

Notas Curso de Estadística II

Maikol Solís Chacón y Luis Barboza Chinchilla

Actualizado el 27 junio, 2022

Índice general

1. Introducción	9
2. Estimación no-paramétrica de densidades	11
2.1. Histograma	11
2.1.1. Construcción Estadística	11
2.1.2. Construcción probabilística	13
2.1.3. Propiedades estadísticas	13
2.1.4. Propiedades estadísticas	13
2.1.5. Sesgo	14
2.1.6. Varianza	15
2.1.7. Error cuadrático medio	15
2.1.8. Error cuadrático medio integrado	17
2.1.9. Ancho de banda óptimo para el histograma	18
2.2. Estimación de densidades basada en kernels.	22
2.2.1. Primera construcción	22
2.2.2. Otra construcción	23
2.2.3. Propiedades Estadísticas	27
2.2.4. Sesgo	28
2.2.5. Error cuadrático medio y Error cuadrático medio integrado	30
2.2.6. Ancho de banda óptimo	31
2.2.6.1. Referencia normal	32
2.2.6.2. Validación Cruzada	34
2.2.7. Intervalos de confianza para estimadores de densidad no paramétricos	36
2.3. Laboratorio	37
2.3.1. Efecto de distintos Kernels en la estimación	38

2.3.2. Efecto del ancho de banda en la estimación	41
2.3.3. Ancho de banda óptimo	47
2.3.4. Validación cruzada	51
2.3.5. Temas adicionales	53
2.4. Ejercicios	60
3. Jackknife y Bootstrap	61
3.1. Caso concreto	61
3.2. Jackknife	62
3.3. Bootstrap	67
3.3.1. Intervalos de confianza	72
3.3.1.1. Intervalo Normal	72
3.3.1.2. Intervalo pivotal	72
3.3.1.3. Intervalo pivotal studentizado	74
3.3.2. Resumiendo	76
3.4. Ejercicios	76
4. Métodos lineales de regresión	79
4.1. Introducción al Aprendizaje Estadístico.	79
4.1.1. Formas de estimar f	80
4.1.2. Medidas de bondad de ajuste	81
4.2. Regresión lineal	82
4.2.1. Forma matricial	82
4.2.2. Laboratorio	85
4.3. Propiedades estadísticas	92
4.3.1. Prueba t	94
4.3.2. Prueba F	94
4.3.3. Laboratorio	96
4.4. Medida de bondad de ajuste	99
4.4.1. Laboratorio	100
4.4.1.1. R^2	101
4.4.1.2. R^2 ajustado	101
4.4.1.3. <code>summary</code>	101
4.5. Predicción	102
4.5.1. Laboratorio	103
4.5.1.1. Ajuste de la regresión sin intervalos de confianza	104
4.5.1.2. Ajuste de la regresión con intervalos de confianza	105

4.5.1.3. Ajuste de la regresión con intervalos de confianza y predicción	106
4.6. Interacciones	110
4.6.1. Laboratorio	111
4.7. Supuestos	115
4.7.1. Chequeos básicos de las hipótesis de regresión lineal	116
4.7.1.1. Linealidad, Errores con esperanza nula, Homocedasticidad	116
4.7.1.2. Independencia de los errores	121
4.7.1.3. Normalidad de los errores	125
4.7.1.4. Multicolinealidad	129
4.7.2. Otros chequeos importantes	132
4.7.2.1. Puntos extremos	132
4.7.2.2. Puntos de apalancamiento (leverage)	135
Distancia de Cook.	135
4.8. Ejercicios	154
5. Regresión Logística	155
5.1. Preliminares	155
5.1.1. Oportunidad relativa (Odds Ratio)	159
5.2. Máxima verosimilitud	159
5.2.1. Resultados adicionales	161
5.3. Diagnósticos del modelo	161
5.3.1. Supuesto de linealidad	161
5.3.2. Valores de gran influencia	163
5.3.3. Multicolinealidad	164
5.4. Predicción y poder de clasificación	164
5.4.1. Curva ROC	168
5.5. Ejercicios	172
6. Métodos de selección de variables y regularización	173
6.1. Estimación del error de prueba	173
6.1.1. Técnica de conjunto de validación	173
6.1.2. Validación cruzada “Leave-One-Out” (LOOCV)	174
6.1.3. Validación cruzada k -veces	175
6.1.4. Validación cruzada para clasificación	175
6.1.5. Otras medidas de error de prueba	176
6.1.5.1. R^2 ajustado	176

