [1] 21.0 21.0 22.8 21.4 18.7 18.1

(beta1 <- solve(t(X) %*% X) %*% t(X) %*% Y)

##          [,1]
## [1,] 5.291624

dfreg <- data.frame(x = X, yreg = X %*% beta1) %>%
  arrange(x)

ggplot(data = data.frame(x = X, y = Y)) + geom_point(aes(x,
y)) + geom_line(data = dfreg, aes(x, yreg), color = "red") +
theme_minimal()
```



```
X <- cbind(1, mtcars$wt)
head(X)
```

```
##      [,1]  [,2]
## [1,]    1 2.620
## [2,]    1 2.875
## [3,]    1 2.320
## [4,]    1 3.215
## [5,]    1 3.440
## [6,]    1 3.460
```

```
Y <- mtcars$mpg
head(Y)
```

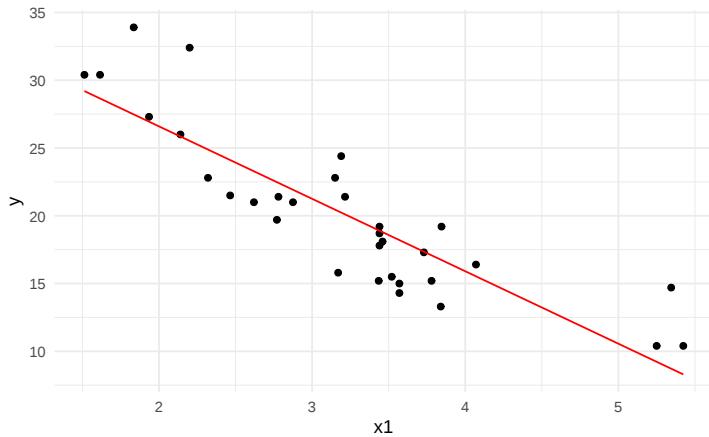
```
## [1] 21.0 21.0 22.8 21.4 18.7 18.1
```

```
(beta01 <- solve(t(X) %*% X) %*% t(X) %*% Y)
```

```
##      [,1]
## [1,] 37.285126
## [2,] -5.344472
```

```
dfreg <- data.frame(x = X, yreg = X %*% beta01) %>%
  arrange(x.2)
```

```
ggplot(data = data.frame(x0 = X[, 1], x1 = X[, 2],
  y = Y)) + geom_point(aes(x1, y)) + geom_line(data = dfreg,
  aes(x.2, yreg), color = "red") + theme_minimal()
```



Ojo obviamente esto se puede hacer más fácil con los siguientes comandos

```
lm(mpg ~ -1 + wt, data = mtcars)
```

```
##
## Call:
## lm(formula = mpg ~ -1 + wt, data = mtcars)
##
## Coefficients:
##       wt
## 5.292
```

```
lm(mpg ~ wt, data = mtcars)
```

```
##
## Call:
## lm(formula = mpg ~ wt, data = mtcars)
##
## Coefficients:
## (Intercept)          wt
##     37.285      -5.344
```

Suponga que queremos incluir una variable categorica como cyl (Número de cilindros). Lo que se debe hacer es convertir esta variable a dummy.

```
X <- model.matrix(mpg ~ cyl, data = mtcars)
```

```
head(X)
```

```
##                                     (Intercept) cyl6 cyl8
## Mazda RX4                               1     1     0
## Mazda RX4 Wag                            1     1     0
## Datsun 710                               1     0     0
## Hornet 4 Drive                          1     1     0
## Hornet Sportabout                       1     0     1
## Valiant                                 1     1     0
```

```
(betas <- solve(t(X) %*% X) %*% t(X) %*% Y)
```

```
##          [,1]
## (Intercept) 26.663636
## cyl6       -6.920779
## cyl8      -11.563636
```

```
(cylreg <- lm(mpg ~ cyl, data = mtcars))
```

```
##
## Call:
## lm(formula = mpg ~ cyl, data = mtcars)
##
## Coefficients:
## (Intercept)      cyl6      cyl8
## 26.664        -6.921     -11.564
```

```
(betaslm <- coefficients(cylreg))
```

```
## (Intercept)      cyl6      cyl8
## 26.663636    -6.920779   -11.563636
```

```
# Efecto cyl4: cyl4 = 1, cyl6 = 0, cyl8 = 0
```

```
betaslm[1]
```

```
## (Intercept)
```

```
##      26.66364
```

```
# Efecto cyl6: cyl4 = 1, cyl6 = 1, cyl8 = 0
```

```
betaslm[1] + betaslm[2]
```

```
## (Intercept)
```

```
##      19.74286
```

```
# Efecto cyl8: cyl4 = 1, cyl6 = 0, cyl8 = 1
```

```
betaslm[1] + betaslm[3]
```

```
## (Intercept)
```

```
##      15.1
```

5.3. Propiedades estadísticas

Uno de los supuestos fundamentales de regresión lineal es que

$$\varepsilon \sim \mathcal{N}(0, \sigma^2 I)$$

En ese caso

$$Y = X\beta + \varepsilon \sim \mathcal{N}(X\beta, \sigma^2 I)$$

Y además

$$\begin{aligned}
\hat{\beta} &= (X^\top X)^{-1} X^\top Y \\
&\sim \mathcal{N}\left((X^\top X)^{-1} X^\top X \beta, ((X^\top X)^{-1} X^\top) \sigma I ((X^\top X)^{-1} X^\top)^\top\right) \\
&\sim \mathcal{N}\left(\beta, \sigma^2 (X^\top X)^{-1}\right)
\end{aligned}$$

Es decir, que

$$\begin{aligned}
\mathbb{E}[\hat{\beta}] &= \beta \\
\text{Var}(\hat{\beta}) &= \sigma^2 (X^\top X)^{-1}
\end{aligned}$$

Ejercicio 5.2. Encuentre la varianza para β_0 y β_1 para el caso de la regresión simple.

La estimación de σ^2

$$\begin{aligned}
\hat{\sigma}^2 &= \frac{1}{n-p-1} \sum_{i=1}^n (Y_i - \hat{Y}_i)^2 \\
&= \frac{1}{n-p-1} \|Y - X\hat{\beta}\|^2 \\
&= \frac{1}{n-p-1} \|Y - \text{Proy}_V Y\|^2
\end{aligned}$$

Otra forma de verlo es

$$\begin{aligned}
Y - \text{Proy}_V Y &= X\beta + \varepsilon - \text{Proy}_V(X\beta + \varepsilon) \\
&= X\beta - \underbrace{\text{Proy}_V(X\beta)}_{\in V} + \varepsilon - \underbrace{\text{Proy}_V(\varepsilon)}_{=0} \\
&= X\beta - X\beta + \varepsilon \\
&= \text{Proy}_{V^\perp}(\varepsilon)
\end{aligned}$$

$$\hat{\sigma}^2 = \frac{1}{\dim(V^\perp)} \|\text{Proy}_{V^\perp} \varepsilon\|$$

Cumple con la propiedad que $\mathbb{E}[\hat{\sigma}] = \sigma^2$.

Y además $(n - p - 1)\hat{\sigma}^2 \sim \sigma^2 \chi_{n-p-1}^2$.

5.3.1. Prueba t

Dado que los coeficientes β son normales, se puede hacer la prueba de hipótesis

$$H_0 : \beta_j = 0 \quad \text{vs} \quad H_1 : \beta_j \neq 0.$$

El estadístico es

$$z_j = \frac{\hat{\beta}_j}{\hat{\sigma} \sqrt{v_j}}$$

donde v_j es el j -ésimo elemento de la diagonal de $(X^\top X)^{-1}$.

Bajo H_0 $z_j \sim t_{n-p-1}$ y se rechaza H_0 si

$$|z_j| > t_{n-p-1, 1-\frac{\alpha}{2}}$$

5.3.2. Prueba F

$$H_0 : \beta_1 = \dots = \beta_p = 0 \quad \text{vs} \quad H_1 : \text{al menos un } \beta \text{ no es cero.}$$

En este caso queremos comparar el modelo nulo $Y = \beta_0 + \varepsilon$ contra el modelo completo $Y = \beta_0 + \beta_1 X_1 + \dots + \beta_p X_p + \varepsilon$.

Defina

$$\begin{aligned} TSS &= \sum_{i=1}^n (Y_i - \bar{Y})^2 \\ RSS &= \sum_{i=1}^n (Y_i - \hat{Y})^2 \end{aligned}$$

TSS = Total sum of squares

RSS = Residual sum of squares

Entonces

$$F = \frac{\frac{TSS-RSS}{p}}{\frac{RSS}{n-p-1}} \sim \frac{\chi_p^2}{\chi_{n-p-1}^2}.$$

Rechazamos H_0 si

$$F > F_{p,n-p-1,1-\alpha}.$$

5.3.3. Laboratorio

Siguiendo con nuestro ejemplo, vamos a explorar un poco más la función `lm`.

```
fit <- lm(mpg ~ wt, data = mtcars)
summary(fit)

##
## Call:
## lm(formula = mpg ~ wt, data = mtcars)
##
## Residuals:
##     Min      1Q  Median      3Q     Max 
## -4.5432 -2.3647 -0.1252  1.4096  6.8727 
##
## Coefficients:
##             Estimate Std. Error t value     Pr(>|t|)    
## (Intercept) 37.2851    1.8776 19.858 < 0.000000000000002 ***
## wt          -5.3445    0.5591 -9.559     0.000000000129 ***
## ---        
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1 
##
## Residual standard error: 3.046 on 30 degrees of freedom
## Multiple R-squared:  0.7528, Adjusted R-squared:  0.7446 
## F-statistic: 91.38 on 1 and 30 DF,  p-value: 0.0000000001294
```

```

fit <- lm(mpg ~ wt + cyl, data = mtcars)
summary(fit)

##
## Call:
## lm(formula = mpg ~ wt + cyl, data = mtcars)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.5890 -1.2357 -0.5159  1.3845  5.7915
##
## Coefficients:
##             Estimate Std. Error t value     Pr(>|t|)
## (Intercept) 33.9908    1.8878 18.006 < 0.0000000000000002 ***
## wt          -3.2056    0.7539 -4.252     0.000213 ***
## cyl6        -4.2556    1.3861 -3.070     0.004718 **
## cyl8        -6.0709    1.6523 -3.674     0.000999 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.557 on 28 degrees of freedom
## Multiple R-squared:  0.8374, Adjusted R-squared:  0.82
## F-statistic: 48.08 on 3 and 28 DF,  p-value: 0.00000000003594

fit <- lm(mpg ~ ., data = mtcars)
summary(fit)

##
## Call:
## lm(formula = mpg ~ ., data = mtcars)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.5087 -1.3584 -0.0948  0.7745  4.6251
##
## Coefficients:

```

```

##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) 23.87913  20.06582   1.190  0.2525
## cyl6        -2.64870   3.04089  -0.871  0.3975
## cyl8        -0.33616   7.15954  -0.047  0.9632
## disp         0.03555   0.03190   1.114  0.2827
## hp          -0.07051   0.03943  -1.788  0.0939 .
## drat         1.18283   2.48348   0.476  0.6407
## wt          -4.52978   2.53875  -1.784  0.0946 .
## qsec         0.36784   0.93540   0.393  0.6997
## vsStraight-Line 1.93085   2.87126   0.672  0.5115
## ammanual    1.21212   3.21355   0.377  0.7113
## gear4        1.11435   3.79952   0.293  0.7733
## gear5        2.52840   3.73636   0.677  0.5089
## carb2       -0.97935   2.31797  -0.423  0.6787
## carb3        2.99964   4.29355   0.699  0.4955
## carb4        1.09142   4.44962   0.245  0.8096
## carb6        4.47757   6.38406   0.701  0.4938
## carb8        7.25041   8.36057   0.867  0.3995
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.833 on 15 degrees of freedom
## Multiple R-squared:  0.8931, Adjusted R-squared:  0.779
## F-statistic:  7.83 on 16 and 15 DF,  p-value: 0.000124

```

5.4. Medida de bondad de ajuste

La prueba F nos dice si un modelo es nulo o no, pero no nos dice si tengo dos modelos cuál es mejor que otro.

Hay varias medidas para comparar modelos (la veremos con más detalle en otro capítulo):

- Error estándar residual (σ)
- R^2 y R^2 ajustado
- C_p de Mallows
- Akaike Information Criterion (AIC)

- Bayesian Information Criterion (BIC)

Los índices C_p de Mallows, AIC y BIC los veremos después.

Error estándar residual Se define como

$$\begin{aligned} \text{RSE} &= \sqrt{\hat{\sigma}^2} \\ &= \sqrt{\frac{1}{n-p-1} \sum_{i=1}^n (Y_i - \hat{Y}_i)^2} \\ &= \sqrt{\frac{\text{RSS}}{n-p-1}} \end{aligned}$$

Entre más pequeño mejor, pero **depende de las unidades de Y** .

Estadístico R^2

$$R^2 = \frac{\text{TSS} - \text{RSS}}{\text{TSS}} = 1 - \frac{\text{RSS}}{\text{TSS}}$$

- **RSS:** Varianza sin explicar por el modelo **completo**.
- **TSS:** Varianza sin explicar por el modelo **nulo**.

Estadístico R^2 ajustado

$$R_{adj}^2 = 1 - \frac{\frac{\text{RSS}}{n-p-1}}{\frac{\text{TSS}}{n-1}}$$

5.4.1. Laboratorio

```
# Número de datos
n <- 1000
# Número de variables
p <- 2
```

```
x1 <- rnorm(1000)
x2 <- runif(1000)
y <- 1 + x1 + x2 + rnorm(1000, sd = 0.5)

fit <- lm(y ~ x1 + x2)
```

5.4.1.1. R^2

```
(TSS <- sum((y - mean(y))^2))

## [1] 1402.26

(RSS <- sum((y - fitted(fit))^2))

## [1] 250.6091

1 - RSS/TSS

## [1] 0.821282
```

Otra forma de entender el R^2 es notando que

```
cor(y, fitted(fit))^2

## [1] 0.821282
```

5.4.1.2. R^2 ajustado

```
(TSS_adj <- TSS/(n - 1))

## [1] 1.403663
```

```
(RSS_adj <- RSS/(n - p - 1))
```

```
## [1] 0.2513632
```

```
1 - RSS_adj/TSS_adj
```

```
## [1] 0.8209235
```

5.4.1.3. `summary`

```
summary(fit)
```

```
##
## Call:
## lm(formula = y ~ x1 + x2)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.44716 -0.34692 -0.01095  0.34280  1.53693
##
## Coefficients:
##             Estimate Std. Error t value            Pr(>|t|)
## (Intercept)  0.94616   0.03196  29.60 <0.0000000000000002 ***
## x1          1.03322   0.01614  64.02 <0.0000000000000002 ***
## x2          1.09560   0.05491  19.95 <0.0000000000000002 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.5014 on 997 degrees of freedom
## Multiple R-squared:  0.8213, Adjusted R-squared:  0.8209
## F-statistic: 2291 on 2 and 997 DF,  p-value: < 0.0000000000000022
```

5.5. Predicción

Hay dos tipos de errores que se deben considerar en regresiones lineales:

1. **Error Reducible:** Recuerde que $\hat{Y} = \hat{X}\hat{\beta}$ es el estimador de la función $f(X) = X\beta = \beta_0 + \beta_1X_1 + \cdots + \beta_pX_p$.

Por lo tanto su error (reducible) es:

$$(f(X) - \hat{Y})^2.$$

Para un conjunto de datos X_0 , tenemos que

$$\begin{aligned}\hat{\beta} &\sim \mathcal{N}(\beta, \sigma^2((X_0^\top X_0)^{-1})) \\ \implies \hat{Y} &= \hat{X}_0\hat{\beta} \sim \mathcal{N}(\hat{X}_0\beta, \sigma^2 X_0^\top ((X_0^\top X_0)^{-1} X_0))\end{aligned}$$

Por lo tanto un **intervalo de confianza** al $1 - \alpha$ para $X\beta$ es

$$X_0\beta \pm z_{1-\frac{\alpha}{2}}\hat{\sigma}\sqrt{X_0^\top (X_0^\top X_0)^{-1} X_0}.$$

2. **Error irreducible:** Aún conociendo perfectamente los β 's, existe el error desconocido $\varepsilon \sim \mathcal{N}(0, \sigma^2)$ del modelo

$$Y = X\beta + \varepsilon.$$

Entonces la varianza total de la predicción sería

$$\sigma^2 + \sigma^2 X_0^\top ((X_0^\top X_0)^{-1} X_0)$$

Entonces un **intervalo de predicción** al $1 - \alpha$ debe tomar en cuenta ese error y por lo tanto

$$X_0\beta \pm z_{1-\frac{\alpha}{2}}\hat{\sigma}\sqrt{1 + X_0^\top (X_0^\top X_0)^{-1} X_0}.$$

Resumiendo

- **Intervalo de confianza:** es la incertidumbre que existe alrededor de la línea de regresión.
- **Intervalo de predicción:** es la incertidumbre que existe alrededor del proceso general que generaron los datos bajo el supuesto de linealidad.

5.5.1. Laboratorio

```

lm.r <- lm(mpg ~ wt, data = mtcars)

range(mtcars$wt)

## [1] 1.513 5.424

(datos_nuevos <- data.frame(wt = c(2.5, 3, 3.5)))

##      wt
## 1 2.5
## 2 3.0
## 3 3.5

predict(object = lm.r, newdata = datos_nuevos, interval = "confidence")

##          fit      lwr      upr
## 1 23.92395 22.55284 25.29506
## 2 21.25171 20.12444 22.37899
## 3 18.57948 17.43342 19.72553

predict(object = lm.r, newdata = datos_nuevos, interval = "prediction")

##          fit      lwr      upr
## 1 23.92395 17.55411 30.29378
## 2 21.25171 14.92987 27.57355
## 3 18.57948 12.25426 24.90469

```

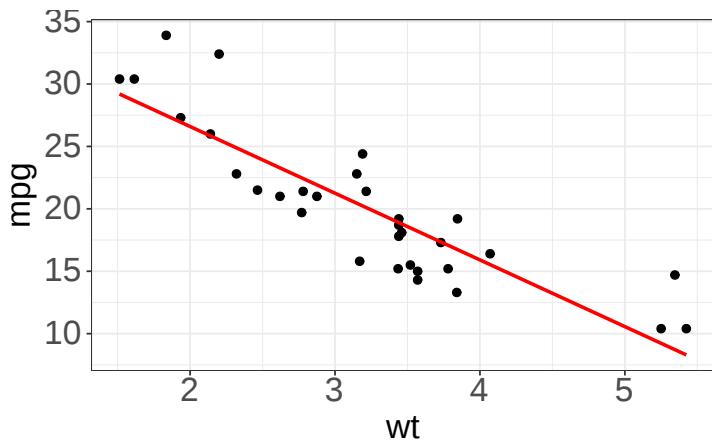
5.5.1.1. Ajuste de la regresión sin intervalos de confianza

```

p <- ggplot(mtcars, aes(x = wt, y = mpg))
p <- p + geom_point(size = 2)           # Use círculos de tamaño 2
p <- p + geom_smooth(method = lm,       # Agregar la línea de regresión
                      se = FALSE,      # NO incluir el intervalo de confianza
                      size = 1,
                      col = "red")      # Línea de color rojo
p <- p + theme_bw()                    # Tema de fondo blanco
p <- p + theme(axis.text = element_text(size = 20), # Aumentar el tamaño
               axis.title = element_text(size = 20)) # de letra en los ejes

# Dibujar el gráfico
p

```



```

# # Guardar el gráfico en un archivo pdf
# ggsave(filename = 'linear_reg_sin_IC.pdf') #

```

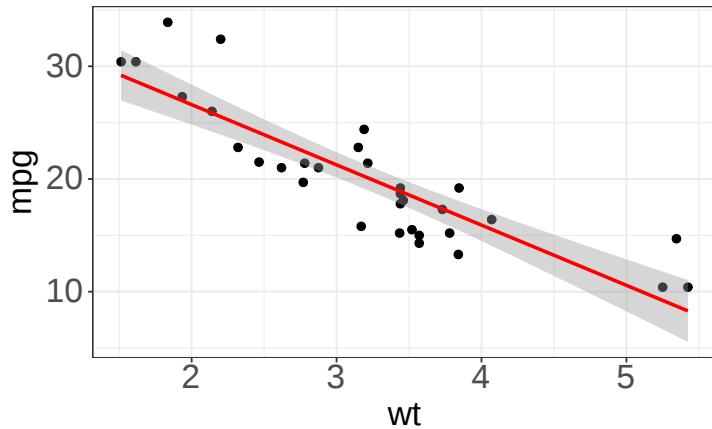
5.5.1.2. Ajuste de la regresión con intervalos de confianza

```

p <- ggplot(mtcars, aes(x = wt, y = mpg))
p <- p + geom_point(size = 2)           # Use círculos de tamaño 2
p <- p + geom_smooth(method = lm,       # Agregar la línea de regresión
                      se = TRUE,        # Incluir el intervalo de confianza
                      size = 1,
                      col = "red")      # Línea de color rojo
p <- p + theme_bw()                    # Tema de fondo blanco
p <- p + theme(axis.text = element_text(size = 20), # Aumentar el tamaño
               axis.title = element_text(size = 20)) # de letra en los ejes

# Dibujar el gráfico
p

```



```

# # Guardar el gráfico en un archivo pdf
# ggsave(filename = 'linear_reg_con_IC.pdf') #

```

5.5.1.3. Ajuste de la regresión con intervalos de confianza y predicción

```

# Agregamos a mtcars el intervalo de predicción
# para cada dato
mtcars.pred <- data.frame(mtcars, predict(lm.r, interval = "prediction"))

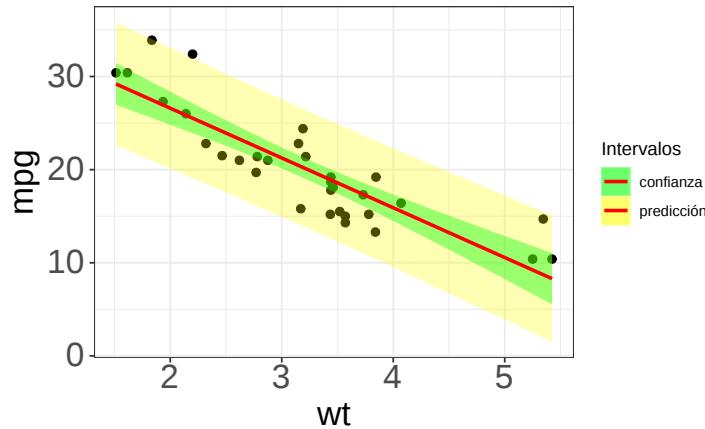
```

```

p <- ggplot(mtcars.pred, aes(x = wt, y = mpg))
# Use círculos de tamaño 2
p <- p + geom_point(size = 2)
# Agregue una banda de tamaño [lwr, upr] para cada
# punto y llame la 'predicción'
p <- p + geom_ribbon(aes(ymin = lwr, ymax = upr, fill = "predicción"),
                      alpha = 0.3)
# Agregue el intervalo de confianza usual y llame a
# ese intervalo 'confianza'
p <- p + geom_smooth(method = lm, aes(fill = "confianza"),
                      size = 1, col = "red")
# Para agregar bien las leyendas
p <- p + scale_fill_manual("Intervalos", values = c("green",
                                                    "yellow"))
p <- p + theme_bw()
p <- p + theme(axis.text = element_text(size = 20),
                axis.title = element_text(size = 20))

# Dibujar el gráfico
p

```



```

# # Guardar el gráfico en un archivo pdf
# ggsave(filename = 'linear_reg_con_IC_IP.pdf') #

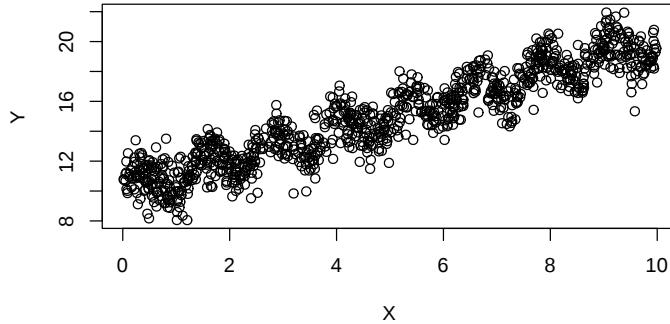
```

Repitamos el mismo ejercicio anterior pero con un caso más sencillo.

```
n <- 1000

X <- runif(n, 0, 10)
Y <- 10 + sin(5 * X) + X + rnorm(1000, 0, 1)
toyex.initial <- data.frame(X, Y) %>% arrange(X)

plot(toyex.initial)
```



```
lm.toyex.initial <- lm(Y ~ X, data = toyex.initial)

summary(lm.toyex.initial)

## 
## Call:
## lm(formula = Y ~ X, data = toyex.initial)
## 
## Residuals:
##      Min       1Q   Median       3Q      Max 
## -4.1440 -0.8238  0.0088  0.8823  3.0482 
## 
## Coefficients:
##             Estimate Std. Error t value    Pr(>|t|)    
## (Intercept) 10.0000    0.0088 1124.44   <2e-16 ***
## X            1.0000    0.0088 112.73   <2e-16 ***
```

```

## (Intercept) 10.11325    0.07220   140.07 <0.0000000000000002 ***
## X           0.97761    0.01269    77.02 <0.0000000000000002 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.186 on 998 degrees of freedom
## Multiple R-squared:  0.856, Adjusted R-squared:  0.8558
## F-statistic:  5932 on 1 and 998 DF,  p-value: < 0.00000000000000022

toyex.pred.initial <- data.frame(toyex.initial, predict(lm.toyex.initial,
  interval = "prediction"))

```

Ahora, quisiera generar muchas muestras del mismo experimento

```

toyex.pred <- NULL

for (i in 1:10) {
  X <- runif(n, 0, 10)
  Y <- 10 + sin(5 * X) + X + rnorm(1000, 0, 1)
  toyexi <- data.frame(im = i, X, Y)
  toyexi <- toyexi %>% arrange(X)
  toyex.pred <- bind_rows(toyex.pred, data.frame(toyexi,
    predict(lm.toyex.initial, interval = "prediction")))
}

for (i in 1:10) {
  toyex.pred$fit <- fitted(lm(formula = Y ~ X, data = toyex.pred[toyex.pred$im ==
    i, ]))
}

toyex.pred$im <- as.factor(toyex.pred$im)

library(gganimate)

ggplot(data = toyex.pred, aes(x = X, y = Y)) + geom_point(size = 1) +
  geom_smooth(data = toyex.initial, method = lm,

```

```

mapping = aes(fill = "confianza"), size = 1,
col = "red") + geom_ribbon(data = toyex.pred.initial,
mapping = aes(x = X, ymin = lwr, ymax = upr, fill = "predicción",
), alpha = 0.3) + labs(title = paste0("Muestra #: {closest_state}"))
scale_fill_manual("Intervalos", values = c("green",
"yellow")) + theme_bw() + theme(axis.text = element_text(size = 20),
axis.title = element_text(size = 20)) + transition_states(im)

```

5.6. Interacciones

En el modelo clásico

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \varepsilon$$

Aumentemos en 1 unidad X_1 y rescribamos el modelo original

$$\begin{aligned} Y &= \beta_0 + \beta_1(X_1 + 1) + \beta_2 X_2 + \varepsilon \\ Y &= (\beta_0 + \beta_1) + \beta_1 X_1 + \beta_2 X_2 + \varepsilon \\ Y &= \tilde{\beta}_0 + \beta_1 X_1 + \beta_2 X_2 + \varepsilon \end{aligned}$$

Es decir, el modelo original sigue siendo el mismo aunque hayamos cambiado el X_1 . Este fenómeno ocurre siempre bajo transformaciones lineales de las variables.

Ahora suponga que tenemos el siguiente modelo y aumentamos en 1 el X_1

$$\begin{aligned} Y &= \beta_0 + \tilde{\beta}_1 X_1 X_2 + \varepsilon \\ \implies Y &= \beta_0 + \beta_1(X_1 + 1)X_2 + \varepsilon \\ \implies Y &= \beta_0 + \beta_1 X_2 + \beta_1 X_1 X_2 + \varepsilon \end{aligned}$$

OJO. Terminamos con un modelo diferente con el que empezamos. Esto es indeseable ya que no hay consistencia en la modelación,

Una forma de arreglar el problema es incluir las interacciones junto con todos sus efectos directos.

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_1 X_2 + \varepsilon$$

Esto se le conoce como principio de jerarquía. No es importante si los efectos directos son relevante o no dentro del modelo, siempre se deben de incluir para manter la consistencia.

Ejercicio 5.3. Compruebe que para el caso anterior, si aumenta en una unidad X_1 , el modelo se mantiene.

5.6.1. Laboratorio

Generamos una base de datos nueva con solamente `wt` centrado

```
# La función across y where solo funciona solo para
# dplyr 1.0 Si tienen otra versión, pueden usar
# mutate_if

mtcars_centered <- mtcars %>% mutate(across("wt", scale,
    scale = FALSE, center = TRUE))

# Si no se tiene dplyr 1.0

mtcars_centered <- mtcars %>% mutate_at("wt", scale,
    scale = FALSE, center = TRUE)
```

Compare lo que ocurre con los coeficientes de la base original y la nueva base.

```
summary(lm(mpg ~ wt + disp, data = mtcars))
```

```
## 
## Call:
```

```

## lm(formula = mpg ~ wt + disp, data = mtcars)
##
## Residuals:
##   Min     1Q Median     3Q    Max
## -3.4087 -2.3243 -0.7683  1.7721  6.3484
##
## Coefficients:
##             Estimate Std. Error t value      Pr(>|t|)
## (Intercept) 34.96055   2.16454 16.151 0.0000000000000491 ***
## wt          -3.35082   1.16413 -2.878   0.00743 **
## disp         -0.01773   0.00919 -1.929   0.06362 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.917 on 29 degrees of freedom
## Multiple R-squared:  0.7809, Adjusted R-squared:  0.7658
## F-statistic: 51.69 on 2 and 29 DF,  p-value: 0.000000002744

```

```
summary(lm(mpg ~ wt + disp, data = mtcars_centered))
```

```

##
## Call:
## lm(formula = mpg ~ wt + disp, data = mtcars_centered)
##
## Residuals:
##   Min     1Q Median     3Q    Max
## -3.4087 -2.3243 -0.7683  1.7721  6.3484
##
## Coefficients:
##             Estimate Std. Error t value      Pr(>|t|)
## (Intercept) 24.18011   2.18221 11.081 0.0000000000612 ***
## wt          -3.35082   1.16413 -2.878   0.00743 **
## disp         -0.01773   0.00919 -1.929   0.06362 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.917 on 29 degrees of freedom
```

```
## Multiple R-squared:  0.7809, Adjusted R-squared:  0.7658
## F-statistic: 51.69 on 2 and 29 DF,  p-value: 0.0000000002744
```

Supongamos que formamos un modelo con solo la interacción y no incluimos los efectos directos.

```
summary(lm(mpg ~ wt * disp - wt - disp, data = mtcars))

##
## Call:
## lm(formula = mpg ~ wt * disp - wt - disp, data = mtcars)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -4.259 -2.603 -1.657  2.165  8.589
##
## Coefficients:
##             Estimate Std. Error t value            Pr(>|t|)
## (Intercept) 26.2621926  1.0418029 25.208 < 0.000000000000002 ***
## wt:disp     -0.0072897  0.0009721 -7.499  0.000000233 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.614 on 30 degrees of freedom
## Multiple R-squared:  0.6521, Adjusted R-squared:  0.6405
## F-statistic: 56.24 on 1 and 30 DF,  p-value: 0.00000002329

summary(lm(mpg ~ wt * disp - wt - disp, data = mtcars_centered))

##
## Call:
## lm(formula = mpg ~ wt * disp - wt - disp, data = mtcars_centered)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -5.878 -2.775 -1.162  2.409 11.150
```

```

## 
## Coefficients:
##             Estimate Std. Error t value     Pr(>|t|)    
## (Intercept) 21.460008   0.859706  24.962 < 0.0000000000000002 ***
## wt:disp     -0.013127   0.002714  -4.837      0.0000369 ***
## ---        
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## Residual standard error: 4.592 on 30 degrees of freedom
## Multiple R-squared:  0.4382, Adjusted R-squared:  0.4195 
## F-statistic:  23.4 on 1 and 30 DF,  p-value: 0.00003686

```

El modelo correcto sería el siguiente:

```
summary(lm(mpg ~ wt + disp + wt * disp, data = mtcars))
```

```

## 
## Call:
## lm(formula = mpg ~ wt + disp + wt * disp, data = mtcars)
## 
## Residuals:
##    Min     1Q Median     3Q    Max 
## -3.267 -1.677 -0.836  1.351  5.017 
## 
## Coefficients:
##             Estimate Std. Error t value     Pr(>|t|)    
## (Intercept) 44.081998   3.123063 14.115 0.000000000000296 ***
## wt          -6.495680   1.313383 -4.946 0.0000321670456650 ***
## disp        -0.056358   0.013239 -4.257      0.00021 ***  
## wt:disp      0.011705   0.003255  3.596      0.00123 ** 
## ---        
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## Residual standard error: 2.455 on 28 degrees of freedom
## Multiple R-squared:  0.8501, Adjusted R-squared:  0.8341 
## F-statistic: 52.95 on 3 and 28 DF,  p-value: 0.00000000001158

```

```
summary(lm(mpg ~ wt + disp + wt * disp, data = mtcars_centered))

##
## Call:
## lm(formula = mpg ~ wt + disp + wt * disp, data = mtcars_centered)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -3.267 -1.677 -0.836  1.351  5.017
##
## Coefficients:
##             Estimate Std. Error t value     Pr(>|t|)
## (Intercept) 23.183772   1.857605 12.480 0.00000000000587 ***
## wt          -6.495680   1.313383 -4.946 0.000032167045665 ***
## disp        -0.018699   0.007741 -2.416      0.02248 *
## wt:disp      0.011705   0.003255  3.596      0.00123 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.455 on 28 degrees of freedom
## Multiple R-squared:  0.8501, Adjusted R-squared:  0.8341
## F-statistic: 52.95 on 3 and 28 DF,  p-value: 0.00000000001158
```

Ejercicio 5.4. Repita los comandos anteriores con la siguiente base de datos y explique los resultados.

```
mtcars_scaled <- mtcars %>% mutate(across(c("wt", "disp"),
  scale, scale = TRUE, center = TRUE))
```

5.7. Hipótesis en regresión lineal

Hasta ahora hemos visto el modelo de regresión como un conjunto de partes separadas.

5.7.1. Hipotésis

Independencia lineal El supuesto es que el modelo es lineal.

Errores con esperanza nula Esto quiere decir que $\mathbb{E}(\varepsilon_i) = 0$.

Homocedasticidad $\text{Var}(\varepsilon_t) = \mathbb{E}(\varepsilon_t - \mathbb{E}\varepsilon_t)^2 = \mathbb{E}\varepsilon_t^2 = \sigma^2$ para todo t . Es decir, la varianza del modelo no depende de las variables independientes u otro factor. En otras palabras, el **error irreducible** es completamente ajeno a las variables independientes del modelo.

Normalidad de los residuos $\varepsilon \sim N(0, \sigma^2)$.

Independencia de los errores $\text{Cov}(\varepsilon_t, \varepsilon_s) = \mathbb{E}(\varepsilon_t - \mathbb{E}\varepsilon_t)(\varepsilon_s - \mathbb{E}\varepsilon_s) = \mathbb{E}\varepsilon_t\varepsilon_s = 0$ para todo t, s con $t \neq s$. Esto es una extensión del supuesto anterior y quiere decir, que además de los errores no depende de las variables, tampoco pueden depender entre si. Es decir, si para una observación dada existe un error, este no debe depender del error de otra observación.

Esto puede provocar que los errores usados para intervalos de confianza y predicción sean subestimados. Es decir que un intervalo del 95 % tendrá menos confianza y se rechazaría más fácilmente la hipótesis nula de las pruebas t y F .

Multicolineidad Se asume que cada una de las variables es independiente de las otras. Es decir que cada variable explica «un aspecto o característica» del modelo. Sin embargo puede pasar que varias variables expliquen la misma característica y el modelo tenga que volverse **inestable** por decidir entre las dos variables. Por ejemplo: la temperatura en grados centígrados y fareheit.

En este caso habría dos columnas linealmente dependientes y por lo tanto $(X^\top X)^{-1}$ se acercaría a una matriz singular con determinante cercano a 0.

Esto generaría que $\text{Var}(\beta)$ sea alto ya que

$$\beta = (X^\top X)^{-1} X^\top Y.$$

Más observaciones que predictores En este caso siempre podremos construir correctamente la regresión y sus índices. (Volveremos a esto cuando veamos selección de modelos)

5.7.2. Chequeos básicos de las hipótesis de regresión lineal

5.7.2.1. Independencia lineal, Errores con esperanza nula, Homocedasticidad

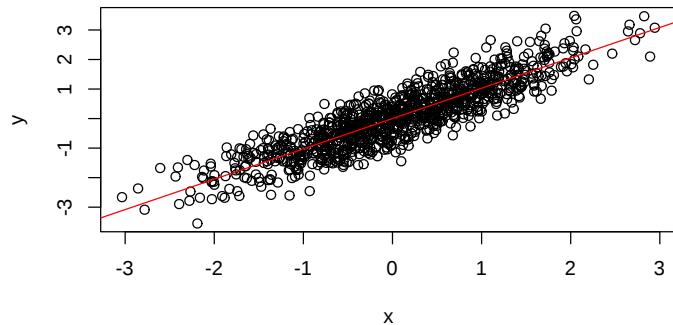
Estos supuestos se puede constatar a partir de un gráfico de residuos ya que en el caso ideal $e_i = \hat{Y}_i - Y_i \perp \hat{Y}_i$. Entonces si este gráfico presenta patrones, quiere indicar que la regresión, no es lineal, que los errores no tienen esperanza nula y que la varianza no es constante.

Se pueden aplicar transformaciones para resolver estos problemas. Normalmente se usan transformaciones como raíz cuadrada o logaritmos.

Ejemplo 5.2. Caso ideal

```
x <- rnorm(1000)
y <- x + rnorm(1000, sd = 0.5)

fit <- lm(y ~ x)
plot(x, y)
abline(a = coef(fit)[1], b = coef(fit)[2], col = "red")
```



```
plot(fitted(fit), residuals(fit))
abline(h = 0, col = "red")
```

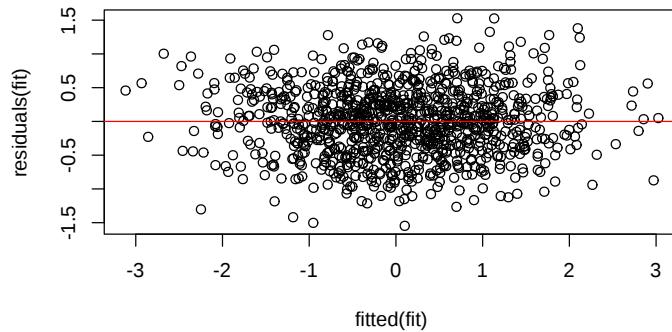
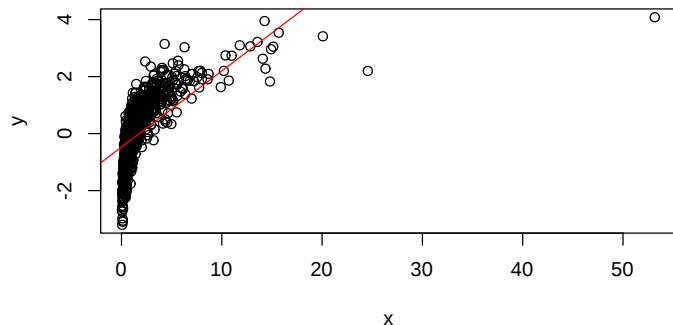


Figura 5.2: Gráfico de residuos caso lineal

Caso no-lineal

```
x <- exp(rnorm(1000))
y <- log(x) + rnorm(1000, sd = 0.5)

fit <- lm(y ~ x)
plot(x, y)
abline(a = coef(fit)[1], b = coef(fit)[2], col = "red")
```



```
plot(fitted(fit), residuals(fit))
abline(h = 0, col = "red")
```

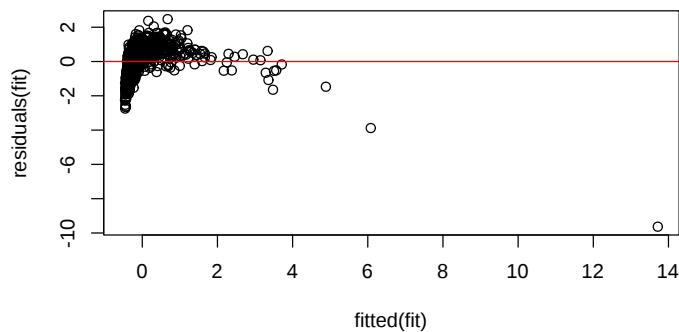


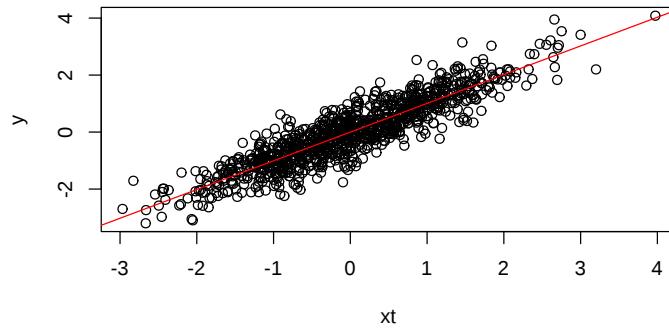
Figura 5.3: Gráfico de residuos caso no-lineal

Caso no-lineal transformado

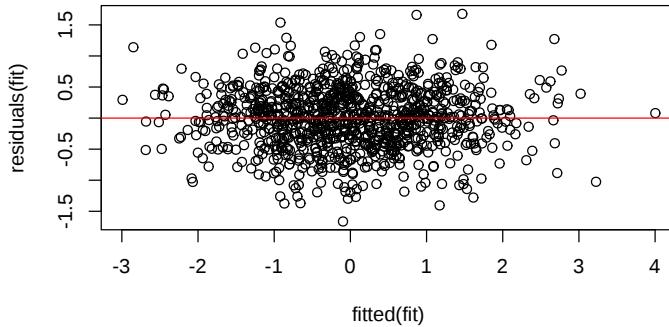
```
xt <- log(x)

fit <- lm(y ~ xt)
```

```
plot(xt, y)
abline(a = coef(fit)[1], b = coef(fit)[2], col = "red")
```



```
plot(fitted(fit), residuals(fit))
abline(h = 0, col = "red")
```



5.7.2.2. Independencia de los errores

En este caso defina $\rho(k) = \text{Cov}(\varepsilon_i, \varepsilon_{i+k})$. Si los residuos son independientes, entonces debe ocurrir que

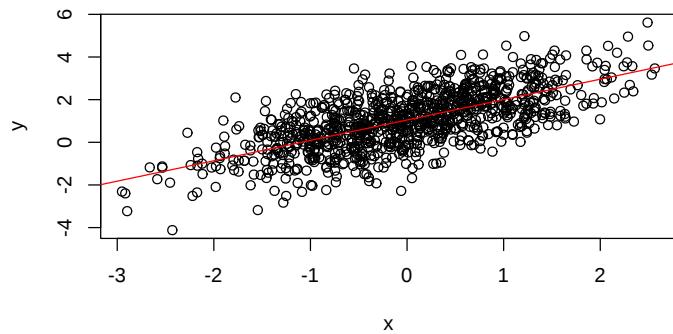
$$\rho(k) = \begin{cases} 1 & k = 0 \\ 0 & k \neq 0. \end{cases}$$

Se calcula la función de autocorrelación y se gráfica para analizar su comportamiento

Caso ideal

```
x <- rnorm(1000)
y <- 1 + x + rnorm(1000, sd = 1)
```

```
fit <- lm(y ~ x)
plot(x, y)
abline(a = coef(fit)[1], b = coef(fit)[2], col = "red")
```



```
summary(fit)
```

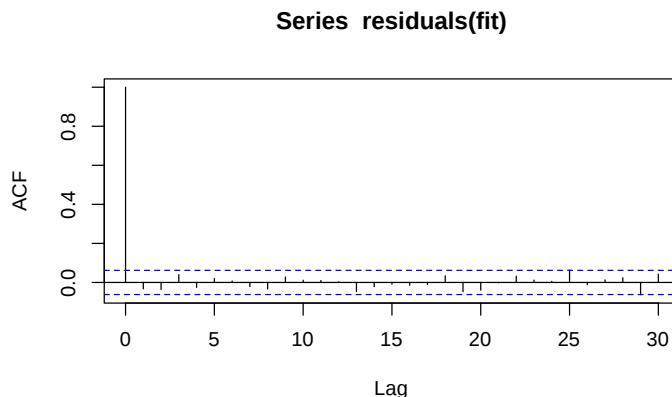
```
##
## Call:
## lm(formula = y ~ x)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -3.0000 -0.5000  0.0000  0.5000  3.0000
```

```

## -3.2665 -0.6871  0.0002  0.6670  2.9410
##
## Coefficients:
##             Estimate Std. Error t value     Pr(>|t|)
## (Intercept) 1.04643   0.03183 32.87 <0.0000000000000002 ***
## x           0.95650   0.03287 29.10 <0.0000000000000002 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.007 on 998 degrees of freedom
## Multiple R-squared:  0.4589, Adjusted R-squared:  0.4584
## F-statistic: 846.6 on 1 and 998 DF,  p-value: < 0.0000000000000002

```

`acf(residuals(fit))`



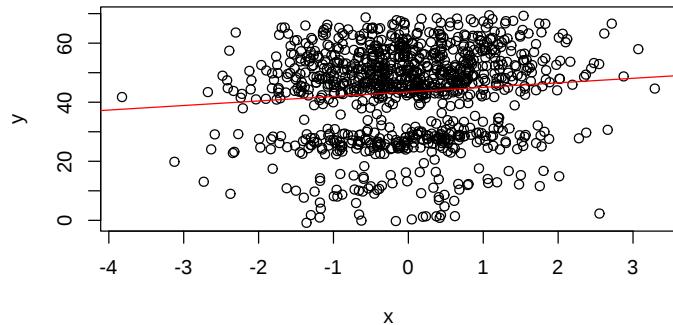
Caso errores auto-correlacionados

```

x <- rnorm(1000)
y <- 1 + x + diffinv(rnorm(999, sd = 1), lag = 1)

fit <- lm(y ~ x)
plot(x, y)
abline(a = coef(fit)[1], b = coef(fit)[2], col = "red")

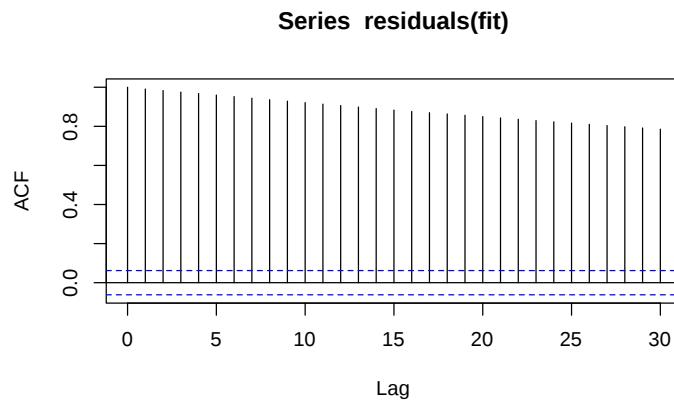
```



```
summary(fit)
```

```
##
## Call:
## lm(formula = y ~ x)
##
## Residuals:
##    Min     1Q   Median     3Q    Max
## -45.109 -13.583   3.439  11.036  26.104
##
## Coefficients:
##             Estimate Std. Error t value            Pr(>|t|)
## (Intercept) 43.4918    0.4791  90.787 < 0.0000000000000002 ***
## x           1.5347    0.4771   3.217      0.00134 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 15.15 on 998 degrees of freedom
## Multiple R-squared:  0.01026,    Adjusted R-squared:  0.00927
## F-statistic: 10.35 on 1 and 998 DF,  p-value: 0.001339
```

```
acf(residuals(fit))
```



5.7.2.3. Normalidad de los errores

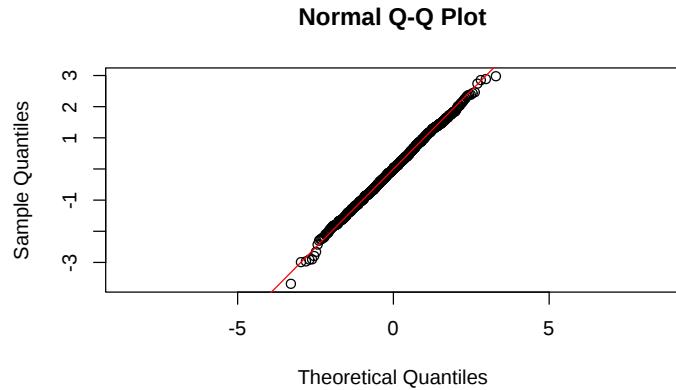
Este hipótesis es crucial para hacer las pruebas t y F que vimos anteriormente.

Para revisar si se cumple solo basta hacer una `qqplot` de los residuos.

Caso ideal

```
x <- rnorm(1000)
y <- 1 + x + rnorm(1000, sd = 1)
fit <- lm(y ~ x)
```

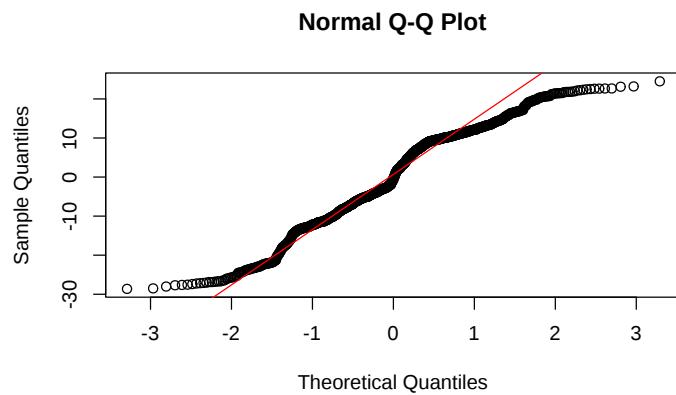
```
qqnorm(residuals(fit), asp = 1)
qqline(residuals(fit), col = "red")
```



Caso errores auto-correlacionados

```
x <- rnorm(1000)
y <- 1 + x + diffinv(rnorm(999, sd = 1), lag = 1)
fit <- lm(y ~ x)

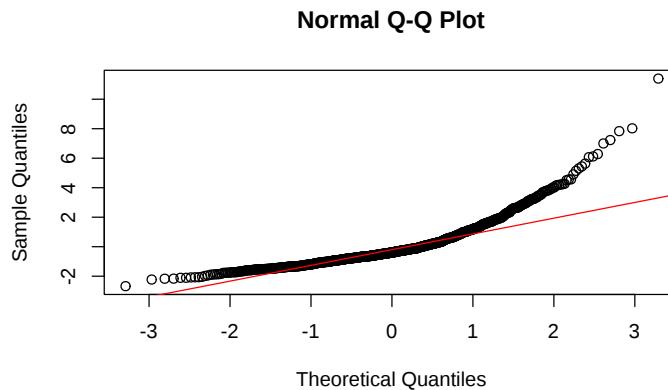
qqnorm(residuals(fit), asp = 0)
qqline(residuals(fit), col = "red")
```



Caso no-lineal

```
x <- rnorm(1000)
y <- x^2 + rnorm(1000, sd = 0.5)
fit <- lm(y ~ x)
```

```
qqnorm(residuals(fit), asp = 0)
qqline(residuals(fit), col = "red")
```



```
x <- rnorm(1000)
y <- x^2 + rnorm(1000, sd = 0.5)
fit <- lm(y ~ x + I(x^2))
summary(fit)
```

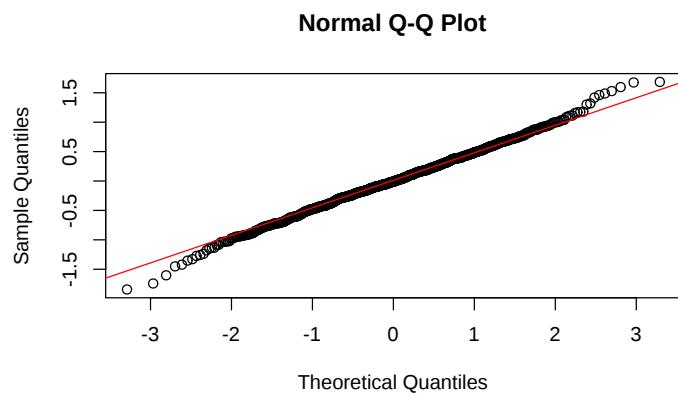
```
##
## Call:
## lm(formula = y ~ x + I(x^2))
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.84371 -0.30372 -0.01256  0.32728  1.68466
##
## Coefficients:
##             Estimate Std. Error t value            Pr(>|t|)
## (Intercept) 0.01010   0.01979   0.511           0.6098
```

```

## x           -0.03009   0.01587  -1.896          0.0582 .
## I(x^2)      0.99172   0.01196  82.906 <0.0000000000000002 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.5001 on 997 degrees of freedom
## Multiple R-squared:  0.8734, Adjusted R-squared:  0.8731
## F-statistic:  3438 on 2 and 997 DF,  p-value: < 0.0000000000000022

qqnorm(residuals(fit), asp = 0)
qqline(residuals(fit), col = "red")

```



5.7.2.4. Multicolinealidad

Hay dos formas de detectar multicolinealidad

1. Analizar la matriz de correlaciones de las variables (solamente detecta colinealidad entre pares).
2. Analizar la correlación multiple entre un predictor y el resto.

Defina $R_{X_j|X_{-j}}^2$ como el R^2 de la regresión multiple entre X_j vs el resto de covariables.

Si $R^2_{X_j|X_{-j}}$ es cercano a 1 entonces hay alta correlación entre X_j y el resto.

Defina el factor de inflación de la varianza como:

$$\text{VIF}(\hat{\beta}_j) = \frac{1}{1 - R^2_{X_j|X_{-j}}}$$

Si VIF es alto

- Quitar las variables
- Combinar variables

Hay muchos paquetes que tienen implementado la función `vif` (car, rms, entre otros).

Caso variables colineales

La variable `wt` está en unidades de 1000lb. La convertimos a Kilogramos.