6.1.5.2.	C_p de Mallows	176
6.1.5.3.	Estimador de máxima verosimilitud (MLE) . .	176
6.1.5.4.	Akaike Information Criterion (AIC).	176
6.1.5.5.	C_p de Mallows	177
6.1.5.6.	Estimador de máxima verosimilitud (MLE) . .	178
6.1.5.7.	Akaike Information Criterion (AIC).	179
6.1.5.8.	Bayesian Information Criterion (BIC)	179
6.1.5.9.	Notas adicionales	180
6.2.	Selección de variables	181
6.2.1.	Selección del mejor subconjunto.	181
6.2.2.	Selección de modelos hacia adelante (Forward Stepwise Selection)	183
6.2.3.	Selección de modelos hacia atrás (Backward Stepwise Selection)	183
6.3.	Métodos de regularización	184
6.3.1.	Regresión Ridge	184
6.3.2.	Regresión Lasso	185
6.3.3.	Explicación gráfica	186
6.4.	Laboratorio	187
6.4.1.	Cross-Validation	187
6.4.1.1.	Leave-one-out Cross Validation (LOOCV) . .	187
6.4.1.2.	K-Fold Cross Validation	189
6.4.2.	Selección de variables	190
6.4.2.1.	Análisis exploratorio	190
6.4.2.2.	Regresión forward y backward	200
6.4.2.3.	Regresión Ridge	207
6.4.3.	Regresión Lasso	216
6.5.	Ejercicios	219
7.	Otros Clasificadores	221
7.1.	Clasificador Bayesiano	221
7.2.	Método de k vecinos más cercanos (KNN) . .	222
7.3.	Análisis Discriminante	223
7.3.1.	Análisis discriminante lineal	225
7.3.1.1.	Caso p=1	225
7.3.1.2.	Caso p>1	225
7.3.2.	Análisis discriminante cuadrático	226
7.4.	Laboratorio	227

7.4.1. Clasificador logístico	231
7.4.2. Análisis Discriminante Lineal	234
7.4.3. Análisis Discriminante Cuadrático	236
7.4.4. K vecinos más cercanos	237
7.5. Ejercicios	252
8. Análisis en componentes principales	253
8.1. Representación gráfica	253
8.2. Aprendizaje no-supervisado	254
8.3. Primer componente principal	255
8.4. Segunda componente principal	257
8.5. Círculo de correlaciones	258
8.6. Volvamos a nuestro ejemplo	258
8.7. ¿Cuántos componentes usar?	260
8.8. Laboratorio	262
8.9. Ejercicios	265
9. Cálculo Bayesiano Computacional	267
9.1. Repaso de Estadística Bayesiana	267
9.1.1. Modelo de un parámetro	267
9.1.2. Modelo de más de un parámetro	272
9.2. Motivación: Cálculo de Integrales	275
9.3. Ejemplo base: modelo beta-binomial.	275
9.4. Aproximación de Laplace	279
9.5. Simulación	283
9.5.1. Simulación Monte Carlo	283
9.5.2. Muestreo por rechazo	284
9.6. Muestreo por importancia	287
9.7. Remuestreo por importancia	290
9.8. Métodos Monte Carlo	291
9.8.1. Ejemplo del viajero con una moneda	291
9.8.2. Cadenas de Markov	294
9.8.3. El algoritmo de Metropolis-Hastings	295
9.8.4. ¿Por qué el algoritmo de Metropolis Hastings funciona? .	296
9.8.5. Extensión al caso del viajero	297
9.9. El problema del viajero con dos monedas	302
9.9.1. Muestreo de Gibbs	310
9.10. Uso de JAGS	319

9.11. Uso de STAN	328
-----------------------------	-----

Capítulo 1

Introducción

Estas son las notas de clase del curso CA0403: Estadística Actuarial II para el primer semestre del 2022.

Capítulo 2

Estimación no-paramétrica de densidades

2.1. Histograma

El histograma es una de las estructuras básicas en estadística y es una herramienta descriptiva que permite visualizar la distribución de los datos sin tener conocimiento previo de los mismos. En esta sección definiremos el histograma más como un estadístico que como una herramienta de visualización de datos.

2.1.1. Construcción Estadística

Suponga que X_1, X_2, \dots, X_n es una muestra independiente que proviene de una distribución desconocida f . En este caso no asumiremos que f tenga alguna forma particular, que permita definirla de manera paramétrica como en el curso anterior.

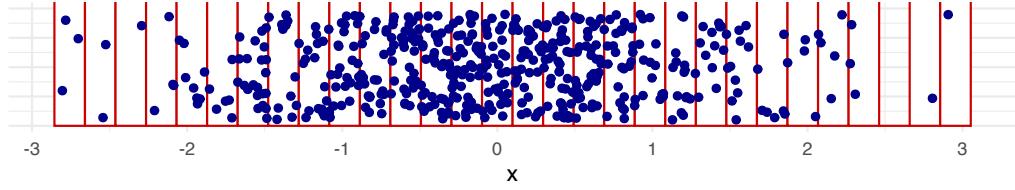
Construcción:

- 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 el segmento B_j . Denótelo como n_j .

12 CAPÍTULO 2. ESTIMACIÓN NO-PARAMÉTRICA DE DENSIDADES



- Divida el número de observaciones en B_j por el tamaño de muestra n y el ancho de banda h de cada caja.

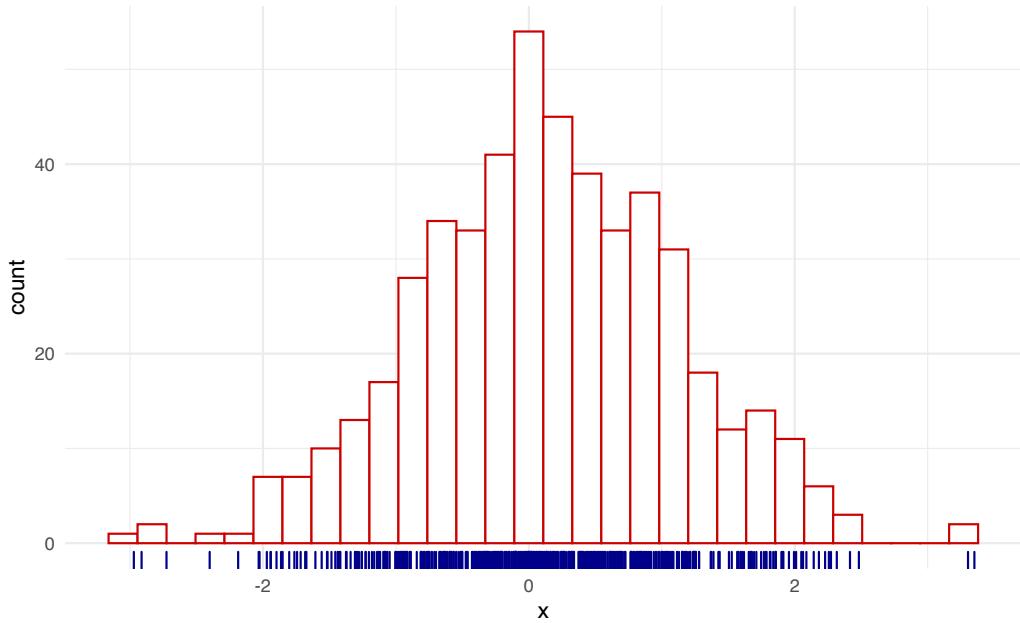
$$f_j = \frac{n_j}{nh}$$

De esta forma si se suma las áreas definidas por el histograma da un total de 1.

- 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 probabilística

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

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

Otra forma de aproximarla es:

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

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

Note que el estimador de histograma \hat{f}_h tiende a ser más suave conforme aumenta el ancho de banda h .

2.1.4. Propiedades estadísticas

Suponga que $x_0 = 0$ y que $x \in B_j$ es un punto fijo, entonces el estimador evaluado en x es:

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

2.1.5. Sesgo

Para calcular el sesgo primero calculamos:

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

donde $I(X_i \in B_j)$ es una variable Bernoulli con valor esperado:

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

Entonces,

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

y por lo tanto el sesgo de $\hat{f}_h(x)$ es:

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

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

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

Entonces se puede concluir que:

- $\hat{f}_h(x)$ es un estimador sesgado de $f(x)$.
- El sesgo tiende a ser cero cerca del punto medio de B_j .
- El sesgo es creciente con respecto a la pendiente de la verdadera densidad evaluada en el punto medio m_j .

2.1.6. Varianza

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

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

La variable I es una bernoulli con parametro $\int_{(j-1)h}^h f(u)du$ 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:

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

Consecuencias:

- La varianza del estimador es proporcional a $f(x)$.
- La varianza decrece si el ancho de banda h crece.

2.1.7. Error cuadrático medio

El error cuadrático medio del histograma es el

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

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

Retomando los términos anteriores se puede comprobar que:

$$\text{MSE}(\hat{f}_h(x)) = \frac{1}{nh} f(x) + f' \left\{ \left(j - \frac{1}{2} \right) h \right\}^2 \left\{ \left(j - \frac{1}{2} \right) h - x \right\}^2 \quad (2.1)$$