```
mtcars_kg <- mtcars %>% mutate(wt_kg = wt * 1000 *
  0.4535 + rnorm(32))

fit_kg <- lm(mpg ~ disp + wt + wt_kg, data = mtcars_kg)
summary(fit_kg)

##
## Call:
## lm(formula = mpg ~ disp + wt + wt_kg, data = mtcars_kg)
##
## Residuals:
##      Min       1Q   Median       3Q      Max 
## -4.0609 -1.8566 -0.6442  1.1658  6.1471 
##
## Coefficients:
##             Estimate Std. Error t value            Pr(>|t|)    
## (Intercept) 36.263681   2.143546 16.918 0.0000000000000311 ***
## disp        -0.016980   0.008712 -1.949           0.0614 .
```

```

## wt           455.378192 220.448899   2.066          0.0482 *
## wt_kg        -1.012338   0.486488  -2.081          0.0467 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.762 on 28 degrees of freedom
## Multiple R-squared:  0.8103, Adjusted R-squared:  0.7899
## F-statistic: 39.86 on 3 and 28 DF,  p-value: 0.0000000003079

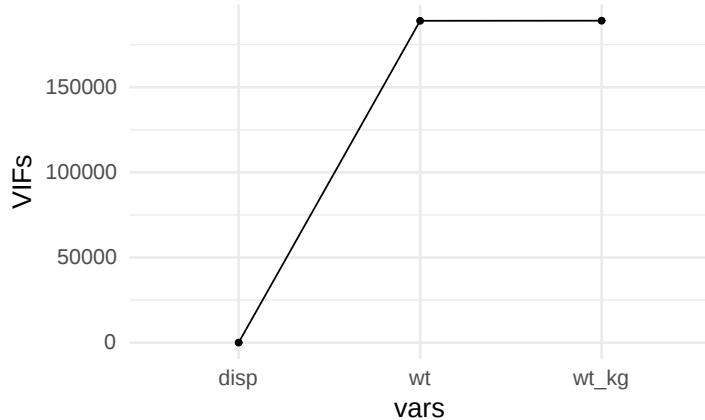
library(car)
options(scipen = 1000)

VIFs <- vif(fit_kg)

VIFs <- as.data.frame(VIFs) %>% rownames_to_column(var = "vars")

ggplot(VIFs, aes(x = vars, y = VIFs, group = 1)) +
  geom_point() + geom_line() + theme_minimal(base_size = 16)

```



5.7.3. Otros chequeos importantes

5.7.3.1. Puntos extremos

Estos puntos son aquellos que Y_i esta lejos de \hat{Y}_i . Otra forma de verlo son aquellos puntos que tienen residuos muy altos.

Se puede hacer un gráfico de los residuos vs los valores ajustados como en 5.2 y 5.3.

¿Qué tan grande deben ser los residuos?

Solución: Se debe escalar los residuos adecuadamente.

Se construyen los residuos semi-studentizados

$$r_i^s = \frac{e_i}{\sqrt{\text{Var}(e_i)}}$$

Como $H = X(X^\top X)^{-1}X^\top$ es la matriz de proyección entonces sabemos que

$$\begin{aligned}\hat{Y} &= HY \\ e &= Y - \hat{Y}\end{aligned}$$

Entonces tenemos que

$$\begin{aligned}\text{Var}(e) &= \text{Var}((I - H)Y) \\ &= (I - H)^2 \text{Var}(Y) \\ &= (I - H)\sigma^2 \quad (I - H \text{ es idempotente})\end{aligned}$$

Por lo tanto

$$\text{Var}(e_i) = (1 - h_{ii})\sigma^2$$

Para cada observación se calcula los residuos de la forma

$$r_i^s = \frac{e_i}{\sqrt{(1 - h_{ii})\sigma^2}}$$

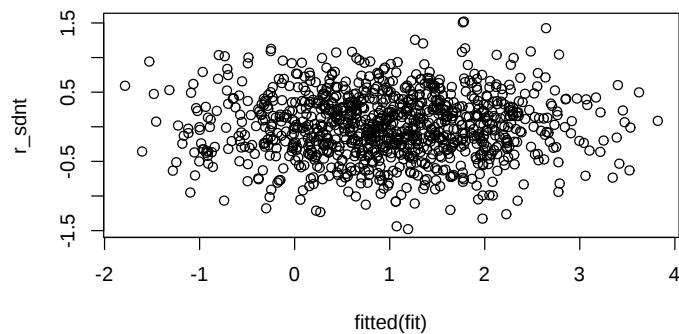
Caso sin valores extremos

```

x <- rnorm(1000)
y <- 1 + x + rnorm(1000, sd = 0.5)
fit <- lm(y ~ x)

X <- model.matrix(y ~ x)
H <- X %*% solve(t(X) %*% X) %*% t(X)
I <- diag(1, nrow = 1000)
I_H <- I - H
r_sdnt <- residuals(fit)/sqrt(diag(I_H) * var(y))
plot(fitted(fit), r_sdnt)

```



```
fit
```

```

##
## Call:
## lm(formula = y ~ x)
##
## Coefficients:
## (Intercept)          x
##       0.9893        0.9701

```

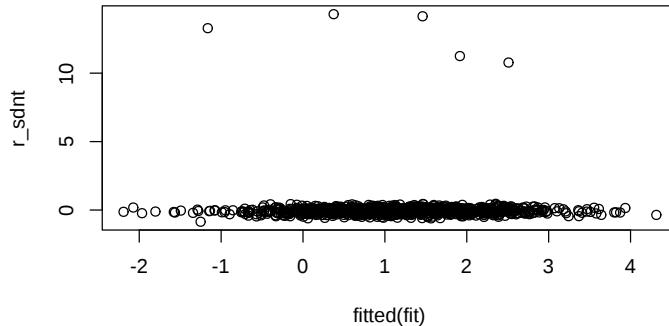
**Caso con valores extremos*

```

x <- rnorm(1000)
y <- 1 + x + rnorm(1000, sd = 0.5)
y[1:5] <- runif(5, 30, 40)
fit <- lm(y ~ x)

X <- model.matrix(y ~ x)
H <- X %*% solve(t(X) %*% X) %*% t(X)
I <- diag(1, nrow = 1000)
I_H <- I - H
r_sdnt <- residuals(fit)/sqrt(diag(I_H) * var(y))
plot(fitted(fit), r_sdnt)

```



fit

```

##
## Call:
## lm(formula = y ~ x)
##
## Coefficients:
## (Intercept)          x
##       1.1505        0.9397

```

5.7.3.2. Puntos de apalancamiento (leverage)

Un outlier puede ser detectado pero aún así este puede no afectar el modelo como un todo.

El r_i^s puede ser alto por 2 razones:

1. los residuos e_i son altos (un outlier)
2. el valor h_{ii} es cercano a 1. (Se tiene que $0 \leq h_{ii} \leq 1$).

Los valores donde $h_{ii} \approx 1$ se les denomina de **gran apalancamiento**.

La regla empírica dice que

$$\sum_{i=1}^n h_{ii} = p + 1 \text{ (Los predictores más el intercepto)}$$

Regla empírica: Si $h_{ii} > \frac{p+1}{n}$ entonces decimos que el punto de **gran apalancamiento**.

Distancia de Cook. La distancia de Cook mide la influencia de las observaciones con respecto al ajuste del modelo lineal con p variables. Esta se define como:

$$D_i = \frac{\sum_{j=1}^n (\hat{Y}_j - \hat{Y}_{j(-i)})^2}{(p+1)\sigma^2}$$

donde $\hat{Y}_{j(-i)}$ significa el ajuste del modelo lineal, removiendo la observación i -ésima.

Caso base

```
set.seed(42)
apa_df = data.frame(x = 1:10, y = 10:1 + rnorm(n = 10))
```

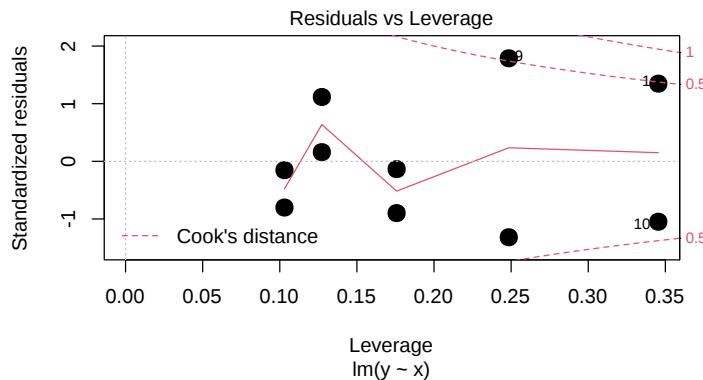
```

modelo <- lm(y ~ x, data = apa_df)
coef(modelo)

## (Intercept)           x
## 11.3801152 -0.9696033

plot(modelo, 5, col = c(rep("black", 10), "red"), cex = 2,
      pch = 16)

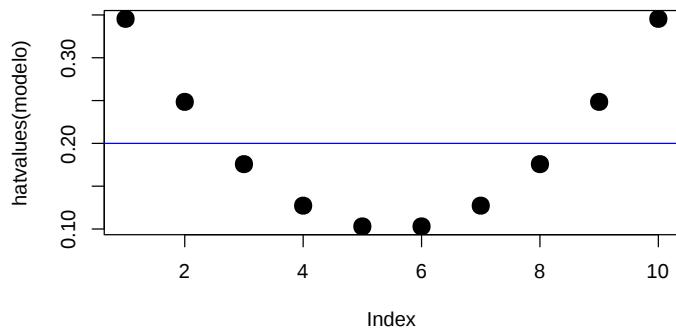
```



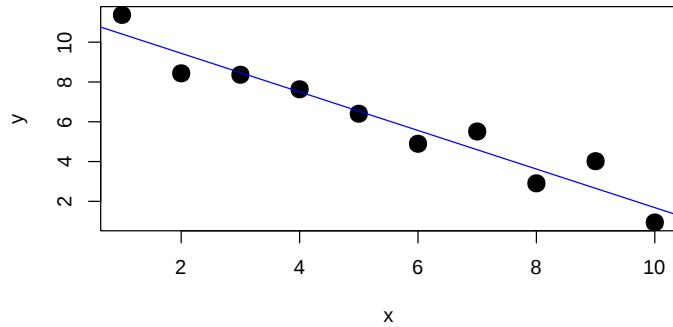
```

plot(hatvalues(modelo), col = c(rep("black", 10), "red"),
      cex = 2, pch = 16)
abline(h = 2/10, col = "blue")

```



```
plot(apa_df, col = c(rep("black", 10), "red"), cex = 2,
      pch = 16)
abline(a = coef(modelo)[1], b = coef(modelo)[2], col = "blue")
```

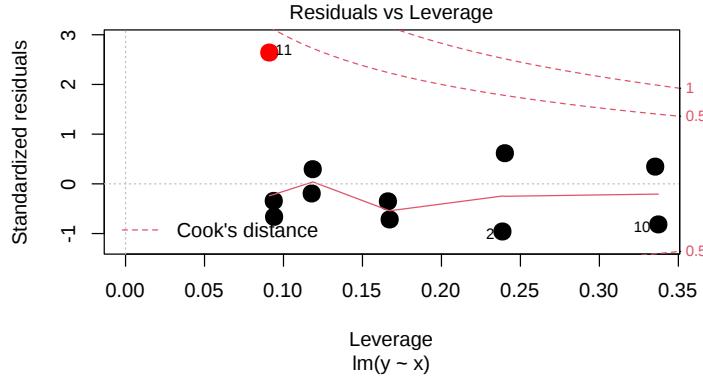


Bajo apalancamiento, residuos grandes, influencia pequeña

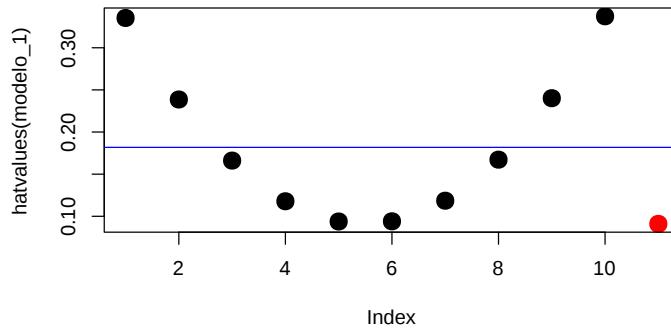
```
p_1 <- c(5.4, 11)
apa_df_1 <- rbind(apa_df, p_1)
modelo_1 <- lm(y ~ x, data = apa_df_1)
coef(modelo_1)
```

```
## (Intercept)          x
## 11.8509232 -0.9749534
```

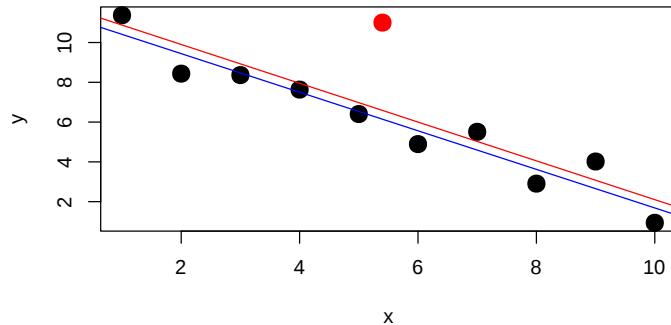
```
plot(modelo_1, 5, col = c(rep("black", 10), "red"),
      cex = 2, pch = 16)
```



```
plot(hatvalues(modelo_1), col = c(rep("black", 10),
  "red"), cex = 2, pch = 16)
abline(h = 2/11, col = "blue")
```



```
plot(apa_df_1, col = c(rep("black", 10), "red"), cex = 2,
  pch = 16)
abline(a = coef(modelo)[1], b = coef(modelo)[2], col = "blue")
abline(a = coef(modelo_1)[1], b = coef(modelo_1)[2],
  col = "red")
```

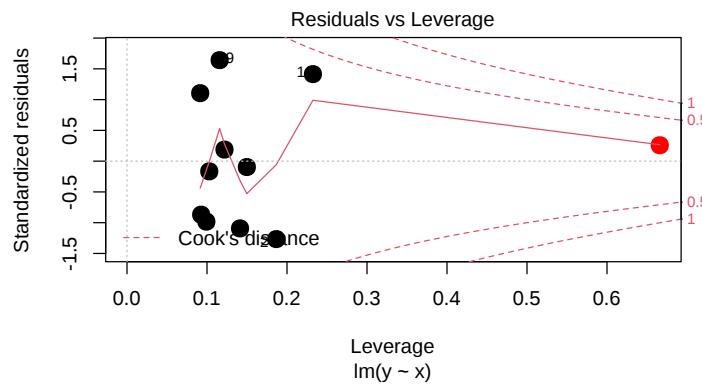


Alto apalancamiento, residuo pequeño, influencia pequeña

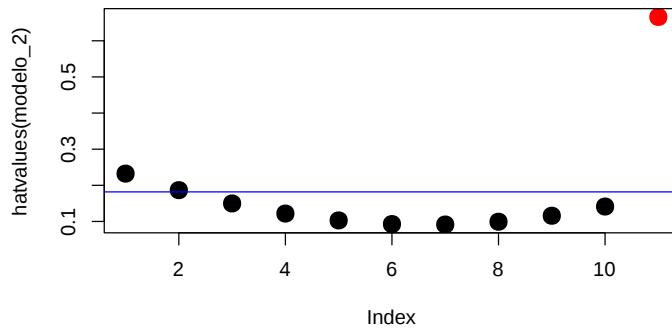
```
p_2 <- c(18, -5.7)
apa_df_2 <- rbind(apa_df, p_2)
modelo_2 <- lm(y ~ x, data = apa_df_2)
coef(modelo_2)
```

```
## (Intercept)          x
## 11.2888153 -0.9507397
```

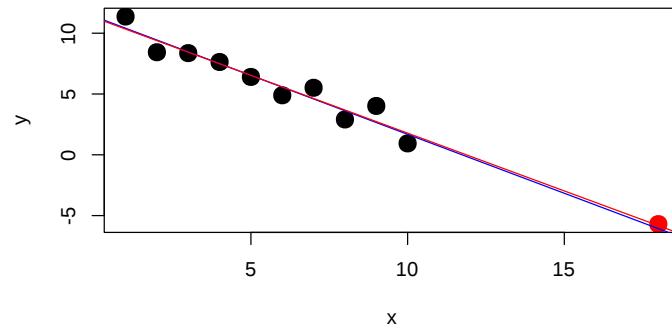
```
plot(modelo_2, 5, col = c(rep("black", 10), "red"),
     cex = 2, pch = 16)
```



```
plot(hatvalues(modelo_2), col = c(rep("black", 10),
  "red"), cex = 2, pch = 16)
abline(h = 2/11, col = "blue")
```



```
plot(apa_df_2, col = c(rep("black", 10), "red"), cex = 2,
  pch = 16)
abline(a = coef(modelo)[1], b = coef(modelo)[2], col = "blue")
abline(a = coef(modelo_2)[1], b = coef(modelo_2)[2],
  col = "red")
```

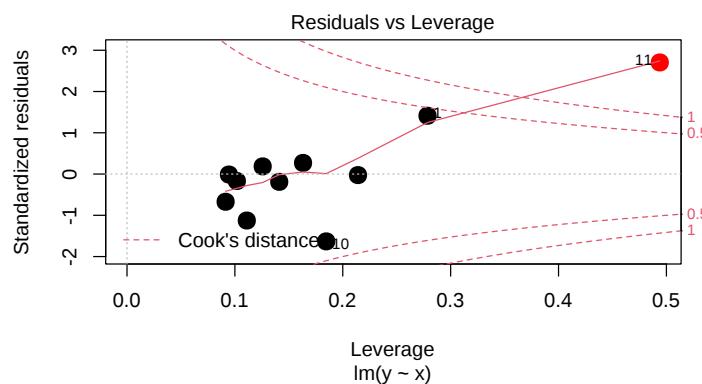


Alto apalancamiento, residuo alto, influencia grande

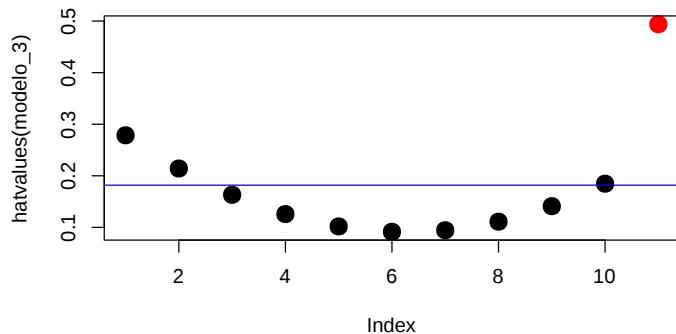
```
p_3 <- c(14, 5.1)
apa_df_3 <- rbind(apa_df, p_3)
modelo_3 <- lm(y ~ x, data = apa_df_3)
coef(modelo_3)
```

```
## (Intercept)           x
##   9.6572209  -0.5892241
```

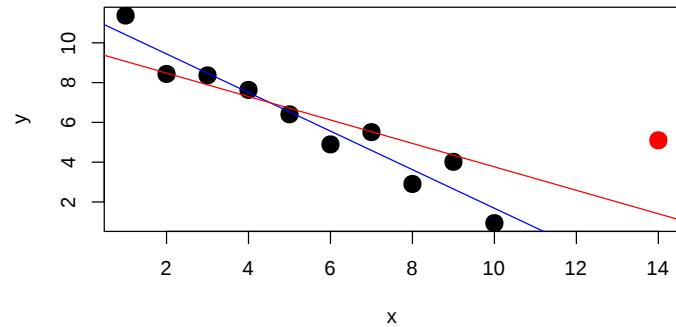
```
plot(modelo_3, 5, col = c(rep("black", 10), "red"),
     cex = 2, pch = 16)
```