$$+ o(h) + o\left(\frac{1}{nh}\right) \quad (2.2)$$

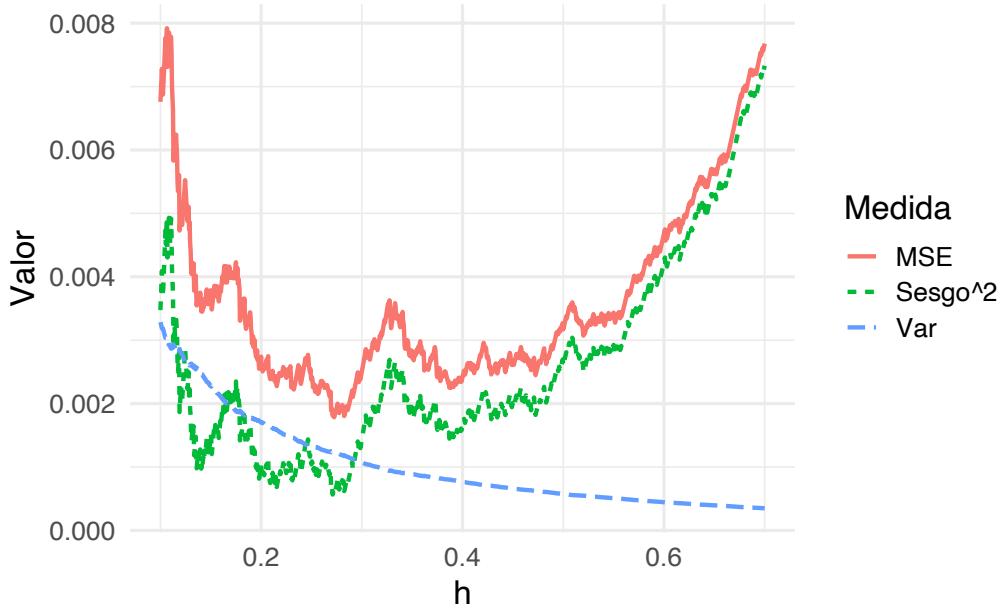
Nota. Si $h \rightarrow 0$ y $nh \rightarrow \infty$ entonces $\text{MSE}(\hat{f}_h(x)) \rightarrow 0$. Es decir, conforme usamos más observaciones, pero el ancho de banda de banda no decrece tan rápido, entonces el error cuadrático medio converge a 0.

Como $\text{MSE}(\hat{f}_h(x)) \rightarrow 0$ (convergencia en \mathbb{L}^2) implica que $\hat{f}_h(x) \xrightarrow{\mathcal{P}} f(x)$, entonces \hat{f}_h es consistente. Además según la fórmula (2.2), concluimos lo siguiente:

- 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 ∞)

Ejercicio 2.3. Si $f \sim N(0, 1)$, aproxime los componentes de sesgo, varianza y MSE, y grafíquelos para distintos valores de h .

Solución:



2.1.8. Error cuadrático medio integrado

Uno de los problemas con el MSE $(\hat{f}_h(x))$ es que depende de x y de la función de densidad f (desconocida). Integrando con respecto a x el MSE se logra resolver el primer problema:

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

Al MISE se le llama error cuadrático medio integrado. Además,

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

la cual es una buena aproximación si $h \rightarrow 0$. A este último término se le llama MISE asintótico.

2.1.9. Ancho de banda óptimo para el histograma

El MISE tiene un comportamiento asintótico similar al observado en el MSE. La figura siguiente presenta el comportamiento de la varianza, sesgo y MISE para nuestro ejemplo anterior:

Un problema frecuente en los histogramas es que la mala elección del parámetro h causa que estos no capturen toda la estructura de los datos. Por ejemplo, en el siguiente caso se muestra histogramas construidos a partir de 1000 números aleatorios según una $N(0, 1)$, bajo 4 distintas escogencias de ancho de banda.

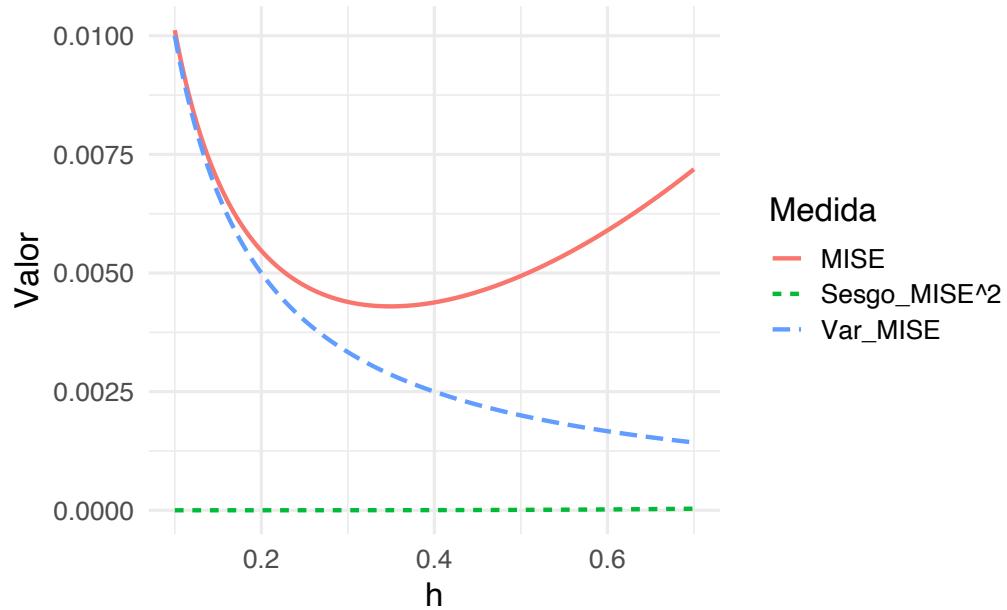
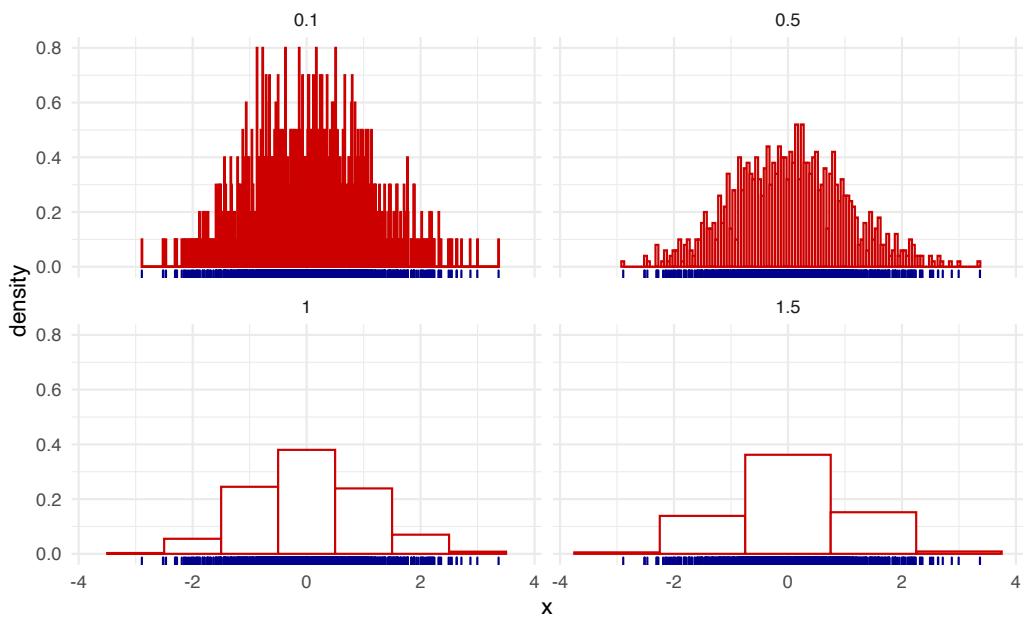