```
plot(hatvalues(modelo_3), col = c(rep("black", 10),
                           "red"), cex = 2, pch = 16)
abline(h = 2/11, col = "blue")
```

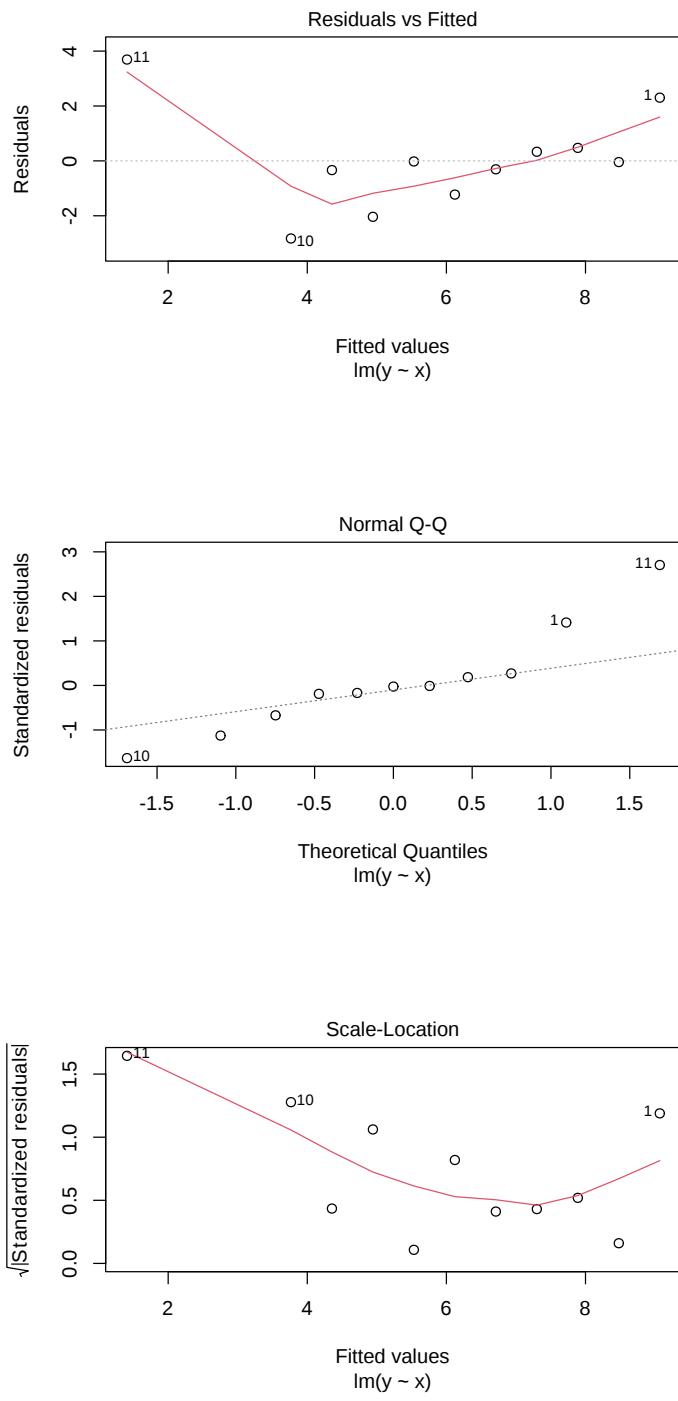


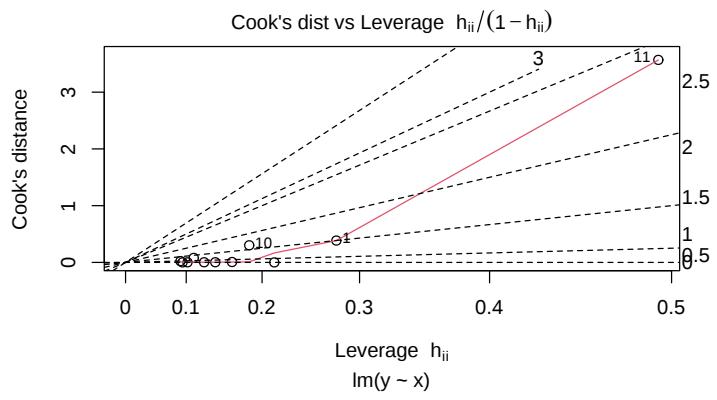
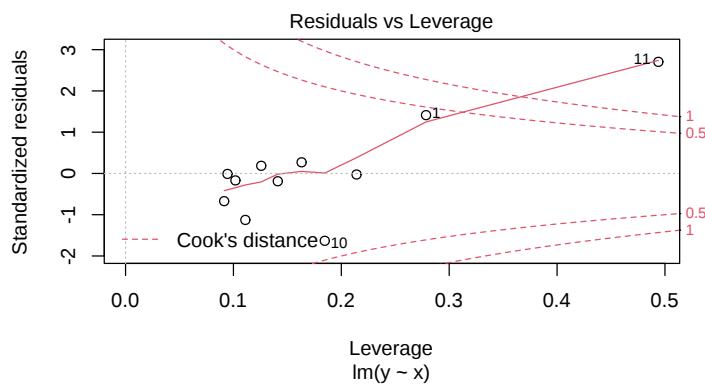
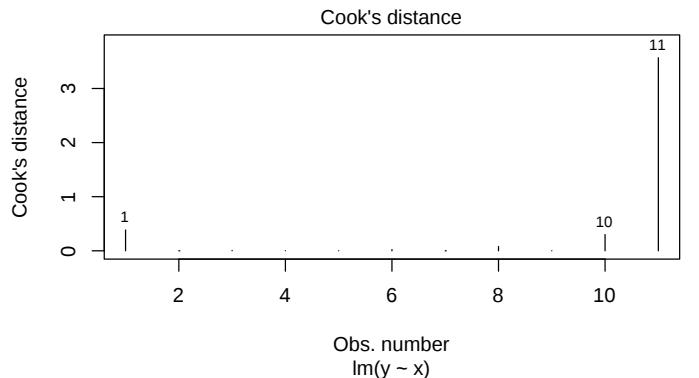
```
plot(apa_df_3, col = c(rep("black", 10), "red"), cex = 2,
      pch = 16)
abline(a = coef(modelo)[1], b = coef(modelo)[2], col = "blue")
abline(a = coef(modelo_3)[1], b = coef(modelo_3)[2],
       col = "red")
```



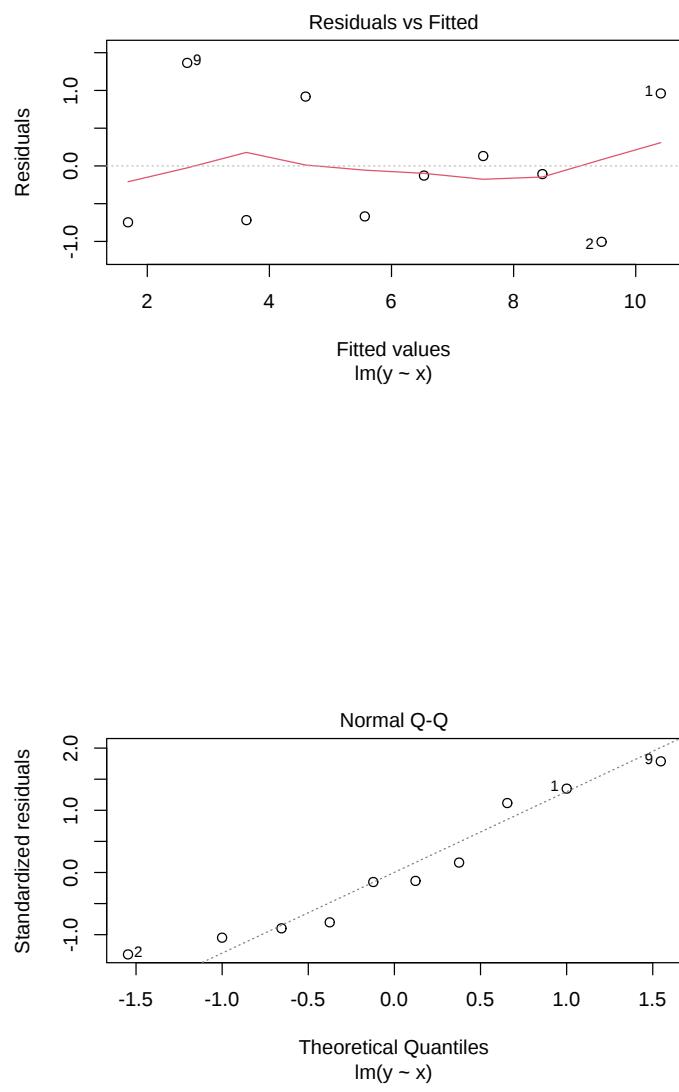
```
? (stats:::plot.lm)
```

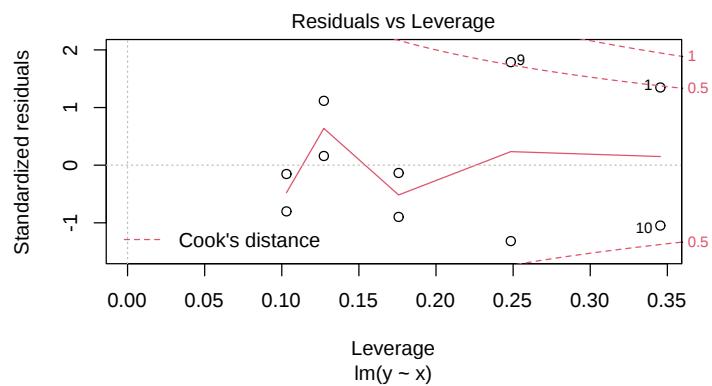
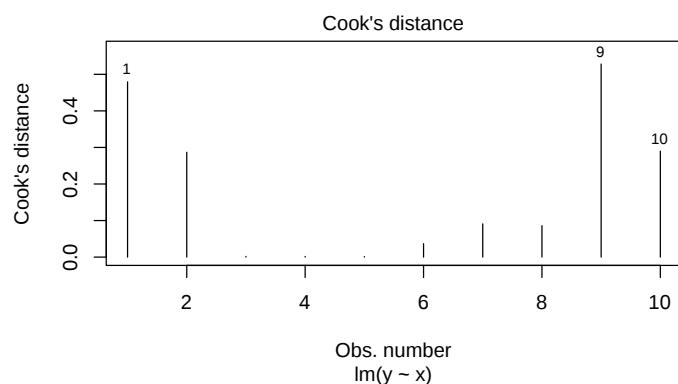
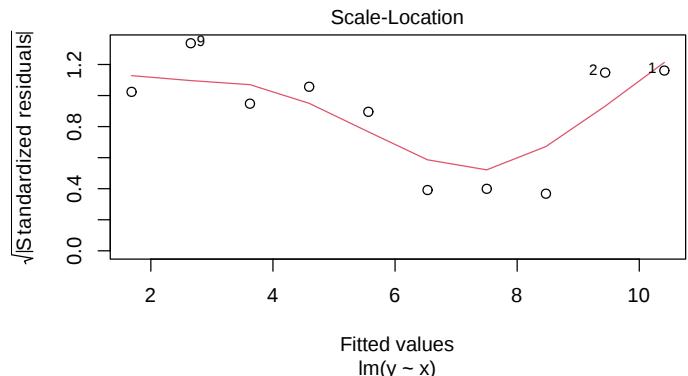
```
plot(modelo_3, which = 1:6)
```

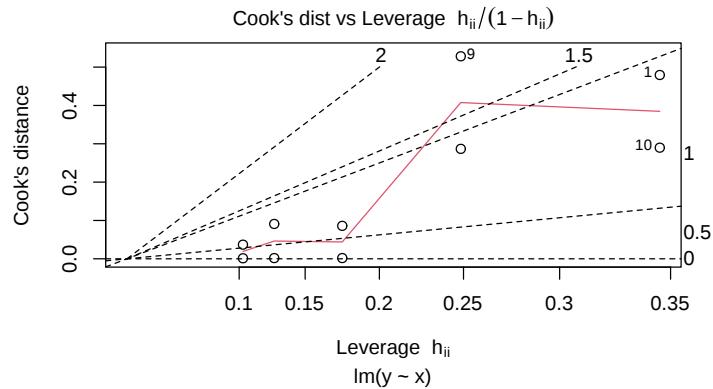




```
plot(modelo, which = 1:6)
```







5.8. Ejercicios

Del libro (James y col. 2013)

- Capítulo 3: 1, 3, 4, 5, 8, 9

Capítulo 6

Regresión Logística

Asuma que ahora la variable Y solo contiene valores 0 o 1 y queremos hacer la regresión

$$Y = \beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p + \varepsilon.$$

El problema es que $\mathbb{E}[Y|\mathbf{X}] = \mathbb{P}(Y = 1|\mathbf{X})$ y se debe cumplir que

$$0 \leq \mathbb{E}[Y|\mathbf{X}] \leq 1.$$

pero el rango de $\beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p$ es todo \mathbb{R} .

Solución es cambiar Y por $g(Y) \in [0, 1]$.

$$g(X) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p)}}$$

```
titanic <- read.csv("data/titanic.csv")
summary(titanic)
```

```
##   PassengerId      Survived      Pclass          Name
##   Min.   : 1.0   Min.   :0.0000   Min.   :1.000   Length:891
##   1st Qu.:223.5  1st Qu.:0.0000  1st Qu.:2.000   Class  :character
##   Median :446.0   Median :0.0000  Median :3.000   Mode   :character
```

```

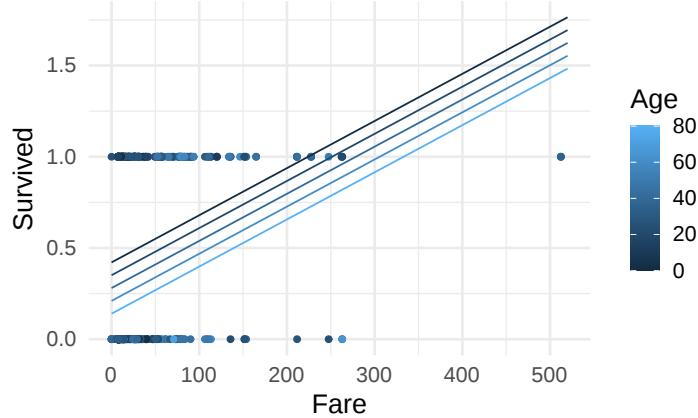
##  Mean    :446.0   Mean    :0.3838   Mean    :2.309
##  3rd Qu.:668.5   3rd Qu.:1.0000   3rd Qu.:3.000
##  Max.   :891.0   Max.   :1.0000   Max.   :3.000
##
##          Sex           Age         SibSp        Parch
##  Length:891       Min.   : 0.42   Min.   :0.000   Min.   :0.0000
##  Class  :character 1st Qu.:20.12   1st Qu.:0.000   1st Qu.:0.0000
##  Mode   :character  Median :28.00   Median :0.000   Median :0.0000
##                      Mean   :29.70   Mean   :0.523   Mean   :0.3816
##                      3rd Qu.:38.00   3rd Qu.:1.000   3rd Qu.:0.0000
##                      Max.   :80.00   Max.   :8.000   Max.   :6.0000
##                      NA's   :177
##          Ticket        Fare        Cabin        Embarked
##  Length:891       Min.   : 0.00   Length:891       Length:891
##  Class  :character 1st Qu.: 7.91   Class  :character  Class  :character
##  Mode   :character  Median : 14.45   Mode   :character  Mode   :character
##                      Mean   : 32.20
##                      3rd Qu.: 31.00
##                      Max.   :512.33
##

```

```
titanic <- titanic %>% select(Survived, Fare, Age) %>%
  drop_na()
```

```
fit_lm <- lm(Survived ~ Fare + Age, data = titanic)
```

```
library(ggiraphExtra)
ggPredict(fit_lm) + theme_minimal(base_size = 16)
```



En lugar de esto, definamos el siguiente modelo

$$Y \sim \text{Bernoulli}(g_{\beta}(\mathbf{X}))$$

con $g_{\beta}(\mathbf{X}) = \mathbb{P}(Y = 1 | \mathbf{X})$.

En R usaremos la función `glm`

```
fit_glm <- glm(Survived ~ Fare + Age, data = titanic,
                 family = "binomial")
summary(fit_glm)
```

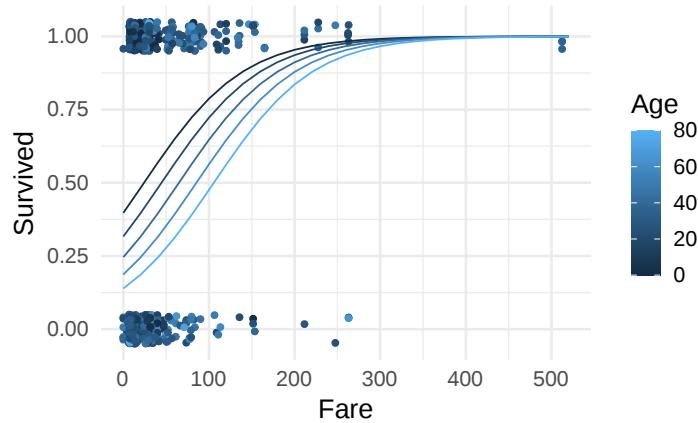
```
##
## Call:
## glm(formula = Survived ~ Fare + Age, family = "binomial", data = titanic)
##
## Deviance Residuals:
##      Min        1Q     Median        3Q       Max
## -2.7605   -0.9232   -0.8214    1.2362    1.7820
##
## Coefficients:
##             Estimate Std. Error z value     Pr(>|z|)
## (Intercept) -0.417055   0.185976  -2.243     0.02493 *
## Fare         0.017258   0.002617   6.596 0.0000000000423 ***
```

```

## Age           -0.017578   0.005666  -3.103          0.00192 ** 
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## (Dispersion parameter for binomial family taken to be 1)
## 
## Null deviance: 964.52 on 713 degrees of freedom
## Residual deviance: 891.34 on 711 degrees of freedom
## AIC: 897.34
## 
## Number of Fisher Scoring iterations: 5

```

```
ggPredict(fit_glm) + theme_minimal(base_size = 16)
```



Nota: Existen otros tipos de regresión y estas se definen a través del parámetro `family`. En este curso solo nos enfocaremos en el parámetro `family="binomial"`.

6.1. Razón de proporción

Defina

$$O(X) = \frac{g(X)}{1 - g(X)} = e^{\beta_0 + \beta_1 X_1 + \dots + \beta_p X_p} \in [0, 1].$$

Es la relación de obtener 1 ó 0.

Por suponga que $\mathbb{P}(Y = 1|\mathbf{X}) = g(\mathbf{X}) = 0,8$ es la probabilidad de pagar la tarjeta de crédito y $1 - g(\mathbf{X}) = 0,2$ es la probabilidad de no pagar.

Se puede escribir $O(X) = \frac{0,8}{0,2} = \frac{4}{1}$, lo que dice que es 4 veces más probable de pagar la tarjeta que no pagarla.

6.2. Máxima verosimilitud

Los valores de β se pueden encontrar por máxima verosimilitud.

Defina $p(\mathbf{X}) = \mathbb{P}(Y = 1|\mathbf{X})$.

La verosimilitud es:

$$L(\beta) = \prod_{i=1}^n p(\mathbf{X}_i)^{Y_i} (1 - p(\mathbf{X}_i))^{1-Y_i}$$

$$\begin{aligned} \ell(\beta) &= \sum_{i=1}^n Y_i \log p(\mathbf{X}_i) + (1 - Y_i) \log (1 - p(\mathbf{X}_i)) \\ &= \sum_{i=1}^n \log (1 - p(\mathbf{X}_i)) + \sum_{i=1}^n Y_i \log \frac{p(\mathbf{X}_i)}{1 - p(\mathbf{X}_i)} \\ &= \sum_{i=1}^n \log (1 - p(\mathbf{X}_i)) + \sum_{i=1}^n Y_i (\mathbf{X}_i \cdot \beta) \\ &= \sum_{i=1}^n -\log (1 + e^{\mathbf{X}_i \cdot \beta}) + \sum_{i=1}^n Y_i (\mathbf{X}_i \cdot \beta) \end{aligned}$$

$$\begin{aligned} \frac{\partial \ell}{\partial \beta} &= -\sum_{i=1}^n \frac{1}{1 + e^{\mathbf{X}_i \cdot \beta}} e^{\mathbf{X}_i \cdot \beta} \mathbf{X}_i + \sum_{i=1}^n Y_i \mathbf{X}_i \\ &= \sum_{i=1}^n (Y_i - p(\mathbf{X}_i)) \mathbf{X}_i \\ &= X^\top (Y - p(\mathbf{X})) \end{aligned}$$

Solución: Netwon-Raphson

Ejercicio 6.1. Muestre que

$$\frac{\partial^2 \ell}{\partial \beta^2} = -\mathbf{X}W\mathbf{X}$$

donde $W = \text{diag}p(\mathbf{X}_i)(1 - p(X_i))$.

El algoritmo de Netwon-Raphson usa el hecho que

$$\beta^{(t)} = \beta^{(t-1)} - \left(\frac{\partial^2 \ell}{\partial \beta^2} \right)^{-1} \frac{\partial \ell}{\partial \beta} \Big|_{\beta^{(t-1)}}$$

Ejercicio 6.2. Muestre que

$$\beta^{(t)} = (\mathbf{X}^\top W \mathbf{X})^{-1} \mathbf{X}^\top Z_\beta,$$

donde $Z_\beta = Z\beta + W_\beta^{-1}(Y - p(X))$.

A esta técnica se le conoce como **mínimos cuadrados ponderados e iterados** o en inglés **Iteratively Re-Weighted Least Squares (IRLS)**.

Se comienza con $\beta^{(0)}$ cualquiera y se va iterando $\beta^{(1)}, \beta^{(2)}, \dots$ hasta encontrar la convergencia.

Para cada t se resuelve el problema

$$\beta^t = \underset{\beta}{\operatorname{argmin}} (Z - X\beta)^\top W (Z - X\beta).$$

6.2.1. Residuos

La suma al cuadrado de los residuos se convierte en un estadístico de pearson:

$$\chi^2 = \sum_{i=1}^n \frac{(Y_i - \hat{p}(X_i))^2}{\hat{p}(X_i)(1 - \hat{p}(X_i))}$$

la cual es una aproximación cuadrática de la devianza (Curso pasado).

$$D = -2\ell(\hat{\beta})$$

Además tenemos los resultados que

- $\hat{\beta} \xrightarrow{\mathbb{P}} \beta$
- $\hat{\beta} \xrightarrow{\mathcal{D}} \mathcal{N}(\beta, (X^\top W X)^{-1})$ (Prueba de Wald)
- Se pueden comparar un modelo completo con un reducido a través de pruebas asintóticas LRT

$$D_c - D_r \sim= \chi^2_{df_c} - \chi^2_{df_r}.$$

6.3. Diagnósticos del modelo

CUIDADO: La función `glm` no tiene un equivalente de `plot` como en los modelos lineales. De esta forma, si se aplica `plot` a un objeto `glm` solo generará los mismos chequeos que el capítulo anterior. Sin embargo estos podrían estar equivocados si no se leen con cuidado.

6.3.1. Supuesto de linealidad

Este supuesto debe ser chequeado con la función `logit` de las respuestas.

```
fit_glm <- glm(Survived ~ Fare + Age, data = titanic,
                 family = "binomial")
summary(fit_glm)

##
## Call:
## glm(formula = Survived ~ Fare + Age, family = "binomial", data = titanic)
##
## Deviance Residuals:
##       Min        1Q    Median        3Q       Max
## -2.7605  -0.9232  -0.8214   1.2362   1.7820
##
```

```

## Coefficients:
##              Estimate Std. Error z value     Pr(>|z|)
## (Intercept) -0.417055   0.185976 -2.243     0.02493 *
## Fare         0.017258   0.002617  6.596 0.0000000000423 ***
## Age          -0.017578   0.005666 -3.103     0.00192 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
## Null deviance: 964.52 on 713 degrees of freedom
## Residual deviance: 891.34 on 711 degrees of freedom
## AIC: 897.34
##
## Number of Fisher Scoring iterations: 5

```

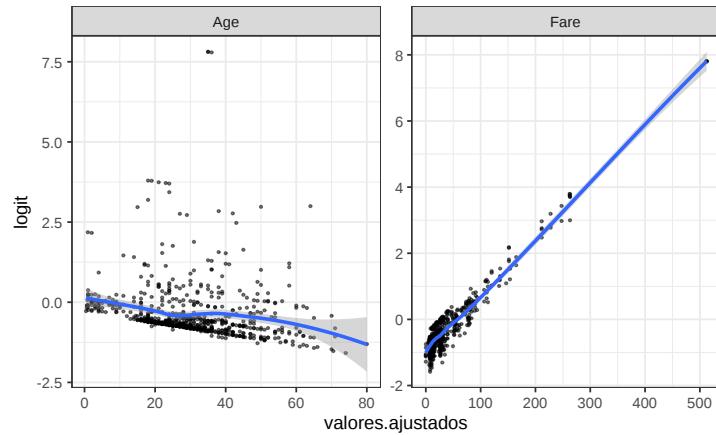
```

probs <- predict(fit_glm, type = "response")

df <- titanic %>% select(Fare, Age) %>% mutate(logit = qlogis(probs)) %>%
  pivot_longer(names_to = "predictores", values_to = "valores.ajustados",
  -logit)

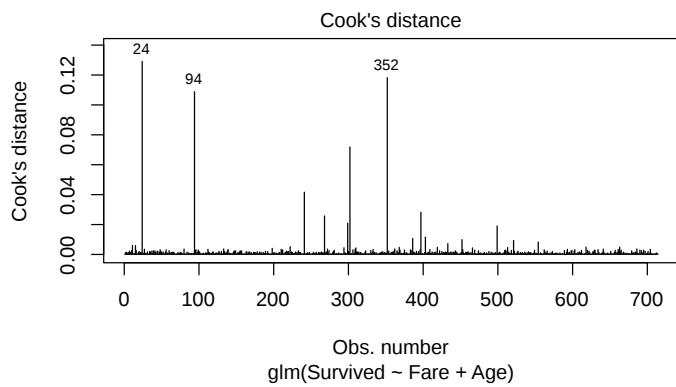
ggplot(df, aes(valores.ajustados, logit)) + geom_point(size = 0.5,
  alpha = 0.5) + geom_smooth(method = "loess") +
  theme_bw() + facet_wrap(~predictores, scales = "free")

```

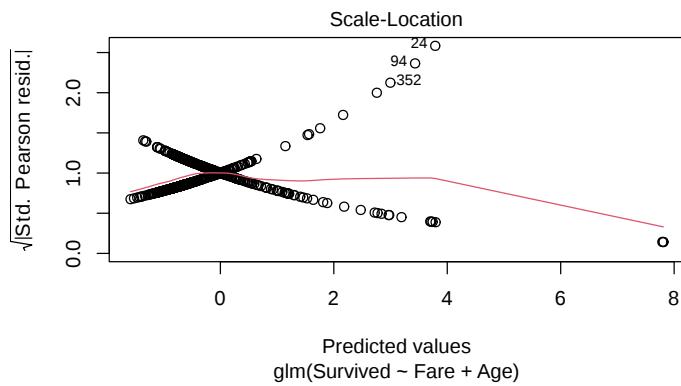
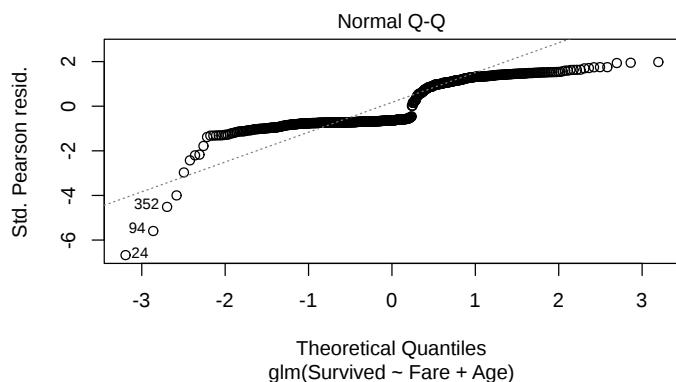
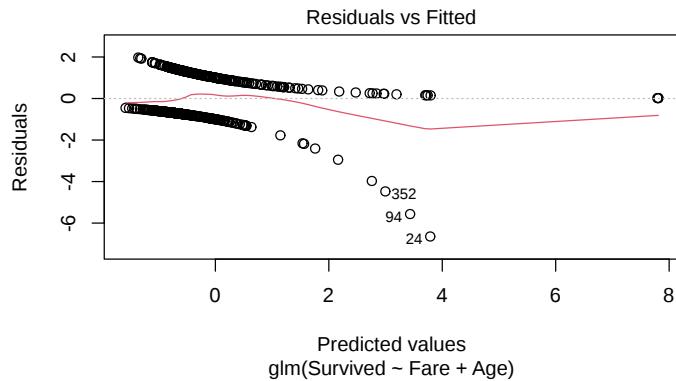


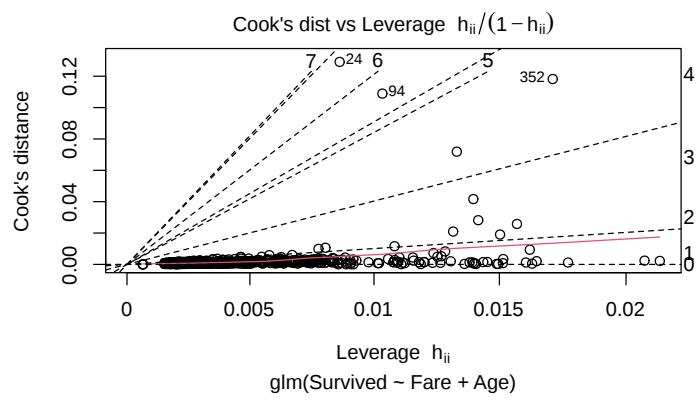
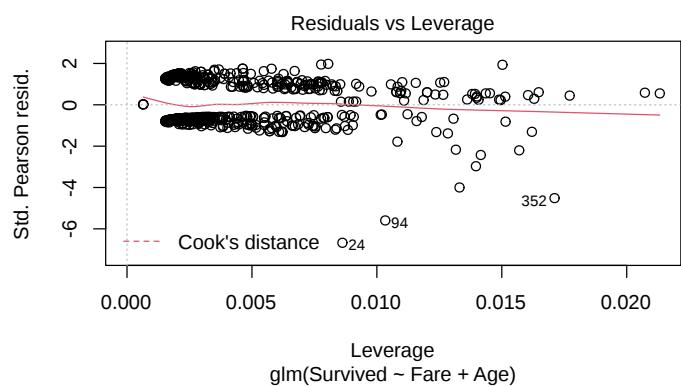
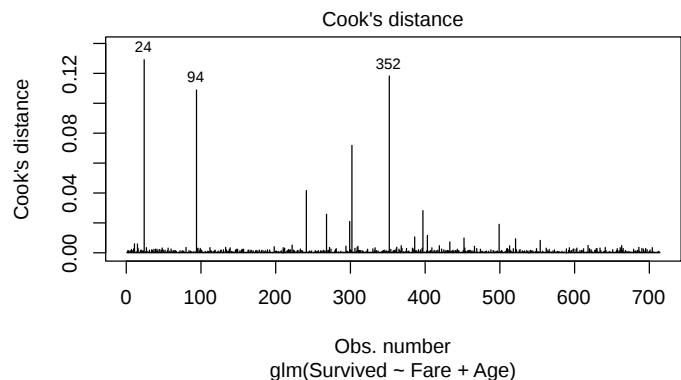
6.3.2. Valor de gran influencia