Figura 2.1:



20 CAPÍTULO 2. ESTIMACIÓN NO-PARAMÉTRICA DE DENSIDADES

Un criterio más preciso para seleccionar el ancho de banda es a través de la minimización del MISE:

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

lo implica que

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

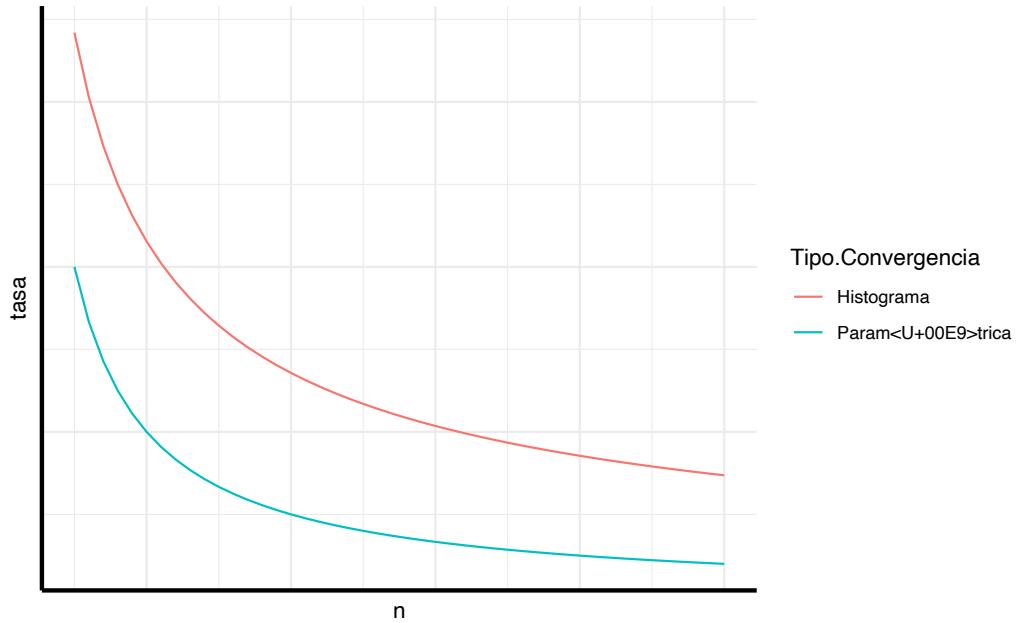
y por lo tanto

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

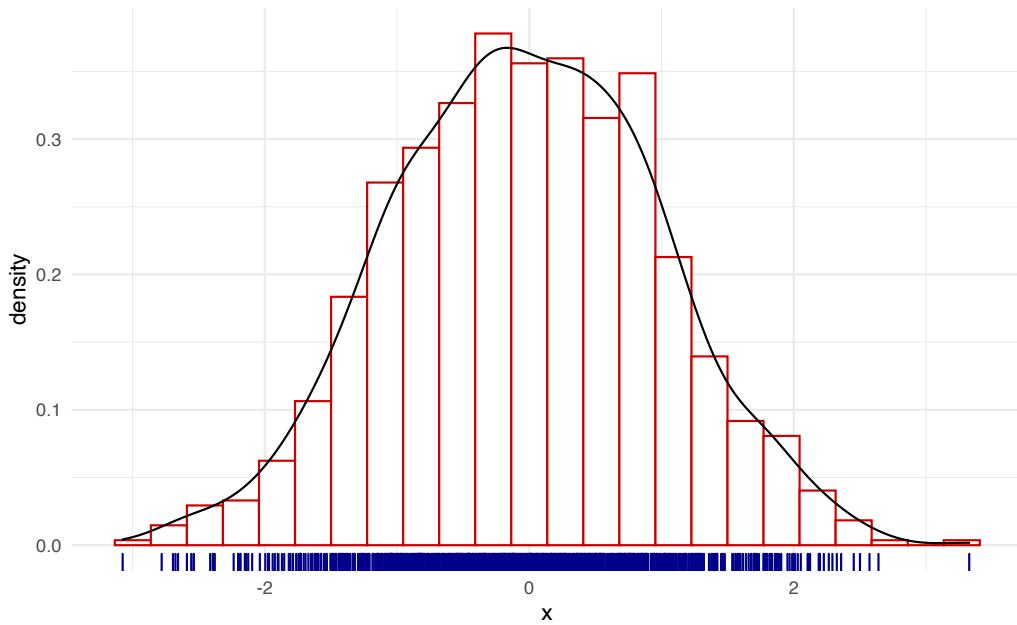
Nota (Recuerde de Estadística I). Si $X_1, \dots, X_n \sim f_\theta$ i.i.d, con $\text{Var}(X) = \sigma^2$ y media θ , recuerde que el estimador $\hat{\theta}$ de θ tiene la característica que

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

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, tal y como se ilustra en el siguiente gráfico:



Finalmente, podemos encontrar el valor óptimo del ancho de banda ($h = 0.2725$) del conjunto de datos en el ejemplo anterior.



Ejercicio 2.4. Verifique que en el caso normal estándar: $h_{opt} \approx 3,5n^{-1/3}$.

2.2. Estimación de densidades basada en kernels.

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

La distribución empírica de F es:

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

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

Nota. ¿Podríamos derivar F_n para encontrar el estimador \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 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 cual es equivalente al caso del histograma.

2.2.2. Otra construcción

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

Nota. ¿Qué pasaría si cambiamos la palabra **contenidos** por **alrededor de** “ x ”?

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

El estimador de histograma se escribe como

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

Note que si definimos

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

con $u = \frac{x - x_i}{h}$, entonces parte del estimador de histograma se puede escribir como:

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

Finalmente se tendría 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? Esto permitiría incluir la noción de “cercanía” de cada dato alrededor de x .

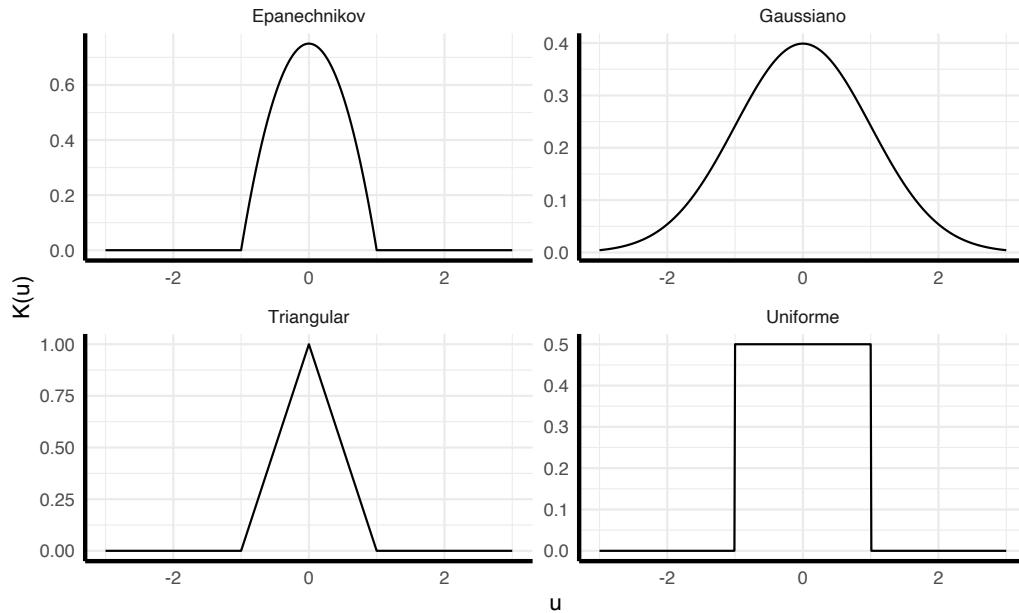
Esta función debería cumplir las siguientes características:

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

24 CAPÍTULO 2. ESTIMACIÓN NO-PARAMÉTRICA DE DENSIDADES

Por ejemplo:

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



Entonces se tendría que la expresión general para un estimador por núcleos (kernel) de f :