```
plot(fit_glm, which = 4)
```



```
plot(fit_glm, which = 1:6)
```





```

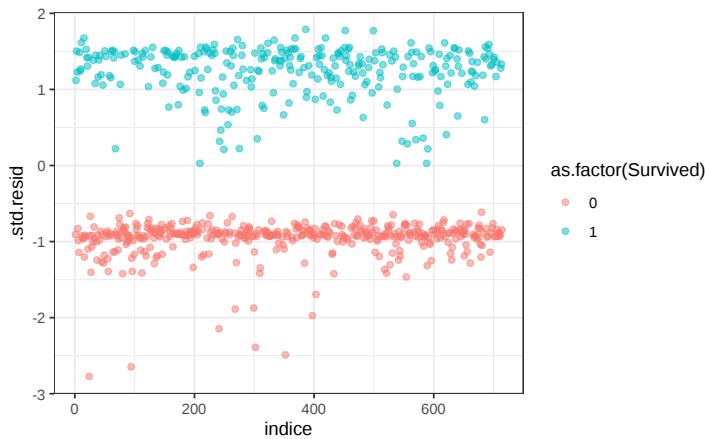
library(broom)
fit_data <- broom::augment(fit_glm) %>% mutate(indice = 1:n())

fit_data %>% top_n(3, .cooksdi)

## # A tibble: 3 x 11
##   Survived   Fare     Age .fitted .se.fit .resid     .hat .sigma .cooksdi .std.resid
##       <int>  <dbl>   <dbl>    <dbl>   <dbl>    <dbl>    <dbl>   <dbl>      <dbl>
## 1       0    263     19     3.79   3.79   0.631   -2.76  0.00862    1.12    0.129
## 2       0    248     24     3.43   3.43   0.584   -2.63  0.0103     1.12    0.109
## 3       0    263     64     3.00   3.00   0.615   -2.47  0.0171     1.12    0.118
## # ... with 1 more variable: indice <int>

ggplot(fit_data, aes(indice, .std.resid)) + geom_point(aes(color = as.factor(Survived), alpha = 0.5)) + theme_bw()

```



```

fit_data %>% filter(abs(.std.resid) > 3)

## # A tibble: 0 x 11
## # ... with 11 variables: Survived <int>, Fare <dbl>, Age <dbl>, .fitted <dbl>,
## #   .se.fit <dbl>, .resid <dbl>, .hat <dbl>, .sigma <dbl>, .cooksdi <dbl>,
## #   .std.resid <dbl>, indice <int>

```

6.3.3. Multicolinealidad

```
car::vif(fit_glm)
```

```
##      Fare      Age
## 1.033878 1.033878
```

6.4. Predicción y poder de clasificación

Si queremos predecir posibles resultados con nuestro modelo logístico, debemos asegurarnos que el error no sea «muy grande».

Ahora el error en este caso, se interpreta diferente que en regresión lineal clásica ya que nuestra salida solamente serán 0's o 1's.

Primero recordemos que el modelo predictivo estaría definido por

$$\hat{p}(X) = \frac{1}{1 + e^{-(\hat{\beta}_0 + \hat{\beta}_1 X_1 + \dots + \hat{\beta}_p X_p)}}$$

donde los β 's son estimados usando IRLS.

Ahora imaginemos que tenemos un conjunto de datos nuevo (X_1^*, \dots, X_p^*) y queremos ver que tipo de respuesta Y^* obtenemos (0 o 1).

Obviamente nuestro modelo puede equivocarse y darnos una respuesta errónea. Por ejemplo digamos que en el caso del `titanic` uno esperaría que personas más jóvenes y que hayan pagado más por su tiquete tengan mayor probabilidad de sobrevivencia.

Entonces tenemos realmente 4 opciones

		Clase	Predicción	
		0	1	
Clase	0	Verdaderos Negativos. (TN)	Falsos Positivos (FP)	N
Real	1	Falsos Negativos (FN)	Verdaderos Positivos (TP)	P
	Total	N^*	P^*	

```

predict_numeric <- predict(fit_glm, type = "response")
predict_01 <- as.numeric(predict_numeric >= 0.5)

matriz_confusion <- table(titanic$Survived, predict_01)

colnames(matriz_confusion) <- c("N", "P")
rownames(matriz_confusion) <- c("N", "P")

matriz_confusion

##      predict_01
##      N      P
##      N 380  44
##      P 201  89

```

Para entender la siguiente tabla vamos a definir los siguientes términos:

Exactitud (Accuracy) Es la tasa de todos los individuos bien etiquetados $(TP + TN)/(TP + TN + FN + FP)$.

Precisión Es la tasa de elementos etiquetados 1 correctamente con respecto a los que fueron etiquetados 1 $Precisión = TP/P^*$

Sensibilidad (Exhaustividad) Es la tasa de elementos etiquetados 1 correctamente con respecto a los que realmente son 1. $Sensibilidad = TP/P$

F-Score Es la media armónica entre la precisión y la sensibilidad. $F-Score = 2 * (Sensibilidad * Precisión)/(Sensibilidad + Precisión)$

Especificidad Es la tasa de elementos etiquetados 0 que realmente estaban etiquetados como 1.

Entonces esto nos da las siguientes posibilidades.

Tipo	Cálculo	Sinónimos
Tasa Falsos Positivos	FP/N	Error Tipo I, 1-Especificidad
Tasa Verdaderos Positivos	TP/P	1-Error Tipo II, Poder, Sensibilidad, Exhaustividad (Recall)

Tipo	Cálculo	Sinónimos
Valor de Predicción Positivos	TP/P^*	Precisión, 1 - Proporción de Falsos Descubrimientos
Valor de Predicción Negativos	TN/N^*	
F-Score	$\frac{2(TP/P^* \times TP/P)}{(TP/P^* + TP/P)}$	

CUIDADO:

- Exactitud funciona bien cuando los datos son simétricos (igual número de FP y FN).
- F-Scores es cuando los datos son asimétricos
- Precisión es qué tan seguro se está de los verdaderos positivos.
- Sensibilidad es que tan seguro es que no se está perdiendo ningún positivo.

En un modelo se debe escoger entre sensibilidad y precisión de acuerdo a ciertas ideas:

- **Sensibilidad** es importante si la ocurrencia de **falsos negativos** es inaceptable. Por ejemplo las prueba COVID-19. Posiblemente se obtendrá más falsos positivos pero este caso es aceptable.
- **Precisión** es importante si se quiere estar más seguro de los **verdaderos positivos**. Por ejemplo detectar **spam** en correos electrónicos.
- **Especificidad** es importante si lo que se quiere es cubrir todos los **verdaderos negativos**, es decir, que no se quieren falsas alarmas. Por ejemplo se hacen pruebas de detección de drogas y si es positivo va a la cárcel. Los **falsos positivos** son intolerables.

```
(TN <- matriz_confusion["N", "N"])
```

```
## [1] 380
```

```
(TP <- matriz_confusion["P", "P"])

## [1] 89

(FP <- matriz_confusion["N", "P"])

## [1] 44

(FN <- matriz_confusion["P", "N"])

## [1] 201

(exactitud <- (TP + TN)/(TP + TN + FP + FN))

## [1] 0.6568627

(precision <- TP/(TP + FP))

## [1] 0.6691729

(sensibilidad <- TP/(TP + FN))

## [1] 0.3068966

(F_score <- 2 * (precision * sensibilidad)/(precision +
sensibilidad))

## [1] 0.4208038

(especificidad <- TN/(TN + FP))

## [1] 0.8962264
```

6.4.1. Curva ROC

Un excelente clasificador debería detectar correctamente los **verdaderos positivos (TP)** e ignorar los **falsos positivos (FP)**.

Puesto de otra forma, si el clasificador es malo, los **verdaderos positivos** serían indistinguibles de los **falsos positivos**.

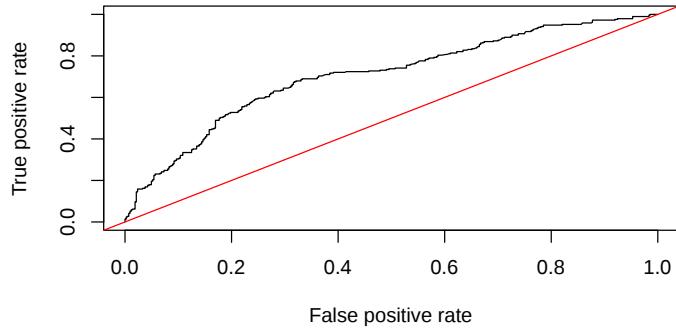
La curva ROC (Receiver Operation Curve) gráfica la Tasa Falsos Positivos vs Sensibilidad del modelo.

```
library(ROCR)

logist.pred.ROCR <- prediction(predict_numeric, titanic$Survived)

logist.perf <- performance(logist.pred.ROCR, "tpr",
  "fpr")

plot(logist.perf)
abline(0, 1, col = "red")
```



```
auc <- performance(logist.pred.ROCR, measure = "auc")

auc@y.values
```

```
## [[1]]
## [1] 0.7063313
```

PELIGRO

Aquí estamos chequeando el poder de clasificación del modelo, con los mismos datos que usamos para ajustar el modelo. Es decir, le estamos diciendo al modelo que compruebe la veracidad de la clasificación que ya hizo previamente.

Esto es incorrecto, ya que el modelo ya sabe «las respuestas» y no estamos midiendo su poder de clasificación.

Para resolver esto, debemos tomar otra muestra de prueba (**training**) que nos diga si el ajuste que hicimos es correcto.

```
titanic$id <- 1:nrow(titanic)

train <- titanic %>% sample_frac(0.75)

test <- titanic %>% anti_join(train, by = "id")

fit_glm <- glm(Survived ~ Fare + Age, data = train,
                 family = "binomial")

predict_numeric <- predict(fit_glm, newdata = test,
                           type = "response")
predict_01 <- as.numeric(predict_numeric >= 0.5)

matriz_confusion <- table(test$Survived, predict_01)

colnames(matriz_confusion) <- c("N", "P")
rownames(matriz_confusion) <- c("N", "P")

matriz_confusion

##      predict_01
##      N   P
##      N 93  9
##      P 57 19
```

```
(TN <- matriz_confusion["N", "N"])

## [1] 93

(TP <- matriz_confusion["P", "P"])

## [1] 19

(FP <- matriz_confusion["N", "P"])

## [1] 9

(FN <- matriz_confusion["P", "N"])

## [1] 57

(exactitud <- (TP + TN)/(TP + TN + FP + FN))

## [1] 0.6292135

(precision <- TP/(TP + FP))

## [1] 0.6785714

(sensibilidad <- TP/(TP + FN))

## [1] 0.25

(F_score <- 2 * (precision * sensibilidad)/(precision +
sensibilidad))

## [1] 0.3653846
```

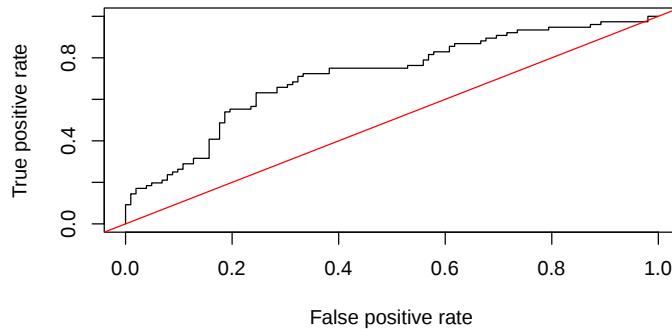
```
(especificidad <- TN/(TN + FP))

## [1] 0.9117647

logist.pred.ROCR <- prediction(predict_numeric, test$Survived)

logist.perf <- performance(logist.pred.ROCR, "tpr",
  "fpr")

plot(logist.perf)
abline(0, 1, col = "red")
```



```
auc <- performance(logist.pred.ROCR, measure = "auc")

auc@y.values

## [[1]]
## [1] 0.7138803
```

6.5. Ejercicios

- Del libro (James y col. 2013):

- Capítulo 4: 1, 6, 10, 11. (En esta sección no vimos LDA, QDA ni k-vecinos más cercanos, así que ignoren esas partes y concentrense en hacer los análisis correctos para regresión logística).

Capítulo 7

Métodos de selección de variables

Cuando se construye un modelo de regresión (lineal o logística) existe la posibilidad de que existan más variables que datos disponibles.

Si se recuerdan del Capítulo 4, el R^2 ajustado tenía un factor $n - p - 1$ en el denominador y si $n > p$ este tipo de indicador no se podría estimar.

En este capítulo veremos cómo construir modelos más pequeños y hacer comparaciones entre ellos.

7.1. Selección del mejor subconjunto.

En este caso trataremos de seleccionar el mejor subconjunto de un total de p variables. Claramente si escogieramos solo k variables existiría $\binom{p}{k}$ modelos diferentes que escoger.

El algoritmo para este caso sería:

Algoritmo: 1. Sea M_0 el modelo nulo (solo tiene constantes). 1. Para $k = 1, 2, \dots, p$ (número de variables), a. Ajuste todos los $\binom{p}{k}$ modelos que contengan k predictores. b. Seleccione el mejor entre esos $\binom{p}{k}$ modelos. El «mejor» es el que tenga el RSS menor, o el R^2 más grande. Llame a este modelo

M_k . 1. Seleccione el mejor modelo entre M_0, M_1, \dots, M_p seleccione aquel que tenga mejor error de validación cruzada, C_p , AIC , BIC o R^2 ajustado.

Nota: Más adelante veremos qué es validación cruzada, C_p , AIC y BIC

Ejemplo: $Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_3$.

Puede ser que el mejor modelo sea

- $Y = \beta_0$,
- $Y = \beta_0 + \beta_1 X_1$,
- $Y = \beta_0 + \beta_2 X_2$,
- $Y = \beta_0 + \beta_3 X_3$,
- $Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2$,
- $Y = \beta_0 + \beta_1 X_1 + \beta_3 X_3$, entre otras.

De los que tienen $k = 1$ variable, hay $\binom{3}{1} = 3$ modelos. Para $k = 2$, son $\binom{3}{2} = 3$, y para $k = 3$, solo un modelo. Para $k = 1$, se ajustan los 3 y al mejor se le llama M_1 . Así para los otros k . Obtenidos estos modelos, se escoge el que tenga la mejor medida, con respecto a los errores antes mencionados.

Notas:

- La parte 2.b. se hace con la muestra de entrenamiento. **Objetivo: Minimizar el error de entrenamiento.**
- La parte 3 se selecciona con los datos de prueba. **Objetivo: Minimizar el error de prueba.**
- Si se usa el RSS o R^2 , siempre se selecciona el modelo con el número mayor de variables. Este es un problema de sobreajuste.
- El gran problema es la cantidad de variables y los modelos por ajustar, los cuales son $\sum_{k=1}^n \binom{p}{k}$. Computacionalmente ineficiente.

7.1.1. Error de prueba

El problema con el error de prueba es que contiene una fuente de variabilidad no observada por el modelo.

2 soluciones

- Estimar indirectamente el error de prueba al añadirle un factor de sobreajuste. Es decir, agregarle artificialmente la variabilidad no observada, pero incluyendo más sesgo.
- Estimar directamente el error de prueba usando validación cruzada.

7.1.1.1. Validación cruzada

Dividir aleatoriamente los datos totales en 2 partes (con tamaños comparables)

- Conjunto de prueba: Se ajusta el modelo.
- Conjunto de validación: El modelo ajustado se usa para predecir en este conjunto.

Se usa una medida de validación en cada escogencia de los dos conjuntos (MSE en el caso cuantitativo)

1. Puede que el varianza del estimador tenga una varianza muy alta.
2. El error de validación tiende a ser mayor que el error de entrenamiento.
Puede haber sobreestimación.

Validación cruzada «Leave-One-Out» (LOOCV) Una sola observación (X_i, Y_i) se usa en el conjunto de validación. Observaciones restantes, se usan en el conjunto de entrenamiento.

Defina

$$MSE_i = (Y_i - \hat{Y}_i)^2$$

como el error cometido por usar la observación i como muestra de prueba y el resto de valores como muestra de entrenamiento.

El estimador LOOCV es

$$CV_n = \frac{1}{n} \sum_{i=1}^n MSE_i$$

Ventajas

1. Menos sesgo. (Conjunto de prueba de tamaño casi igual que los datos totales).
2. Menos varianza.

Problema: Puede ser lento, dependiendo de la cantidad de datos.

Validación cruzada k -veces Se aplica el mismo principio que LOOCV, pero se divide la muestra en k distintas partes.

$$CV_k = \frac{1}{k} \sum_{i=1}^k MSE_k$$

Ventajas: Es más económico

Desventaja: Nivel intermedio de sesgo

$$\frac{n}{2} < \frac{(k-1)n}{k} < n - 1$$

7.1.1.2. Validación cruzada para clasificación

Se usa

$$CV_n = \frac{1}{n} \sum_{i=1}^n Err_i$$

donde $Err_i = I_{Y_i \neq \hat{Y}_i}$

y CV_k se define similar

7.1.2. Otras medidas de error de entrenamiento

- R^2 ajustado. Recuerde que $R^2 = 1 - \frac{RSS}{TSS}$. Como RSS decrece si se le agrega más variables, entonces $R^2 \nearrow 1$. Vea que $RSS = \sum(y_i - \hat{y}_i)^2$ y $TSS = \sum(y_i - \bar{y}_i)^2$, entonces,

$$R^2 \text{ ajustado} = 1 - \frac{\frac{RSS}{n-k-1}}{\frac{TSS}{n-1}}$$

- C_p . Se usa en ajustes con mínimos cuadrados.

$$C_p = \frac{1}{n} [RSS + 2p\hat{\sigma}^2]$$

donde k es el número de predictores y $\hat{\sigma}^2$ es el estimador de la varianza de los errores ϵ . Si $\hat{\sigma}^2$ es insegado de σ^2 , entonces C_p es un estimador insegado del MSE de prueba.

- C_p de Mallows. Se denota C'_p .

$$C'_p = \frac{RSS}{\hat{\sigma}^2} + 2p - n \implies C_p = \frac{1}{n} \hat{\sigma}^2 [C'_p + n] \propto C'_p$$

- MLE : $2 \ln L(\hat{\beta}|x)$.
- AIC (Akaike Information Criterion).

$$AIC = -2 \log(L(\hat{\beta}|X)) + 2p$$

Esta medida es derivada asintóticamente de $-2n\mathbb{E}[\log(L(\hat{\beta}|X))]$ cuando $n \rightarrow \infty$.

- BIC (Bayesian Information Criterion).

$$AIC = -2 \log(L(\hat{\beta}|X)) + \log(n)p.$$

Representa asintóticamente hablando, el negativo del logaritmo de la distribución posterior.

Validación cruzada LOOCV es asintóticamente equivalente al AIC para modelos de regresión lineal múltiple (Stone 1977)

Una explicación detallada de cada medida la pueden encontrar en el Capítulo 7 (Hastie, Tibshirani y Friedman 2009) o en el artículo (Cavanaugh y Neath 2019).

7.1.3. Selección de modelos hacia adelante (Forward Stepwise Selection)

1. Sea M_0 el modelo nulo.
2. Para $k = 0, 1, \dots, p - 1$,
 - a. Considere los $p - k$ modelos que contenga los predictores en M_k con un predictor adicional.
 - b. Seleccione el mejor entre esos $p - k$ modelos usando el R^2 o RSS . Llámelo M_{k+1} .
3. Seleccione el mejor modelo entre M_0, \dots, M_p usando VC , Cp , AIC , BIC o R^2 ajustado.

Ejemplo: $Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_3$

- $M_0: Y = \beta_0$
- $M_1: Y = \beta_0 + \beta_1 X_1$, $Y = \beta_0 + \beta_2 X_2$ o $Y = \beta_0 + \beta_3 X_3$. De los tres se escoge el mejor (por ejemplo, la segundo) y se le llama M_1 .
- M_2 : a $Y = \beta_0 + \beta_2 X_2$, que es M_1 , se le suma una variable extra ($\beta_1 X_1$ o $\beta_3 X_3$) y se selecciona la mejor.
- M_3 : M_2 más la variable no incluida.

Nota: el número de modelos por calcular usando el mejor subconjunto es n^p , mientras que usando Forward es $1 + \sum_0^{p-1} p - k = \frac{1 + p(1 + p)}{2}$.

7.1.4. Selección de modelos hacia atrás (Backward Stepwise Selection)

1. Sea M_p el modelo completo.
2. Para $k = p, p - 1, \dots, 1$,
 - a. Considera los k modelos que contienen todos excepto uno de los predictores en M_k para un total de $k - 1$ predictores.
 - b. Selecciona el mejor entre esos k modelos usando el R^2 o RSS . Llámalo M_{k+1} .
3. Selecciona el mejor modelo entre M_0, \dots, M_p usando VC , C_p , AIC , BIC o R^2 ajustado.

Ejemplo: $Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_3$

- M_3 : $Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_3$.
- M_2 : se quita una variable (X_1 , X_2 o X_3) y se selecciona el mejor. Por ejemplo, se remueve X_1 .
- M_1 : A M_2 le quito otra variable. En este caso, X_2 o X_3 y se escoge el mejor.
- M_0 : $Y = \beta_0$, el modelo nulo.

7.2. Métodos de regularización

7.2.1. Regresión Ridge

Considere

$$RSS = \sum_{i=1}^n \left(y_i - \beta \sigma \sum_{j=1}^p \beta_j X_{ij} \right)^2$$

y

$$\hat{\beta} = \underset{\beta}{\operatorname{argmin}} RSS$$

La regresión Ridge consiste en determinar

$$\hat{\beta}_\lambda^R = \underset{\beta}{\operatorname{argmin}} \left[RSS + \lambda \sum_{j=1}^n \beta_j^2 \right]$$

Se define:

$$\|\beta_{-0}\|_2^2 = \sum_{j=1}^n \beta_j^2$$

- Si $\lambda = 0$, $\hat{\beta} = \beta_\lambda^R$: caso de máxima varianza, con el menor sesgo posible.
- Si $\lambda \rightarrow +\infty$, $\beta \rightarrow 0$: se sacrifican todos los parámetros β . Máximo sesgo pero varianza nula.

¿Cómo se debe seleccionar el λ ? El método para seleccionarlo es por validación cruzada

7.2.1.1. Estimación clásica por mínimos cuadrados.

$$\hat{\beta} = (X^T \cdot X)^{-1} X^T y$$

Si se multiplica una constante c a X_i , entonces $\hat{\beta} = \frac{\hat{\beta}_i}{c}$. La constante c afecta directamente al $\|\beta_{-0}\|_2^2$, por lo que se recomienda estandarizar las covariables.

7.2.1.2. Ventajas

1. Indica el balance entre sesgo y varianza.
2. Si $p > n$ (mayor cantidad de variables que datos), al realizar mínimos cuadrados, no se puede dar una solución, pero con la forma de regresión de Ridge es posible alcanzarla.
3. Computacionalmente es más eficiente que usando selección de «todos los subconjuntos».

7.2.2. Regresión Lasso

$$\beta_{\lambda}^{LASSO} = \operatorname{argmin}_{\beta} \left(RSS + \lambda \sum_{j=1}^n |\beta_j| \right)$$

Se define

$$\|\beta_{-0}\|_1 = \sum_{j=1}^n |\beta_j|$$

Otra formulación para los métodos vistos son:

1. **Ridge:** $\min_{\beta} RSS$, sujeto a $\sum_{j=1}^p \beta_j^2 \leq s$.
2. **Lasso:** $\min_{\beta} RSS$, sujeto a $\sum_{j=1}^p |\beta_j| \leq s$.

7.2.3. Explicación gráfica

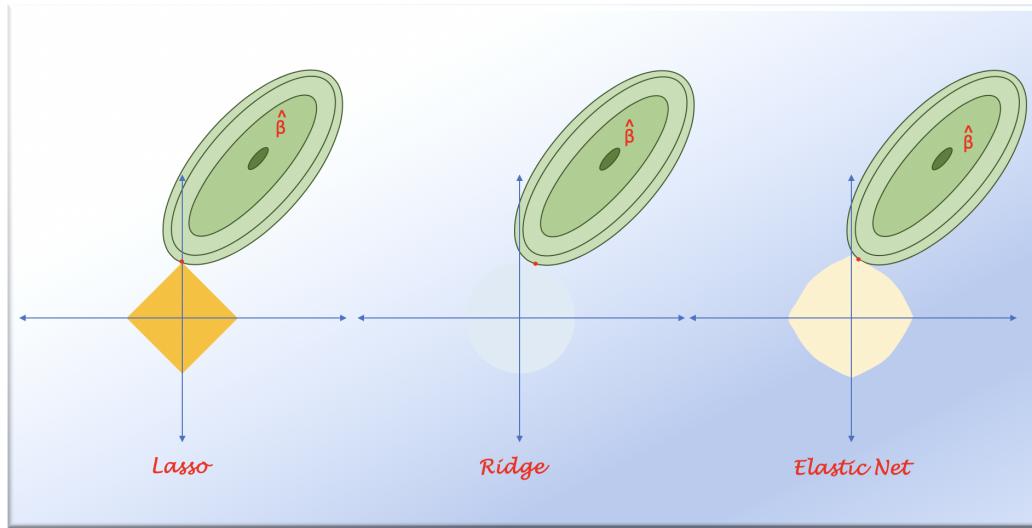


Figura 7.1: Tomado de DataSklr

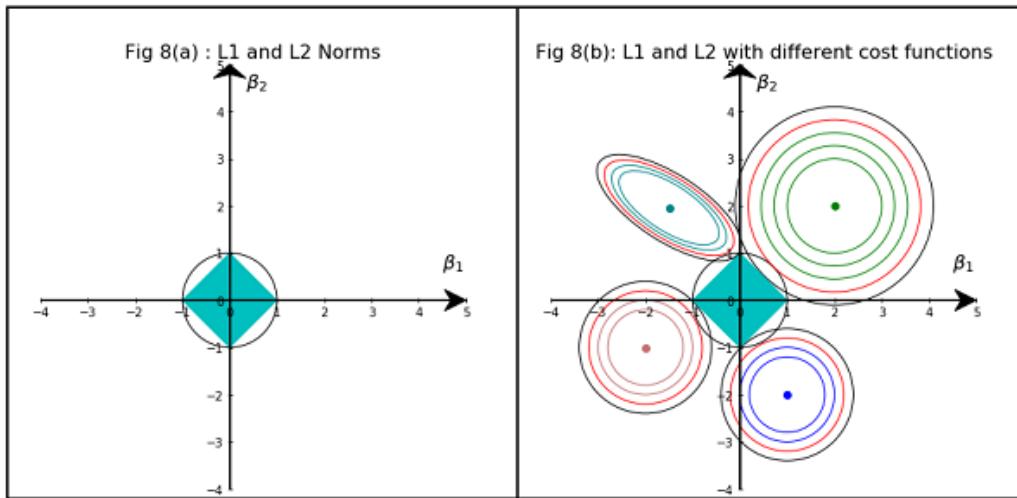


Figura 7.2: Tomado de [Towards to Data Science](#)

7.3. Laboratorio

7.3.1. Cross-Validation

7.3.1.1. Leave-one-out Cross Validation (LOOCV)

Es posible comparar distintos ajustes de modelos usando cross-validation.

Carguemos la base de datos Auto de ISLR.

```
library(ISLR)
data("Auto")
```

Y ajustamos un modelo entre las millas por galon contra los caballos de fuerza de ciertos vehículos.

```
glm.fit <- glm(mpg ~ horsepower, data = Auto)
summary(glm.fit)
```

```
##
## Call:
```

```

## glm(formula = mpg ~ horsepower, data = Auto)
##
## Deviance Residuals:
##       Min        1Q     Median        3Q       Max
## -13.5710   -3.2592   -0.3435    2.7630   16.9240
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept) 39.935861   0.717499  55.66 <2e-16 ***
## horsepower -0.157845   0.006446 -24.49 <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 24.06645)
##
## Null deviance: 23819.0 on 391 degrees of freedom
## Residual deviance: 9385.9 on 390 degrees of freedom
## AIC: 2363.3
##
## Number of Fisher Scoring iterations: 2

```

La librería `boot` tiene funciones para aplicar cross-validation. Por ejemplo,

```

library(boot)
glm.fit <- glm(mpg ~ horsepower, data = Auto)
cv.err <- cv.glm(Auto, glm.fit)
cv.err$delta

```

```
## [1] 24.23151 24.23114
```

En particular se puede usar un `for` para aplicar este mismo procedimiento a múltiples modelos.

```

cv.error.LOOCV = rep(0, 5)
for (i in 1:5) {
  glm.fit = glm(mpg ~ poly(horsepower, i), data = Auto)
  cv.error.LOOCV[i] = cv.glm(Auto, glm.fit)$delta[1]

```

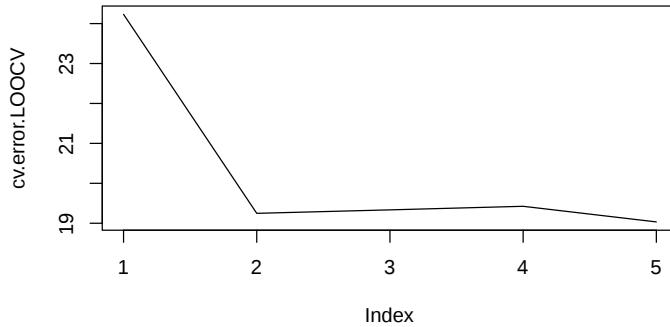
```

}
cv.error.LOOCV

## [1] 24.23151 19.24821 19.33498 19.42443 19.03321

plot(cv.error.LOOCV, type = "l")

```



7.3.1.2. K-Fold Cross Validation

Este procedimiento se puede repetir con los el K-fold.

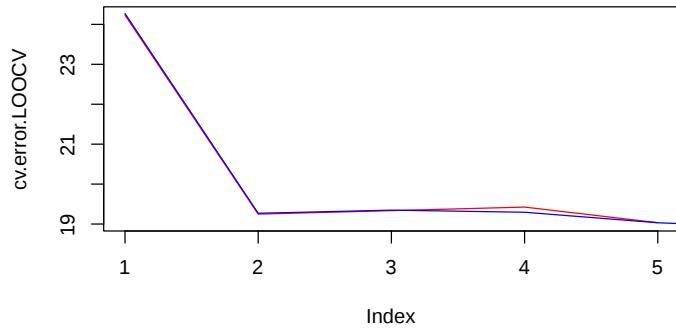
```

set.seed(17)
cv.error.10 = rep(0, 10)
for (i in 1:10) {
  glm.fit = glm(mpg ~ poly(horsepower, i), data = Auto)
  cv.error.10[i] = cv.glm(Auto, glm.fit, K = 10)$delta[1]
}
cv.error.10

## [1] 24.27207 19.26909 19.34805 19.29496 19.03198 18.89781 19.12061 19.1466
## [9] 18.87013 20.95520

```

```
plot(cv.error.LOOCV, type = "l", col = "red")
lines(cv.error.10, type = "l", col = "blue")
```



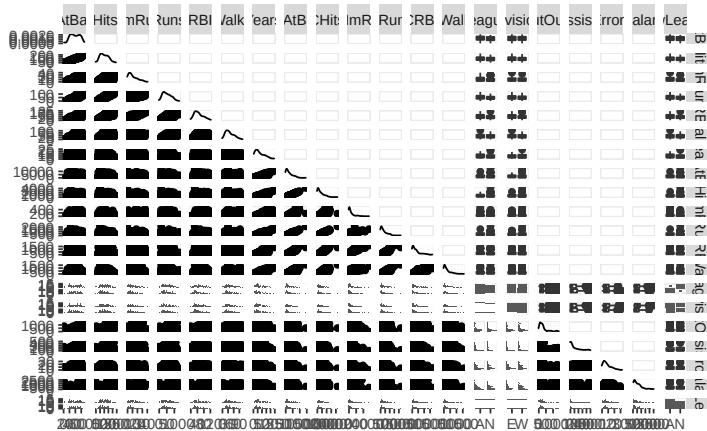
7.3.2. Selección de variables

Cargue los datos `Hitters` del paquete `ISLR` que representan el salario de varios jugadores de béisbol y sus estadística de juego (número de bateos, home runs, carreras, etc.).

7.3.2.1. Análisis exploratorio

Con esta información, haga un análisis exploratorio de los datos usando `ggpairs`.

```
library(GGally)
ggpairs(Hitters)
```



```
summary(Hitters)
```

	AtBat	Hits	HmRun	Runs
##	Min. : 16.0	Min. : 1	Min. : 0.00	Min. : 0.00
##	1st Qu.: 255.2	1st Qu.: 64	1st Qu.: 4.00	1st Qu.: 30.25
##	Median : 379.5	Median : 96	Median : 8.00	Median : 48.00
##	Mean : 380.9	Mean : 101	Mean : 10.77	Mean : 50.91
##	3rd Qu.: 512.0	3rd Qu.: 137	3rd Qu.: 16.00	3rd Qu.: 69.00
##	Max. : 687.0	Max. : 238	Max. : 40.00	Max. : 130.00
##				
	RBI	Walks	Years	CAtBat
##	Min. : 0.00	Min. : 0.00	Min. : 1.000	Min. : 19.0
##	1st Qu.: 28.00	1st Qu.: 22.00	1st Qu.: 4.000	1st Qu.: 816.8
##	Median : 44.00	Median : 35.00	Median : 6.000	Median : 1928.0
##	Mean : 48.03	Mean : 38.74	Mean : 7.444	Mean : 2648.7
##	3rd Qu.: 64.75	3rd Qu.: 53.00	3rd Qu.: 11.000	3rd Qu.: 3924.2
##	Max. : 121.00	Max. : 105.00	Max. : 24.000	Max. : 14053.0
##				
	CHits	CHmRun	CRuns	CRBI
##	Min. : 4.0	Min. : 0.00	Min. : 1.0	Min. : 0.00
##	1st Qu.: 209.0	1st Qu.: 14.00	1st Qu.: 100.2	1st Qu.: 88.75
##	Median : 508.0	Median : 37.50	Median : 247.0	Median : 220.50
##	Mean : 717.6	Mean : 69.49	Mean : 358.8	Mean : 330.12
##	3rd Qu.: 1059.2	3rd Qu.: 90.00	3rd Qu.: 526.2	3rd Qu.: 426.25

```

##   Max.    :4256.0    Max.    :548.00    Max.    :2165.0    Max.    :1659.00
##
##      CWalks          League  Division     PutOuts          Assists
##  Min.    : 0.00    A:175    E:157    Min.    : 0.0    Min.    : 0.0
##  1st Qu.: 67.25   N:147    W:165    1st Qu.:109.2  1st Qu.: 7.0
##  Median :170.50                    Median :212.0  Median :39.5
##  Mean   :260.24                    Mean   :288.9  Mean   :106.9
##  3rd Qu.:339.25                   3rd Qu.:325.0  3rd Qu.:166.0
##  Max.   :1566.00                   Max.   :1378.0  Max.   :492.0
##
##      Errors          Salary        NewLeague
##  Min.    : 0.00    Min.    : 67.5    A:176
##  1st Qu.: 3.00    1st Qu.:190.0    N:146
##  Median : 6.00    Median :425.0
##  Mean   : 8.04    Mean   :535.9
##  3rd Qu.:11.00    3rd Qu.:750.0
##  Max.   :32.00    Max.   :2460.0
##           NA's    :59

```

Para limpiar la base de datos de NA usamos dplyr.

```

library(tidyverse)
Hitters <- Hitters %>% drop_na()

```

7.3.2.2. Selección del mejor subconjunto

Cargue la librería

```
library(leaps)
```

y busque la ayuda de la función `regsubsets`. Use esta función para ajustar todo los posibles modelos de la forma `Salary ~ ..`

Puede guardar estos modelos en ciertas variables (e.g. `regfit.full`) y usar la función `plot`.

```

library(leaps)
regfit.full <- regsubsets(Salary ~ ., Hitters)
summary(regfit.full)

## Subset selection object
## Call: regsubsets.formula(Salary ~ ., Hitters)
## 19 Variables (and intercept)
##          Forced in Forced out
## AtBat      FALSE      FALSE
## Hits       FALSE      FALSE
## HmRun      FALSE      FALSE
## Runs       FALSE      FALSE
## RBI        FALSE      FALSE
## Walks      FALSE      FALSE
## Years      FALSE      FALSE
## CAtBat     FALSE      FALSE
## CHits      FALSE      FALSE
## CHmRun     FALSE      FALSE
## CRuns      FALSE      FALSE
## CRBI       FALSE      FALSE
## CWalks     FALSE      FALSE
## LeagueN    FALSE      FALSE
## DivisionW FALSE      FALSE
## PutOuts    FALSE      FALSE
## Assists    FALSE      FALSE
## Errors     FALSE      FALSE
## NewLeagueN FALSE      FALSE
## 1 subsets of each size up to 8
## Selection Algorithm: exhaustive
##          AtBat Hits HmRun Runs RBI Walks Years CAtBat CHits CHmRun CRuns CRBI
## 1   ( 1 ) " " " " " " " " " " " " " " " " " " " "
## 2   ( 1 ) " " "*" " " " " " " " " " " " " " " " " "
## 3   ( 1 ) " " "*" " " " " " " " " " " " " " " " " "
## 4   ( 1 ) " " "*" " " " " " " " " " " " " " " " " "
## 5   ( 1 ) "*" "*" " " " " " " " " " " " " " " " " "
## 6   ( 1 ) "*" "*" " " " " " " " " " " " " " " " " "
## 7   ( 1 ) " " "*" " " " " " " " " " " " " " " " " "

```

```

## 8  ( 1 ) "*"   "*"   " "   " "   " "   " "   " "   " "   " "   " "   " "   " "   " "   " "
##          CWalks LeagueN DivisionW PutOuts Assists Errors NewLeagueN
## 1  ( 1 ) " "   " "   " "   " "   " "   " "   " "   " "   " "   " "
## 2  ( 1 ) " "   " "   " "   " "   " "   " "   " "   " "   " "   " "
## 3  ( 1 ) " "   " "   " "   "*"   " "   " "   " "   " "   " "
## 4  ( 1 ) " "   " "   " "   "*"   " "   " "   " "   " "   " "
## 5  ( 1 ) " "   " "   " "   "*"   " "   " "   " "   " "   " "
## 6  ( 1 ) " "   " "   " "   "*"   " "   " "   " "   " "   " "
## 7  ( 1 ) " "   " "   " "   "*"   " "   " "   " "   " "   " "
## 8  ( 1 ) "*"   " "   " "   "*"   " "   " "   " "   " "   " "

```

Se puede incluir todas las variables con `nvmax = 19`.

```

regfit.full <- regsubsets(Salary ~ ., Hitters, nvmax = 19)
regfit.full.summary <- summary(regfit.full)
regfit.full.summary

```

```

## Subset selection object
## Call: regsubsets.formula(Salary ~ ., Hitters, nvmax = 19)
## 19 Variables (and intercept)
##          Forced in Forced out
## AtBat      FALSE      FALSE
## Hits       FALSE      FALSE
## HmRun      FALSE      FALSE
## Runs       FALSE      FALSE
## RBI        FALSE      FALSE
## Walks      FALSE      FALSE
## Years      FALSE      FALSE
## CAtBat     FALSE      FALSE
## CHits      FALSE      FALSE
## CHmRun     FALSE      FALSE
## CRuns      FALSE      FALSE
## CRBI       FALSE      FALSE
## CWalks     FALSE      FALSE
## LeagueN    FALSE      FALSE
## DivisionW  FALSE      FALSE
## PutOuts    FALSE      FALSE

```

```

## Assists      FALSE      FALSE
## Errors      FALSE      FALSE
## NewLeagueN  FALSE      FALSE
## 1 subsets of each size up to 19
## Selection Algorithm: exhaustive
##          AtBat Hits HmRun Runs RBI Walks Years CAtBat CHits CHmRun CRuns C
## 1  ( 1 ) " " " " " " " " " " " " " " " " " "
## 2  ( 1 ) " " "*" " " " " " " " " " " " " " "
## 3  ( 1 ) " " "*" " " " " " " " " " " " " " "
## 4  ( 1 ) " " "*" " " " " " " " " " " " " " "
## 5  ( 1 ) "*" "*" " " " " " " " " " " " " " "
## 6  ( 1 ) "*" "*" " " " " " " " " " *" " " " " "
## 7  ( 1 ) " " "*" " " " " " " " " " *" " " " " "
## 8  ( 1 ) "*" "*" " " " " " " " " " " *" " " " "
## 9  ( 1 ) "*" "*" " " " " " " " " " " *" " " " "
## 10 ( 1 ) "*" "*" " " " " " " " " " *" " " " " "
## 11 ( 1 ) "*" "*" " " " " " " " " " *" " " " " "
## 12 ( 1 ) "*" "*" " " " " *" " " " " *" " " " " "
## 13 ( 1 ) "*" "*" " " " " *" " " " " *" " " " " "
## 14 ( 1 ) "*" "*" " " " " *" " " " " *" " " " " "
## 15 ( 1 ) "*" "*" " " " " *" " " " " *" " " " " "
## 16 ( 1 ) "*" "*" " " " " *" " " *" " " *" " " " "
## 17 ( 1 ) "*" "*" " " " " *" " " *" " " *" " " " "
## 18 ( 1 ) "*" "*" " " " " *" " " *" " " *" " " " "
## 19 ( 1 ) "*" "*" " " " " *" " " *" " " *" " " " "
##          CWalks LeagueN DivisionW PutOuts Assists Errors NewLeagueN
## 1  ( 1 ) " " " " " " " " " " " "
## 2  ( 1 ) " " " " " " " " " " "
## 3  ( 1 ) " " " " " " *" " " " " "
## 4  ( 1 ) " " " " " *" " " *" " " " "
## 5  ( 1 ) " " " " " *" " " *" " " " "
## 6  ( 1 ) " " " " " *" " " *" " " " "
## 7  ( 1 ) " " " " " *" " " *" " " " "
## 8  ( 1 ) "*" " " " " *" " " *" " " " "
## 9  ( 1 ) "*" " " " " *" " " *" " " " "
## 10 ( 1 ) "*" " " " " *" " " *" " " " "
## 11 ( 1 ) "*" " " " " *" " " *" " " " "
## 12 ( 1 ) "*" " " " " *" " " *" " " " "

```

```

## 13  ( 1 ) "*"   "*"   "*"   "*"   "*"   "*"   "*"   " "
## 14  ( 1 ) "*"   "*"   "*"   "*"   "*"   "*"   "*"   " "
## 15  ( 1 ) "*"   "*"   "*"   "*"   "*"   "*"   "*"   " "
## 16  ( 1 ) "*"   "*"   "*"   "*"   "*"   "*"   "*"   " "
## 17  ( 1 ) "*"   "*"   "*"   "*"   "*"   "*"   "*"   "*"
## 18  ( 1 ) "*"   "*"   "*"   "*"   "*"   "*"   "*"   "*"
## 19  ( 1 ) "*"   "*"   "*"   "*"   "*"   "*"   "*"   "*"

```

```
str(regfit.full.summary)
```

```

## List of 8
## $ which : logi [1:19, 1:20] TRUE TRUE TRUE TRUE TRUE TRUE ...
## ..- attr(*, "dimnames")=List of 2
## ...$ : chr [1:19] "1" "2" "3" "4" ...
## ...$ : chr [1:20] "(Intercept)" "AtBat" "Hits" "HmRun" ...
## $ rsq : num [1:19] 0.321 0.425 0.451 0.475 0.491 ...
## $ rss : num [1:19] 36179679 30646560 29249297 27970852 27149899 ...
## $ adjr2 : num [1:19] 0.319 0.421 0.445 0.467 0.481 ...
## $ cp : num [1:19] 104.3 50.7 38.7 27.9 21.6 ...
## $ bic : num [1:19] -90.8 -128.9 -135.6 -141.8 -144.1 ...
## $ outmat: chr [1:19, 1:19] " " " " " " ...
## ..- attr(*, "dimnames")=List of 2
## ...$ : chr [1:19] "1 ( 1 )" "2 ( 1 )" "3 ( 1 )" "4 ( 1 )" ...
## ...$ : chr [1:19] "AtBat" "Hits" "HmRun" "Runs" ...
## $ obj :List of 28
## ...$ np : int 20
## ...$ nrbar : int 190
## ...$ d : num [1:20] 2.63e+02 1.10e+08 1.61e+05 1.85e+07 5.58e+03 ...
## ...$ rbar : num [1:190] 722.19 51.49 290.71 11.62 7.31 ...
## ...$ thetab : num [1:20] 535.926 0.382 5.509 0.306 -4.051 ...
## ...$ first : int 2
## ...$ last : int 20
## ...$ vorder : int [1:20] 1 10 6 17 4 8 19 16 5 15 ...
## ...$ tol : num [1:20] 8.11e-09 1.71e-05 7.91e-07 7.08e-06 1.94e-07 ...
## ...$ rss : num [1:20] 53319113 37253973 32381808 30651377 30559801 ...
## ...$ bound : num [1:20] 53319113 36179679 30646560 29249297 27970852 ...
## ...$ nvmax : int 20

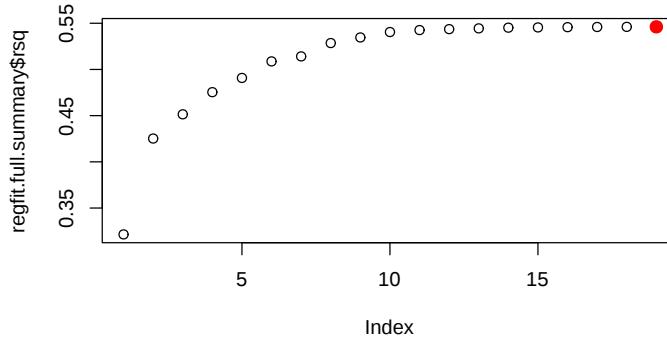
```

```

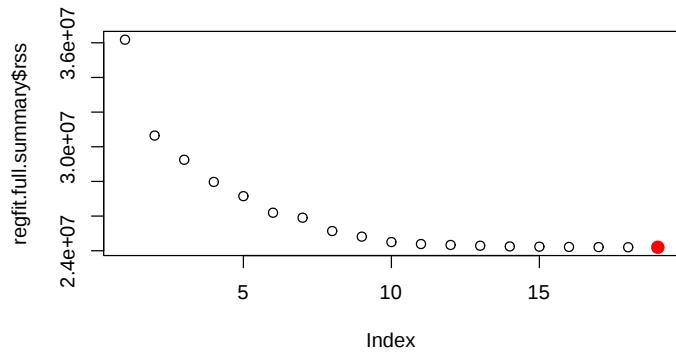
## ..$ ress      : num [1:20, 1] 53319113 36179679 30646560 29249297 27970852
## ..$ ir        : int 20
## ..$ nbest     : int 1
## ..$ lopt      : int [1:210, 1] 1 1 13 1 3 13 1 3 17 13 ...
## ..$ il        : int 210
## ..$ ier       : int 0
## ..$ xnames    : chr [1:20] "(Intercept)" "AtBat" "Hits" "HmRun" ...
## ..$ method    : chr "exhaustive"
## ..$ force.in  : Named logi [1:20] TRUE FALSE FALSE FALSE FALSE FALSE ...
## ... - attr(*, "names")= chr [1:20] "" "AtBat" "Hits" "HmRun" ...
## ..$ force.out : Named logi [1:20] FALSE FALSE FALSE FALSE FALSE FALSE ...
## ... - attr(*, "names")= chr [1:20] "" "AtBat" "Hits" "HmRun" ...
## ..$ sserr     : num 24200700
## ..$ intercept: logi TRUE
## ..$ lindep    : logi [1:20] FALSE FALSE FALSE FALSE FALSE FALSE ...
## ..$ nullrss   : num 53319113
## ..$ nn        : int 263
## ..$ call      : language regsubsets.formula(Salary ~ ., Hitters, nvmax = 1
## ... - attr(*, "class")= chr "regsubsets"
## - attr(*, "class")= chr "summary.regsubsets"

idx <- which.max(regfit.full.summary$rsq)
plot(regfit.full.summary$rsq)
points(idx, regfit.full.summary$rsq[idx], col = "red",
       cex = 2, pch = 20)

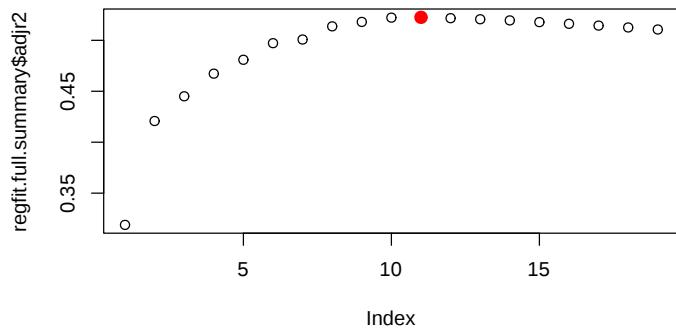
```



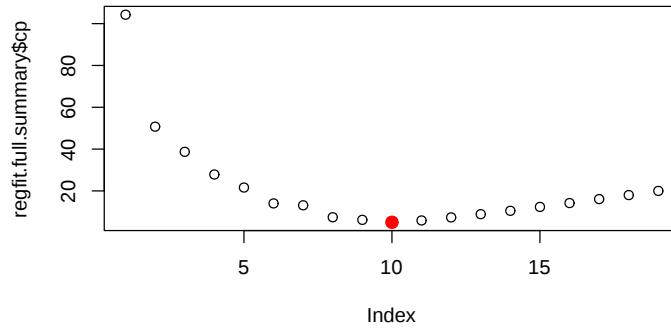
```
idx <- which.min(regfit.full.summary$rss)
plot(regfit.full.summary$rss)
points(idx, regfit.full.summary$rss[idx], col = "red",
       cex = 2, pch = 20)
```



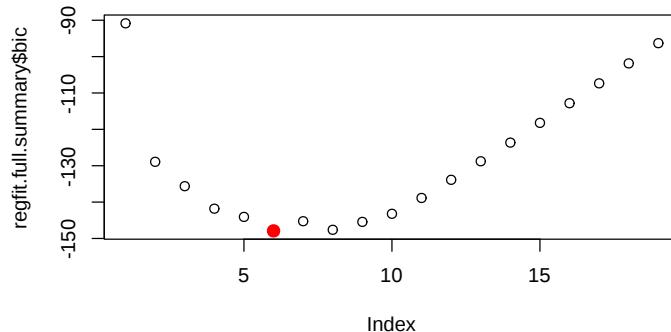
```
idx <- which.max(regfit.full.summary$adjr2)
plot(regfit.full.summary$adjr2)
points(idx, regfit.full.summary$adjr2[idx], col = "red",
       cex = 2, pch = 20)
```



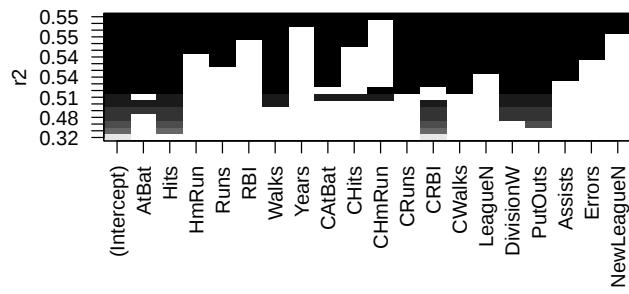
```
idx <- which.min(regfit.full.summary$cp)
plot(regfit.full.summary$cp)
points(idx, regfit.full.summary$cp[idx], col = "red",
       cex = 2, pch = 20)
```



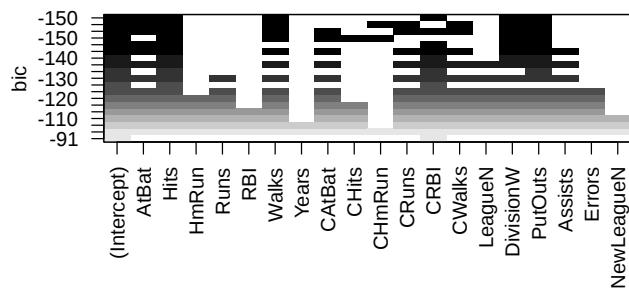
```
idx <- which.min(regfit.full.summary$bic)
plot(regfit.full.summary$bic)
points(idx, regfit.full.summary$bic[idx], col = "red",
       cex = 2, pch = 20)
```



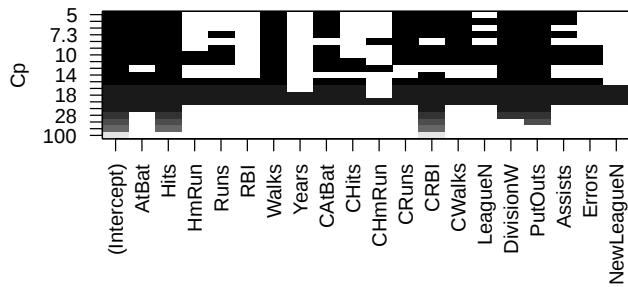
```
plot(regfit.full, scale = "r2")
```



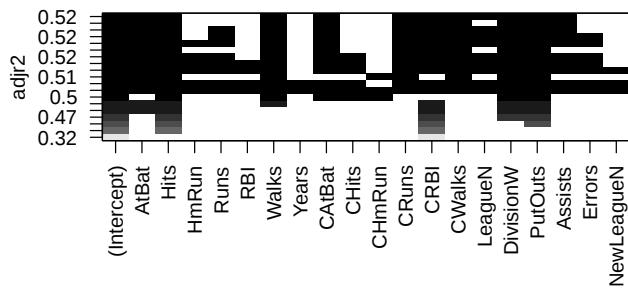
```
plot(regfit.full, scale = "bic")
```



```
plot(regfit.full, scale = "Cp")
```



```
plot(regfit.full, scale = "adjr2")
```



```
coef(regfit.full, 10)
```

##	(Intercept)	AtBat	Hits	Walks	CAtBat	CRBI
##	162.5354420	-2.1686501	6.9180175	5.7732246	-0.1300798	1.40824
##		CRBI	CWalks	DivisionW	PutOuts	Assists
##	0.7743122	-0.8308264	-112.3800575	0.2973726	0.2831680	

7.3.2.3. Regresión forward y backward

La función `regsubsets` tiene un parámetro `method`. Usen los valores `forward` y `backward` y comparen los resultados.

Puede guardar estos modelos en ciertas variables (e.g. `regfit.fwd` y `regfit.bwd`) y usar la función `plot`.