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

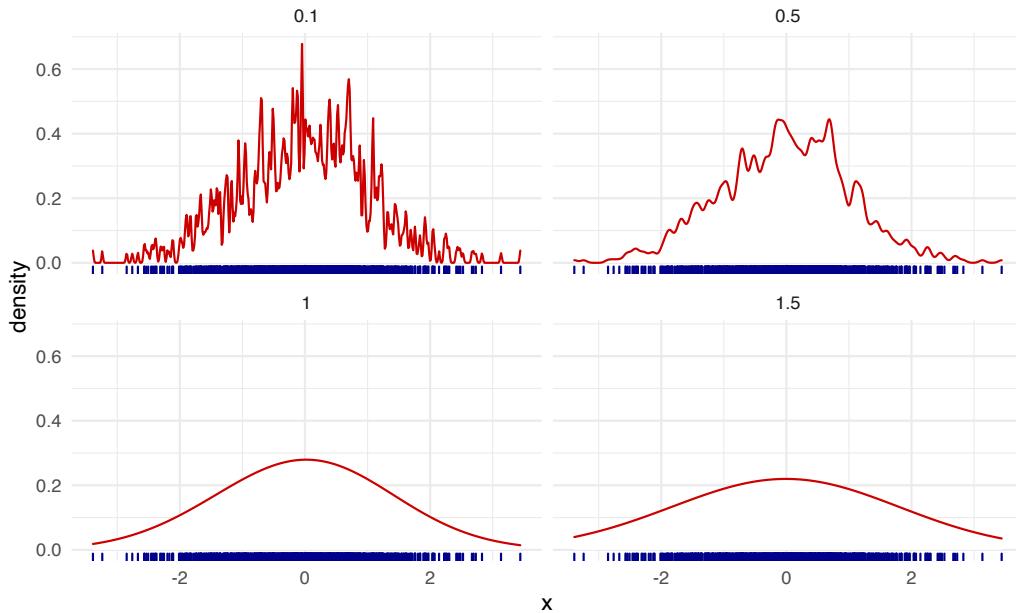
donde x_1, \dots, x_n es una muestra i.i.d. de f ,

$$K_h(\cdot) = \frac{1}{h} K(\cdot/h).$$

y K es un kernel según las 4 propiedades anteriores.

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

Nuevamente controlaríamos la suavidad del estimador a como se ilustra a continuación:

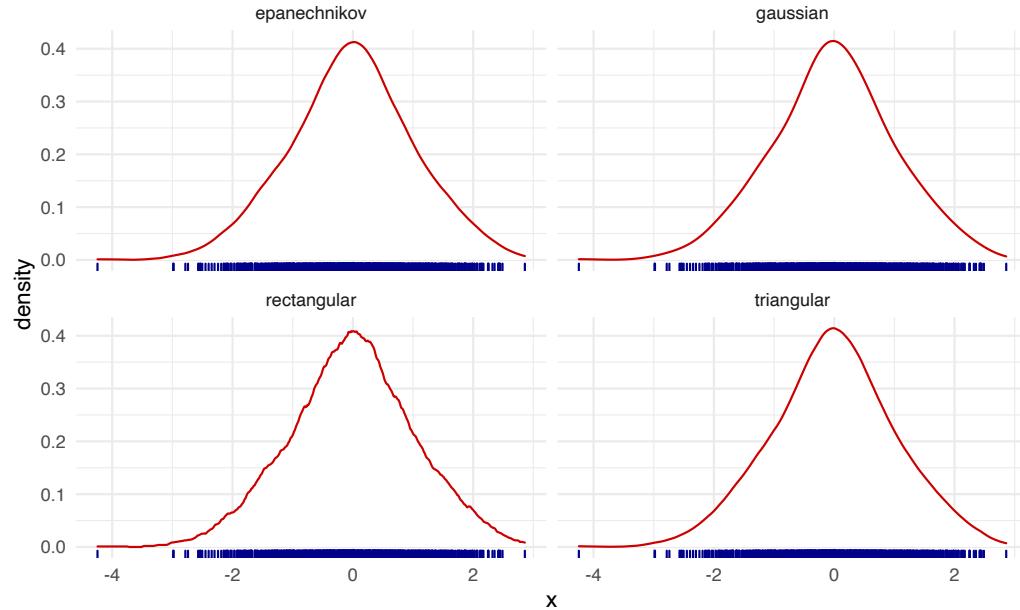


Inconveniente: no tenemos aún un criterio para un h óptimo.