```
regfit.fwd <- regsubsets(Salary ~ ., data = Hitters,
                         nvmax = 19, method = "forward")
summary(regfit.fwd)
```

```
## Subset selection object
## Call: regsubsets.formula(Salary ~ ., data = Hitters, nvmax = 19, method = "forward")
## 19 Variables (and intercept)
##          Forced in Forced out
## AtBat      FALSE      FALSE
## Hits       FALSE      FALSE
## HmRun      FALSE      FALSE
## Runs       FALSE      FALSE
## RBI        FALSE      FALSE
## Walks      FALSE      FALSE
## Years      FALSE      FALSE
## CAtBat     FALSE      FALSE
## CHits      FALSE      FALSE
## CHmRun     FALSE      FALSE
## CRuns      FALSE      FALSE
## CRBI       FALSE      FALSE
## CWalks     FALSE      FALSE
## LeagueN    FALSE      FALSE
## DivisionW  FALSE      FALSE
## PutOuts    FALSE      FALSE
## Assists    FALSE      FALSE
## Errors     FALSE      FALSE
## NewLeagueN FALSE      FALSE
## 1 subsets of each size up to 19
## Selection Algorithm: forward
##          AtBat Hits HmRun Runs RBI Walks Years CAtBat CHits CHmRun CRuns CRBI
```

```

## 1 ( 1 ) " " " " " " " " " " " " " " " " " " " " " "
## 2 ( 1 ) " " "*" " " " " " " " " " " " " " " " " " "
## 3 ( 1 ) " " "*" " " " " " " " " " " " " " " " " " "
## 4 ( 1 ) " " "*" " " " " " " " " " " " " " " " " " "
## 5 ( 1 ) "*" "*" " " " " " " " " " " " " " " " " " "
## 6 ( 1 ) "*" "*" " " " " " " " " " " " " " " " " " "
## 7 ( 1 ) "*" "*" " " " " " " " " " " " " " " " " " "
## 8 ( 1 ) "*" "*" " " " " " " " " " " " " " " " " " "
## 9 ( 1 ) "*" "*" " " " " " " " " " " " " " " " " "
## 10 ( 1 ) "*" "*" " " " " " " " " " " " " " " " " "
## 11 ( 1 ) "*" "*" " " " " " " " " " " " " " " " "
## 12 ( 1 ) "*" "*" " " " " " " " " " " " " " " "
## 13 ( 1 ) "*" "*" " " " " " " " " " " " " " "
## 14 ( 1 ) "*" "*" " " " " " " " " " " " "
## 15 ( 1 ) "*" "*" " " " " " " " " " " "
## 16 ( 1 ) "*" "*" " " " " " " " " " "
## 17 ( 1 ) "*" "*" " " " " " " " " "
## 18 ( 1 ) "*" "*" " " " " " " " "
## 19 ( 1 ) "*" "*" " " " " " " "
##          CWalks LeagueN DivisionW PutOuts Assists Errors NewLeagueN
## 1 ( 1 ) " " " " " " " " " " " " "
## 2 ( 1 ) " " " " " " " " " " " "
## 3 ( 1 ) " " " " " " " " " " "
## 4 ( 1 ) " " " " " " " " " "
## 5 ( 1 ) " " " " " " " " "
## 6 ( 1 ) " " " " " " "
## 7 ( 1 ) "*" " " " " "
## 8 ( 1 ) "*" " " " "
## 9 ( 1 ) "*" " " " "
## 10 ( 1 ) "*" " " " "
## 11 ( 1 ) "*" " " " "
## 12 ( 1 ) "*" " " " "
## 13 ( 1 ) "*" " " " "
## 14 ( 1 ) "*" " " " "
## 15 ( 1 ) "*" " " " "
## 16 ( 1 ) "*" " " " "
## 17 ( 1 ) "*" " " " "
## 18 ( 1 ) "*" " " "

```

```

## 19  ( 1 ) "*"      "*"      "*"      "*"      "*"      "*"      "*"
regfit.bwd <- regsubsets(Salary ~ ., data = Hitters,
    nvmax = 19, method = "backward")
summary(regfit.bwd)

## Subset selection object
## Call: regsubsets.formula(Salary ~ ., data = Hitters, nvmax = 19, method = "backward")
## 19 Variables  (and intercept)
##          Forced in Forced out
## AtBat      FALSE      FALSE
## Hits       FALSE      FALSE
## HmRun      FALSE      FALSE
## Runs       FALSE      FALSE
## RBI        FALSE      FALSE
## Walks      FALSE      FALSE
## Years      FALSE      FALSE
## CAtBat     FALSE      FALSE
## CHits      FALSE      FALSE
## CHmRun     FALSE      FALSE
## CRuns      FALSE      FALSE
## CRBI       FALSE      FALSE
## CWalks     FALSE      FALSE
## LeagueN    FALSE      FALSE
## DivisionW  FALSE      FALSE
## PutOuts    FALSE      FALSE
## Assists    FALSE      FALSE
## Errors     FALSE      FALSE
## NewLeagueN FALSE      FALSE
## 1 subsets of each size up to 19
## Selection Algorithm: backward
##          AtBat  Hits  HmRun  Runs  RBI  Walks  Years  CAtBat  CHits  CHmRun  CRuns  CRBI
## 1  ( 1 )  " "   " "   " "   " "   " "   " "   " "   " "   " "   " "   " "   " * "   " "
## 2  ( 1 )  " "   "*"   " "   " "   " "   " "   " "   " "   " "   " "   " "   " * "   " "
## 3  ( 1 )  " "   "*"   " "   " "   " "   " "   " "   " "   " "   " "   " "   " * "   " "
## 4  ( 1 )  "*"   "*"   " "   " "   " "   " "   " "   " "   " "   " "   " "   " * "   " "
## 5  ( 1 )  "*"   "*"   " "   " "   " "   " * "   " "   " "   " "   " "   " "   " * "   " "

```

```

## 6  ( 1 )   "*"   "*"   " "   " "   " "   "*"   " "   " "   " "   " "   " "   "*"   "
## 7  ( 1 )   "*"   "*"   " "   " "   " "   "*"   " "   " "   " "   " "   " "   "*"   "
## 8  ( 1 )   "*"   "*"   " "   " "   " "   "*"   " "   " "   " "   " "   " "   "*"   "
## 9  ( 1 )   "*"   "*"   " "   " "   " "   "*"   " "   " "   " "   "*"   " "   " "   "*"   "
## 10 ( 1 )   "*"   "*"   " "   " "   " "   "*"   " "   " "   "*"   " "   " "   "*"   "
## 11 ( 1 )   "*"   "*"   " "   " "   " "   "*"   " "   " "   "*"   " "   " "   "*"   "
## 12 ( 1 )   "*"   "*"   " "   "*"   " "   "*"   " "   " "   "*"   " "   " "   "*"   "
## 13 ( 1 )   "*"   "*"   " "   "*"   " "   "*"   " "   " "   "*"   " "   " "   "*"   "
## 14 ( 1 )   "*"   "*"   "*"   "*"   " "   "*"   " "   " "   "*"   " "   " "   "*"   "
## 15 ( 1 )   "*"   "*"   "*"   "*"   " "   "*"   " "   " "   "*"   " "   " "   "*"   "
## 16 ( 1 )   "*"   "*"   "*"   "*"   " "   "*"   " "   " "   "*"   " "   " "   "*"   "
## 17 ( 1 )   "*"   "*"   "*"   "*"   " "   "*"   " "   " "   "*"   " "   " "   "*"   "
## 18 ( 1 )   "*"   "*"   "*"   "*"   " "   "*"   " "   " "   "*"   " "   " "   "*"   "
## 19 ( 1 )   "*"   "*"   "*"   "*"   " "   "*"   " "   " "   "*"   " "   " "   "*"   "
##          CWalks LeagueN DivisionW PutOuts Assists Errors NewLeagueN
## 1  ( 1 )   " "   " "   " "   " "   " "   " "   " "   " "   " "
## 2  ( 1 )   " "   " "   " "   " "   " "   " "   " "   " "   " "
## 3  ( 1 )   " "   " "   " "   " "   "*"   " "   " "   " "   " "
## 4  ( 1 )   " "   " "   " "   " "   "*"   " "   " "   " "   " "
## 5  ( 1 )   " "   " "   " "   " "   "*"   " "   " "   " "   " "
## 6  ( 1 )   " "   " "   " "   "*"   " "   "*"   " "   " "   " "
## 7  ( 1 )   "*"   " "   " "   "*"   " "   "*"   " "   " "   " "
## 8  ( 1 )   "*"   " "   " "   "*"   " "   "*"   " "   " "   " "
## 9  ( 1 )   "*"   " "   " "   "*"   " "   "*"   " "   " "   " "
## 10 ( 1 )   "*"   " "   " "   "*"   " "   "*"   " "   " "   " "
## 11 ( 1 )   "*"   " "   " "   "*"   " "   "*"   " "   " "   " "
## 12 ( 1 )   "*"   " "   " "   "*"   " "   "*"   " "   " "   " "
## 13 ( 1 )   "*"   " "   " "   "*"   " "   "*"   " "   " "   " "
## 14 ( 1 )   "*"   " "   " "   "*"   " "   "*"   " "   " "   " "
## 15 ( 1 )   "*"   " "   " "   "*"   " "   "*"   " "   " "   " "
## 16 ( 1 )   "*"   " "   " "   "*"   " "   "*"   " "   " "   " "
## 17 ( 1 )   "*"   " "   " "   "*"   " "   "*"   " "   " "   " "
## 18 ( 1 )   "*"   " "   " "   "*"   " "   "*"   " "   " "   " "
## 19 ( 1 )   "*"   " "   " "   "*"   " "   "*"   " "   " "   " "

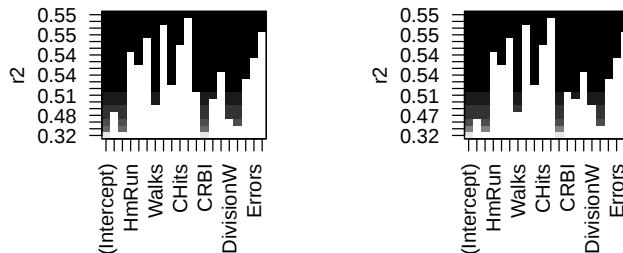
```

```

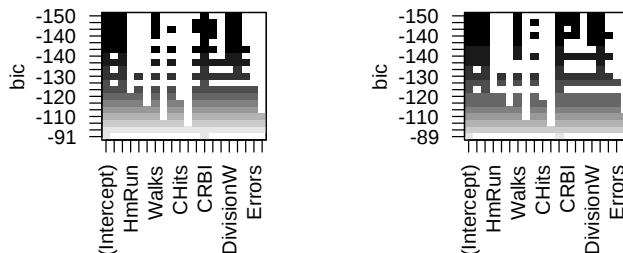
par(mfrow = c(1, 2))
plot(regfit.fwd, scale = "r2")

```

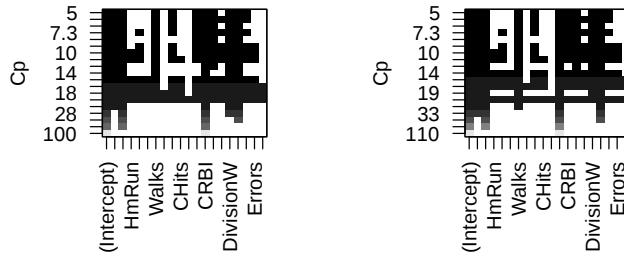
```
plot(regfit.bwd, scale = "r2")
```



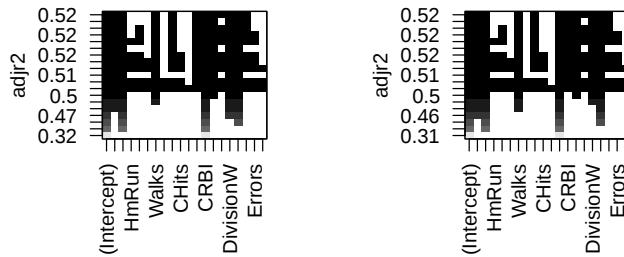
```
par(mfrow = c(1, 2))
plot(regfit.fwd, scale = "bic")
plot(regfit.bwd, scale = "bic")
```



```
par(mfrow = c(1, 2))
plot(regfit.fwd, scale = "Cp")
plot(regfit.bwd, scale = "Cp")
```



```
par(mfrow = c(1, 2))
plot(regfit.fwd, scale = "adjr2")
plot(regfit.bwd, scale = "adjr2")
```



7.3.2.4. Regresión Ridge

```
x <- model.matrix(Salary ~ ., Hitters) [, -1]
y <- Hitters$Salary
```

Usando el paquete `glmnet` y la función con el mismo nombre, ejecute el siguiente comando

```
library(glmnet)
grid <- 10^seq(10, -2, length = 100)
ridge.mod <- glmnet(x, y, alpha = 0, lambda = grid)
```

El factor `lambda` representa el λ de la fórmula

$$\hat{\beta} = \operatorname{argmin}_{\beta} \left\{ RSS + \lambda \|\beta\|_2^2 \right\}.$$

Si no se incluye el parámetro `lambda` del modelo, R construye una secuencia de λ 's estimados por validación cruzada.

Haga lo siguiente:

1. Construya un modelo usando todos los datos (sin separar muestra de entrenamiento y prueba).
2. Construya el siguiente modelo

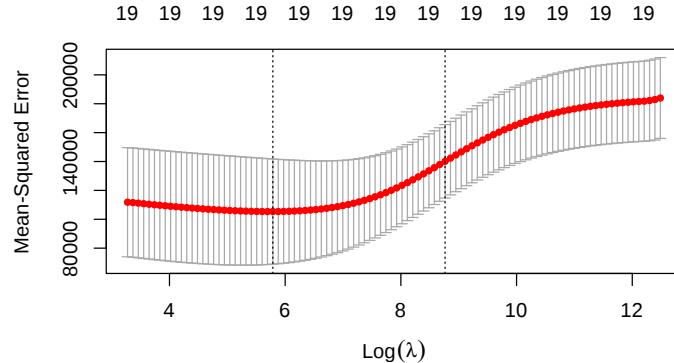
```
set.seed(1)
train <- sample(1:nrow(x), nrow(x)/2)
test <- -train
y.test <- y[test]
ridge.mod <- glmnet(x[train, ], y[train], alpha = 0,
                      lambda = grid)
ridge.pred <- predict(ridge.mod, s = 4, newx = x[test,
                                                ])
# MSE
mean((ridge.pred - y.test)^2)

## [1] 142226.5
```

¿Qué ocurre si se cambia el parámetro `s` de `predict` por un `10e10` (i.e. 10^{10}). Comente los resultados. ¿Y qué ocurre si `s = 0`?

3. Finalmente, ejecute el siguiente código

```
set.seed(1)
cv.out <- cv.glmnet(x[train, ], y[train], alpha = 0)
plot(cv.out)
```



Busque la ayuda de `cv.glmnet` y deduzca qué significa el gráfico.

```
library(glmnet)
grid <- 10^seq(10, -2, length = 100)
ridge.mod <- glmnet(x, y, alpha = 0, lambda = grid)
```

```
ridge.mod$lambda[50]
```

```
## [1] 11497.57
```

```
coef(ridge.mod) [, 50]
```

	(Intercept)	AtBat	Hits	HmRun	Runs
##	407.356050200	0.036957182	0.138180344	0.524629976	0.230701523
##	RBI	Walks	Years	CATBat	CHits
##	0.239841459	0.289618741	1.107702929	0.003131815	0.011653637
##	CHmRun	CRuns	CRBI	CWalks	LeagueN
##	0.087545670	0.023379882	0.024138320	0.025015421	0.085028114
##	DivisionW	PutOuts	Assists	Errors	NewLeagueN
##	-6.215440973	0.016482577	0.002612988	-0.020502690	0.301433531

```
sqrt(sum(coef(ridge.mod)[-1, 50]^2))
```

```
## [1] 6.360612
```

```
ridge.mod$lambda[60]
```

```
## [1] 705.4802
```

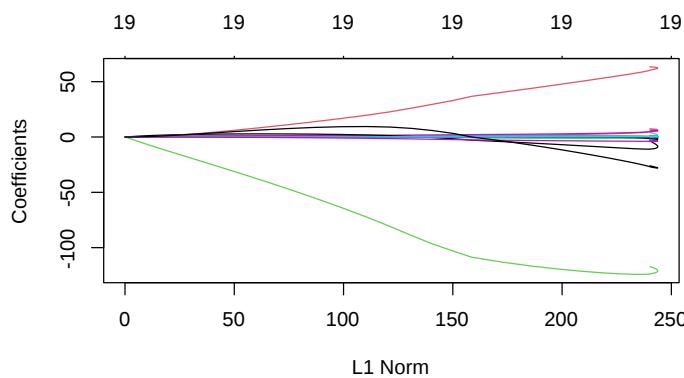
```
coef(ridge.mod) [, 60]
```

	(Intercept)	AtBat	Hits	HmRun	Runs	RBI
##	54.32519950	0.11211115	0.65622409	1.17980910	0.93769713	0.84718546
##	Walks	Years	CAtBat	CHits	CHmRun	CRuns
##	1.31987948	2.59640425	0.01083413	0.04674557	0.33777318	0.09355528
##	CRBI	CWalks	LeagueN	DivisionW	PutOuts	Assists
##	0.09780402	0.07189612	13.68370191	-54.65877750	0.11852289	0.01606037
##	Errors	NewLeagueN				
##	-0.70358655	8.61181213				

```
sqrt(sum(coef(ridge.mod)[-1, 60]^2))
```

```
## [1] 57.11001
```

```
plot(ridge.mod)
```

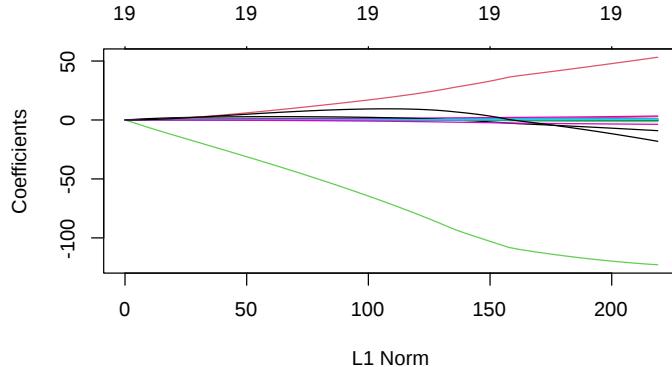


```
ridge.mod <- glmnet(x, y, alpha = 0)
```

```
ridge.mod$lambda
```

```
## [1] 255282.09651 232603.53866 211939.68139 193111.54424 175956.04690
## [6] 160324.59666 146081.80138 133104.29678 121279.67791 110505.52560
## [11] 100688.51928 91743.62874 83593.37763 76167.17236 69400.69070
## [16] 63235.32462 57617.67267 52499.07743 47835.20409 43585.65640
## [21] 39713.62682 36185.57767 32970.95069 30041.90230 27373.06250
## [26] 24941.31507 22725.59739 20706.71795 18867.19020 17191.08102
## [31] 15663.87277 14272.33748 13004.42236 11849.14532 10796.49991
## [36] 9837.36861 8963.44390 8167.15625 7441.60860 6780.51660
## [41] 6178.15419 5629.30400 5129.21215 4673.54708 4258.36204
## [46] 3880.06089 3535.36698 3221.29472 2935.12377 2674.37547
## [51] 2436.79132 2220.31350 2023.06697 1843.34327 1679.58574
## [56] 1530.37597 1394.42159 1270.54502 1157.67330 1054.82879
## [61] 961.12071 875.73740 797.93930 727.05257 662.46322
## [66] 603.61182 549.98861 501.12914 456.61020 416.04621
## [71] 379.08581 345.40887 314.72370 286.76452 261.28915
## [76] 238.07694 216.92684 197.65566 180.09647 164.09720
## [81] 149.51926 136.23638 124.13351 113.10583 103.05782
## [86] 93.90245 85.56042 77.95946 71.03376 64.72332
## [91] 58.97348 53.73443 48.96082 44.61127 40.64813
## [96] 37.03706 33.74679 30.74882 28.01718 25.52821
```

```
plot(ridge.mod)
```



```
predict(ridge.mod, s = 50, type = "coefficients")
```

```
## 20 x 1 sparse Matrix of class "dgCMatrix"
##                                     1
## (Intercept) 4.821654e+01
## AtBat      -3.538650e-01
## Hits       1.953167e+00
## HmRun      -1.285127e+00
## Runs       1.156329e+00
## RBI        8.087771e-01
## Walks      2.709765e+00
## Years      -6.202919e+00
## CAtBat     6.085854e-03
## CHits      1.070832e-01
## CHmRun     6.290984e-01
## CRuns      2.172926e-01
## CRBI       2.152888e-01
## CWalks     -1.488961e-01
## LeagueN    4.586262e+01
## DivisionW -1.182304e+02
## PutOuts    2.501647e-01
## Assists    1.208491e-01
## Errors     -3.277073e+00
## NewLeagueN -9.423459e+00
```

```

set.seed(1)
train <- sample(1:nrow(x), nrow(x)/2)
test <- -train
y.test <- y[test]
ridge.mod <- glmnet(x[train, ], y[train], alpha = 0,
                      lambda = grid)
ridge.pred <- predict.glmnet(ridge.mod, s = 4, newx = x[test,
    ], exact = FALSE)

# MSE
mean((ridge.pred - y.test)^2)

## [1] 142226.5

ridge.pred <- predict(ridge.mod, s = 1e+10, newx = x[test,
    ], exact = TRUE)
mean((ridge.pred - y.test)^2)

## [1] 224669.8

ridge.pred <- predict(ridge.mod, s = 0, newx = x[test,
    ], exact = FALSE)
mean((ridge.pred - y.test)^2)

## [1] 166258.3

lm(y ~ x, subset = train)

##
## Call:
## lm(formula = y ~ x, subset = train)
##
## Coefficients:
## (Intercept)      xAtBat      xHits      xHmRun      xRuns
## xRB

```

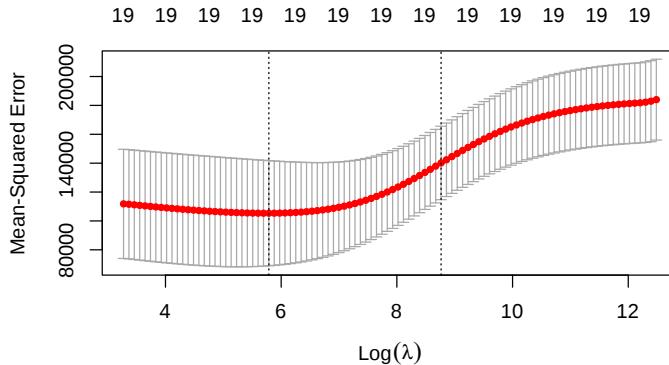
```
##      274.0145      -0.3521     -1.6377      5.8145      1.5424      1.1243
##      xWalks       xYears     xCAtBat     xCHits     xCHmRun     xCRuns
##      3.7287     -16.3773     -0.6412      3.1632      3.4008     -0.9739
##      xCRBI       xCWalks    xLeagueN    xDivisionW    xPutOuts    xAssists
##      -0.6005      0.3379     119.1486     -144.0831      0.1976      0.6804
##      xErrors    xNewLeagueN
##      -4.7128     -71.0951

predict(ridge.mod, s = 0, type = "coefficients", exact = FALSE)

## 20 x 1 sparse Matrix of class "dgCMatrix"
##                               1
## (Intercept) 275.7172204
## AtBat        -0.4118004
## Hits         -1.3490261
## HmRun         6.0905436
## Runs          1.4271781
## RBI           0.9898408
## Walks         3.8088184
## Years        -17.2171582
## CAtBat        -0.6086644
## CHits          3.0142526
## CHmRun         3.1935999
## CRuns          -0.9320183
## CRBI           -0.5105934
## CWalks          0.3200890
## LeagueN        116.5487876
## DivisionW     -145.0182286
## PutOuts         0.1959973
## Assists         0.6722372
## Errors          -4.6629290
## NewLeagueN     -68.0666960

set.seed(1)
cv.out <- cv.glmnet(x[train, ], y[train], alpha = 0)

plot(cv.out)
```



```

bestlam <- cv.out$lambda.min
bestlam

## [1] 326.0828

log(bestlam)

## [1] 5.787151

ridge.pred <- predict(ridge.mod, s = bestlam, newx = x[test,
  ])
mean((ridge.pred - y.test)^2)

## [1] 139833.6

out <- glmnet(x, y, alpha = 0)
predict(out, type = "coefficients", s = bestlam, exact = FALSE)[1:20,
  ]

## (Intercept)      AtBat      Hits      HmRun      Runs
## 15.44383135  0.07715547  0.85911581  0.60103107  1.06369007  0.879361
##       Walks      Years     CAtBat     CHits     CChmRun
## 
```

```

##   1.62444616  1.35254780  0.01134999  0.05746654  0.40680157  0.11456224
##      CRBI        CWalks     LeagueN    DivisionW     PutOuts     Assists
##  0.12116504  0.05299202 22.09143189 -79.04032637  0.16619903  0.02941950
##      Errors    NewLeagueN
## -1.36092945  9.12487767

```

7.3.3. Regresión Lasso

Ejecute los procedimientos anteriores con `glmnet` pero modifique el parámetro `alpha = 0`. Compare los resultados.

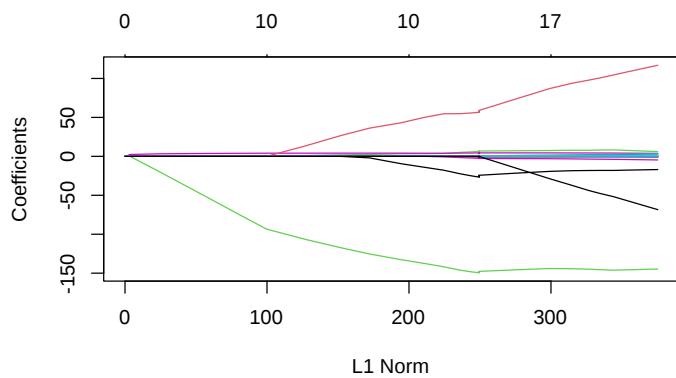
En este caso, se están encontrando los valores de β tal que

$$\hat{\beta} = \underset{\beta}{\operatorname{argmin}} \left\{ RSS + \lambda \|\beta\|_1^2 \right\}.$$

```

lasso.mod <- glmnet(x[train, ], y[train], alpha = 1,
                      lambda = grid)
plot(lasso.mod)

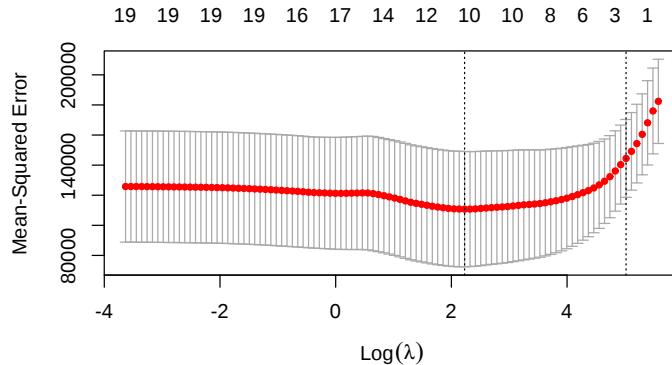
```



```

set.seed(1)
cv.out <- cv.glmnet(x[train, ], y[train], alpha = 1)
plot(cv.out)

```



```

bestlam <- cv.out$lambda.min
bestlam

## [1] 9.286955

log(bestlam)

## [1] 2.228611

lasso.pred <- predict(lasso.mod, s = bestlam, newx = x[test,
  ])
mean((lasso.pred - y.test)^2)

## [1] 143673.6

out <- glmnet(x, y, alpha = 1, lambda = grid)
lasso.coef <- predict(out, type = "coefficients", s = bestlam)[1:20,
  ]
lasso.coef

##      (Intercept)          AtBat          Hits         HmRun         Runs
##      1.27479059     -0.05497143     2.18034583    0.00000000    0.00000000

```

```
##          RBI          Walks         Years        CATBat        CHits
## 0.00000000  2.29192406 -0.33806109  0.00000000  0.00000000
##      CHmRun       CRuns        CRBI        CWalks       LeagueN
## 0.02825013  0.21628385  0.41712537  0.00000000 20.28615023
##   DivisionW     PutOuts      Assists      Errors    NewLeagueN
## -116.16755870 0.23752385  0.00000000 -0.85629148  0.00000000

lasso.coef[lasso.coef != 0]

## (Intercept)        AtBat         Hits        Walks         Years
## 1.27479059 -0.05497143  2.18034583  2.29192406 -0.33806109
##      CHmRun       CRuns        CRBI       LeagueN      DivisionW
## 0.02825013  0.21628385  0.41712537 20.28615023 -116.16755870
##     PutOuts      Errors
## 0.23752385 -0.85629148
```

7.4. Ejercicios

- Del libro (James y col. 2013)
 - Capítulo 5: 2, 5, 8.
 - Capítulo 6: 5, 6, 7, 8, 10.

Capítulo 8

Análisis en componentes principales

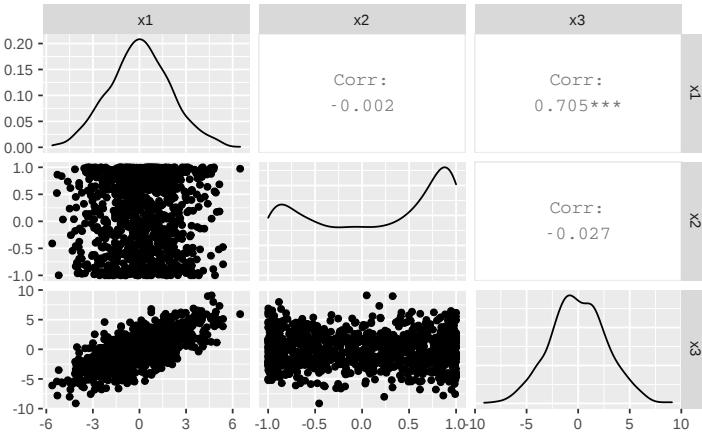
Si se quiere seleccionar la mejor proyección de 2 variables de una nube de puntos X_1, \dots, X_p , se debe hacer $\binom{p}{2}$ gráficos de dispersión. Se debe seleccionar la que tenga menor varianza, es decir, en los siguientes gráficos, la mejor foto sería la que tenga la mayor variabilidad.

Usaremos como base los libros de (Husson, Le y Pagès 2017) y (James y col. 2013).

```
library(rgl)
library(car)
knitr::knit_hooks$set(webgl = hook_webgl, rgl = hook_rgl)
knitr::opts_chunk$set(fig.pos = "!h")
```

```
set.seed(123)
x1 <- rnorm(1000, 0, 2)
x2 <- cos(rnorm(1000, 0, 2))
x3 <- x1 + rnorm(1000, 0, 2)
```

```
GGally::ggpairs(data.frame(x1, x2, x3))
```



```
plot3d(x1, x3, x2, point.col = "black")
plot3d(scale(x1), scale(x2), scale(x3), point.col = "black")
```

El ACP lo que busca es un número reducido de dimensión que represente el máximo de variabilidad en las observaciones eliminando la mayor cantidad de ruido posible.

8.1. Representación gráfica

8.2. Primer componente principal

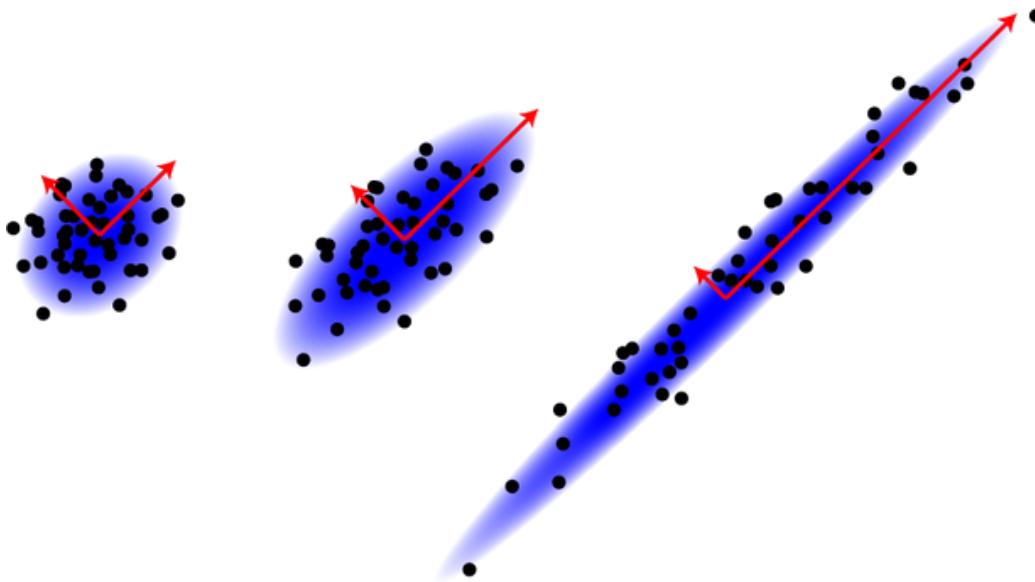
$$Z_1 := \phi_{11}x_1 + \phi_{21}x_2 + \cdots + \phi_{p1}x_p; \quad \text{con } \sum_{j=1}^p \phi_{j1} = 1$$

tal que Z_1 tenga la varianza máxima.

Al vector $\phi_1 = (\phi_{11}, \phi_{21}, \dots, \phi_{p1})$ se le llama *paso o cargas*.

$X = (X_1, \dots, X_p)_{n \times p}$ es la *matriz de diseño* donde cada columna tiene media 0. Se resuelve el problema

$$\hat{\phi}_1 = \underset{\|\phi_1\|_2^2=1}{\operatorname{argmax}} \left\{ \frac{1}{n} \sum_{i=1}^n \left(\sum_{j=1}^p \phi_{j1} X_{ij} \right)^2 \right\}$$

Figura 8.1: Tomado de [The shape of data](#)

La restricción de minimización se puede describir como $\|\phi_1\|_2^2 = \sum_{j=1}^p \phi_{j1}^2 = 1$

Los Z_{11}, \dots, Z_{n1} son los scores del primer componente principal.

ϕ_1 es la dirección en el espacio característico en \mathbb{R}^p en donde los datos tengan la máxima varianza.

Esta última expresión se podría describir de forma matricial como

$$\hat{\phi}_1 = \underset{\|\phi_1\|_2^2=1}{\operatorname{argmax}} \left\{ \phi_1^\top X^\top X \phi_1 \right\}$$

donde $\phi_1 = (\phi_{11}, \phi_{21}, \dots, \phi_{p1})$

dadas las condiciones, esta expresión se podría simplificar un poco más en

$$\hat{\phi}_1 = \underset{\phi_1}{\operatorname{argmax}} \left\{ \frac{\phi_1^\top X^\top X \phi_1}{\phi_1^\top \phi_1} \right\}$$

Dado que la expresión anterior es un coeficiente de Rayleigh, se puede probar que $\hat{\phi}_1$ corresponde al primer vector propio de la matriz $X^\top X = \text{Cov}(X)$ si las columnas de X son centradas.

8.3. Segunda componente principal

$$Z_2 := \phi_{12}x_1 + \phi_{22}x_2 + \cdots + \phi_{p2}x_p$$

$$\underset{\|\phi_2\|_2^2=1}{\operatorname{argmax}} \left\{ \frac{1}{n} \sum_{i=1}^n \left(\sum_{j=1}^p \phi_{j2} X_{ij} \right)^2 \right\}$$

Se tiene, además, que $\forall i, Z_{i2} \perp Z_1$, entonces

$$Z_{i2} \perp Z_1 \implies \phi_2 \perp \phi_1$$

Esto se logra primero construyendo una matriz nueva de diseño, restando a la matrix X original, el primer componente principal.

$$\tilde{X}_2 = X - X\phi_1\phi_1^\top$$

Luego a esa matriz, se le aplica el procedimiento anterior

$$\hat{\phi}_2 = \underset{\phi_2}{\operatorname{argmax}} \left\{ \frac{\phi_2^\top X^\top X \phi_2}{\phi_2^\top \phi_2} \right\}$$

Y nuevamente se puede probar que el componente principal corresponde al segundo vector propio de $X^\top X = \text{Cov}(X)$

De la misma forma se construye $\phi_3, \phi_4, \dots, \phi_p$.

Notas:

- **Escalas:** la varianza de las variables depende de las unidades. El problema es que los pesos ϕ_i son distintos dependiendo de las escalas. La solución es estandarizar las variables: $\frac{X_i - \mu_i}{\hat{\sigma}_i}$.
- **Unicidad:** los componentes principales son únicos, módulo cambio de signo. \end{itemize}

8.4. Circulo de correlaciones

Se puede construir la correlación de cada variable con respecto a cada componente principal

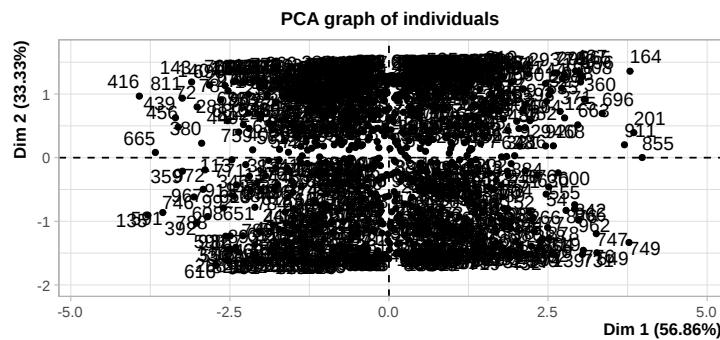
$$\cos(\theta_{i,j'}) = \text{Corr}(X_i, \text{PC}_{j'})$$

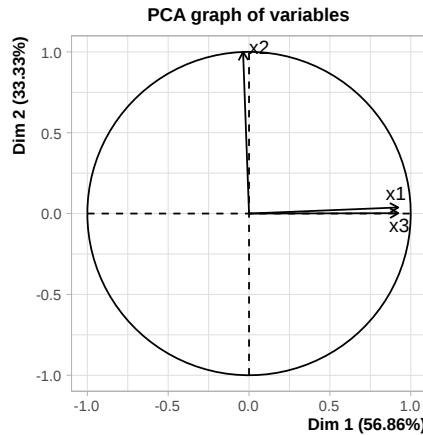
El ángulo $\theta_{i,j'}$ significa la lejanía o cercanía de cierta variable con respecto a cada componente principal.

Además, basados en el el círculo identidad $\cos^2(\theta) + \sin^2(\theta) = 1$, el valor de $\cos^2(\theta_{i,j'})$ representa la «intensidad» con la cual la variable X_i es representada por el componente principal $\text{PC}_{i'}$.

8.5. Volvamos a nuestro ejemplo

```
library("factoextra")
library("FactoMineR")
p <- PCA(scale(cbind(x1, x2, x3)))
```





```
p$var$cor
```

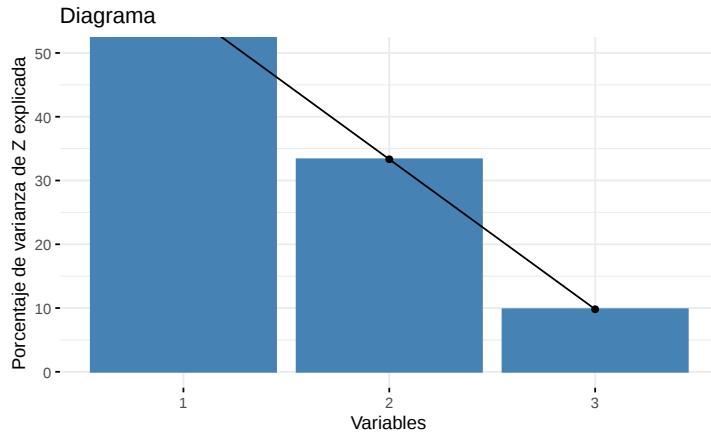
```
##           Dim.1      Dim.2      Dim.3
## x1  0.92280569 0.037753401 -0.38341145
## x2 -0.03690606 0.999225664  0.01363871
## x3  0.92346176 0.002207375  0.38368413
```

```
p$var$cos2
```

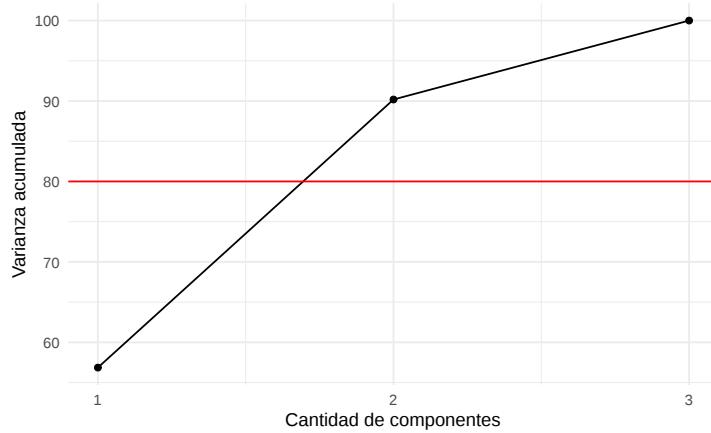
```
##           Dim.1      Dim.2      Dim.3
## x1  0.851570337 1.425319e-03 0.1470043434
## x2  0.001362057 9.984519e-01 0.0001860145
## x3  0.852781615 4.872503e-06 0.1472135129
```

8.6. ¿Cuántos componentes usar?

```
fviz_screeplot(p, addlabels = F, ylim = c(0, 50)) +
  xlab("Variables") + ylab("Porcentaje de varianza de Z explicada") +
  labs(title = "Diagrama")
```



```
qplot(1:3, p$eig[, 3], geom = "point") + xlab("Cantidad de componentes") +
  ylab("Varianza acumulada") + geom_line() + theme_minimal() +
  geom_hline(yintercept = 80, color = "red") + scale_x_continuous(breaks = 1:10)
```



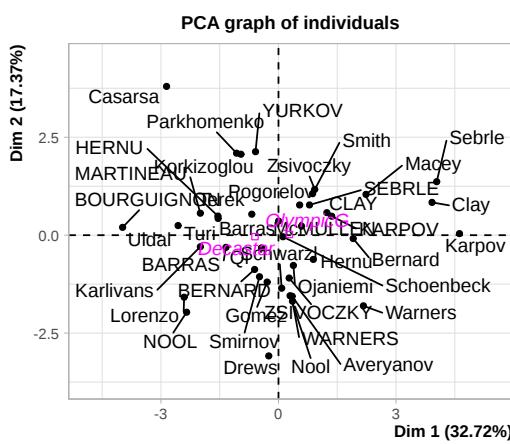
8.7. Laboratorio

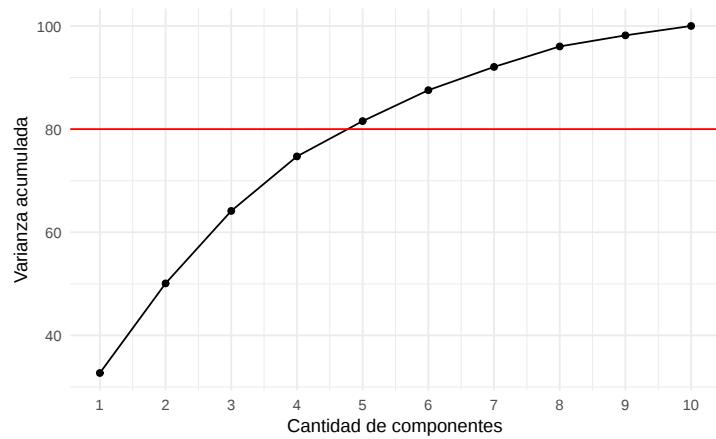
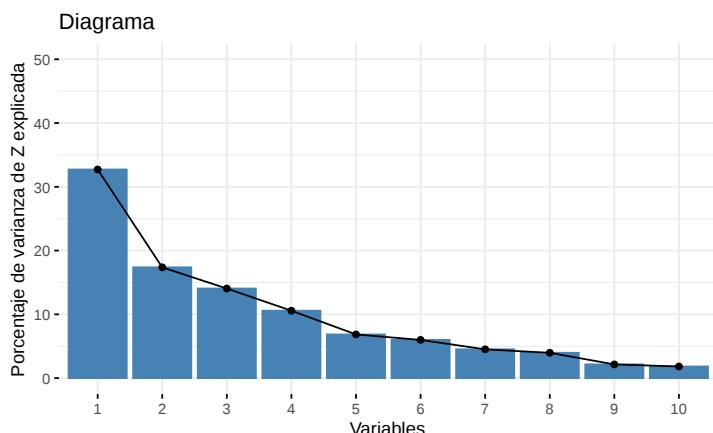
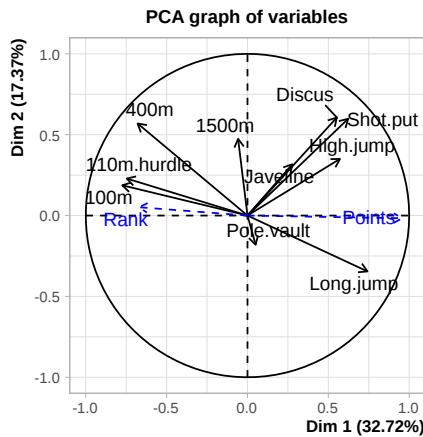
Vamos a usar los datos `decathlon` de `FactomineR` que representa los resultados de varios atletas en pruebas de decathlon en el 2004.

El objetivo es encontrar si hay patrones entre ciudad y tipos de crimen.

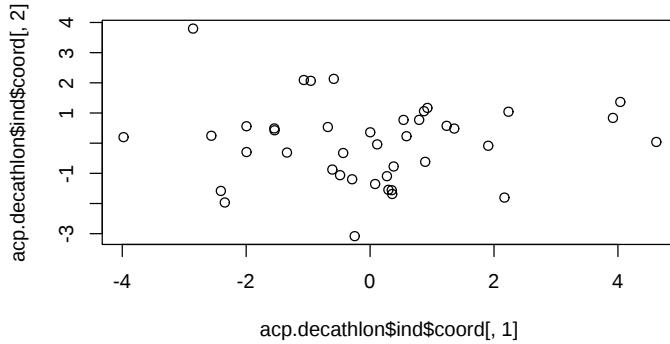
Exploración de datos Ejecute una exploración de datos

##	100m	Long.jump	Shot.put	High.jump	400m
## Min.	:10.44	Min. :6.61	Min. :12.68	Min. :1.850	Min. :46.
## 1st Qu.	:10.85	1st Qu.:7.03	1st Qu.:13.88	1st Qu.:1.920	1st Qu.:48.
## Median	:10.98	Median :7.30	Median :14.57	Median :1.950	Median :49.
## Mean	:11.00	Mean :7.26	Mean :14.48	Mean :1.977	Mean :49.
## 3rd Qu.	:11.14	3rd Qu.:7.48	3rd Qu.:14.97	3rd Qu.:2.040	3rd Qu.:50.
## Max.	:11.64	Max. :7.96	Max. :16.36	Max. :2.150	Max. :53.
##	110m.hurdle	Discus	Pole.vault	Javeline	
## Min.	:13.97	Min. :37.92	Min. :4.200	Min. :50.31	
## 1st Qu.	:14.21	1st Qu.:41.90	1st Qu.:4.500	1st Qu.:55.27	
## Median	:14.48	Median :44.41	Median :4.800	Median :58.36	
## Mean	:14.61	Mean :44.33	Mean :4.762	Mean :58.32	
## 3rd Qu.	:14.98	3rd Qu.:46.07	3rd Qu.:4.920	3rd Qu.:60.89	
## Max.	:15.67	Max. :51.65	Max. :5.400	Max. :70.52	
##	1500m	Rank	Points	Competition	
## Min.	:262.1	Min. : 1.00	Min. :7313	Decastar:13	
## 1st Qu.	:271.0	1st Qu.: 6.00	1st Qu.:7802	OlympicG:28	
## Median	:278.1	Median :11.00	Median :8021		
## Mean	:279.0	Mean :12.12	Mean :8005		
## 3rd Qu.	:285.1	3rd Qu.:18.00	3rd Qu.:8122		
## Max.	:317.0	Max. :28.00	Max. :8893		

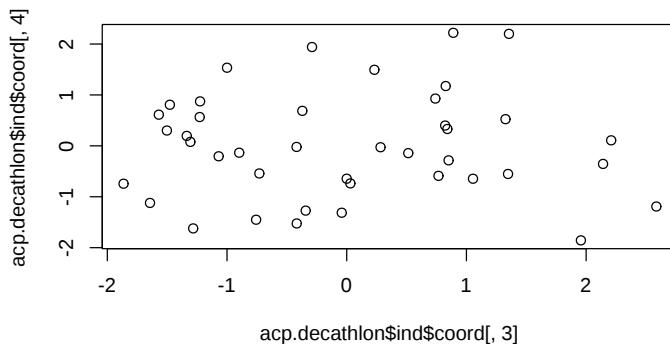




```
plot(acp.decathlon$ind$coord[, 1], acp.decathlon$ind$coord[, 2])
```



```
plot(acp.decathlon$ind$coord[, 3], acp.decathlon$ind$coord[, 4])
```



“”

8.8. Ejercicios

- Del libro (James y col. 2013)

- Capítulo 10: 6, 8 ‘

Bibliografía

- Albert, Jim y col. (2009). *Bayesian Computation with R*. New York, NY: Springer New York.
- Cavanaugh, Joseph E. y Andrew A. Neath (2019). «The Akaike Information Criterion: Background, Derivation, Properties, Application, Interpretation, and Refinements». En: *WIREs Computational Statistics* 11.3. _eprint: <https://onlinelibrary.wiley.com/doi/pdf/10.1002/wics.1460>, e1460.
- Efron, B. (ene. de 1979). «Bootstrap Methods: Another Look at the Jackknife». En: *The Annals of Statistics* 7.1, págs. 1-26.
- Hall, Peter (dic. de 1987). «On Kullback-Leibler Loss and Density Estimation». En: *The Annals of Statistics* 15.4, págs. 1491-1519.
- Hastie, Trevor, Robert Tibshirani y Jerome Friedman (2009). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. New York: Springer.
- Hoffman, Matthew D. y Andrew Gelman (2014). «The No-U-Turn Sampler: Adaptively Setting Path Lengths in Hamiltonian Monte Carlo». En: *Journal of Machine Learning Research* 15.47, págs. 1593-1623.
- Husson, Francois, Sébastien Le y Jérôme Pagès (abr. de 2017). *Exploratory Multivariate Analysis by Example Using R*. en. CRC Press.
- James, Gareth y col. (2013). *An Introduction to Statistical Learning*. Vol. 103. New York, NY: Springer New York.
- Kruschke, John K. (2014). «Doing Bayesian Data Analysis: A Tutorial with R, JAGS, and Stan, Second Edition». En: *Doing Bayesian Data Analysis: A Tutorial with R, JAGS, and Stan, Second Edition*, págs. 1-759.
- Quenouille, M. H. (ene. de 1949). «Approximate Tests of Correlation in Time-Series». En: *Journal of the Royal Statistical Society: Series B (Methodological)* 11.1, págs. 68-84.

- Stone, M. (1977). «An Asymptotic Equivalence of Choice of Model by Cross-Validation and Akaike's Criterion». En: *Journal of the Royal Statistical Society. Series B (Methodological)* 39.1, págs. 44-47.
- Wasserman, Larry (2006). *All of Nonparametric Statistics*. New York, NY: Springer New York.