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

Usando 1000 números aleatorios según una normal estándar, con un ancho de banda fijo ($h = 0,3$) podemos ver que no hay diferencias muy marcadas entre los estimadores por kernel:

26 CAPÍTULO 2. ESTIMACIÓN NO-PARAMÉTRICA DE DENSIDADES



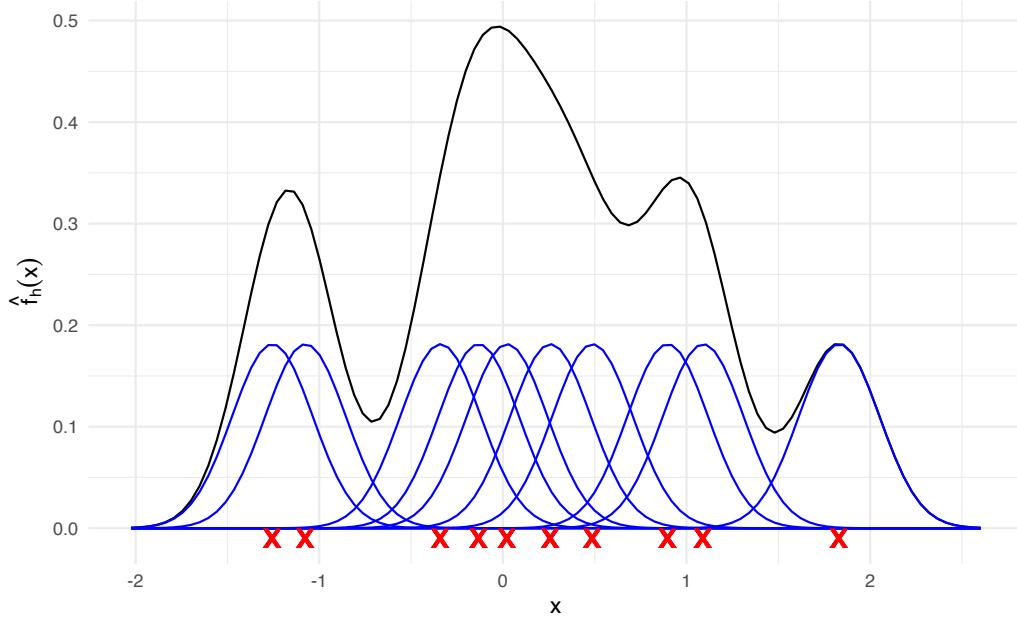
Recordemos nuevamente la fórmula

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

Cada sumando de esta expresión es una función de la variable x . Si la integramos se obtiene que

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

En el siguiente gráfico se generan 10 puntos aleatorios según una normal estándar (rojo) y se grafica cada uno de los 10 componentes del estimador de la densidad usando kernels gaussianos (azul). El estimador resultante aparece en color negro. Note que cada uno de los 10 componentes tiene la misma área bajo la curva, la cual en este caso es 0.1.



2.2.3. Propiedades Estadísticas

Al igual que en el caso de histograma, también aplica lo siguiente:

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

donde

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

En el caso de la 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

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

2.2.4. Sesgo

Para el sesgo tenemos

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

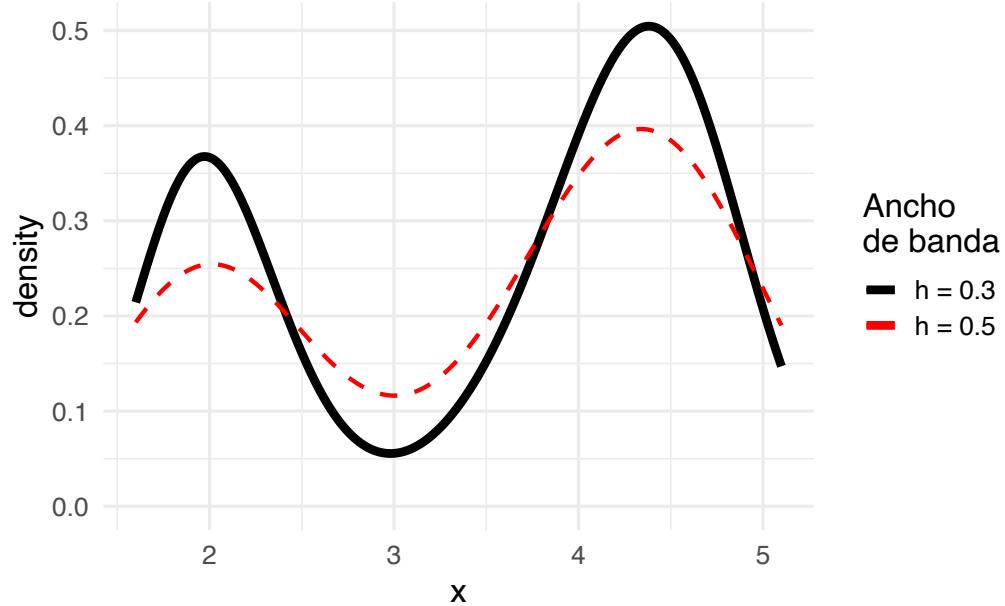
Ejercicio 2.5. Usando el cambio de variable $s = \frac{u-x}{h}$ y las propiedades del kernel pruebe que

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

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 $\int (f'''(x))^2 dx < \infty$.

En el siguiente gráfico se ilustra el estimador no paramétrico de la distribución de tiempos entre erupciones en la muy conocida tabla de datos *faithful*. El estimador se calcula bajo dos distintas escogencias de ancho de banda.



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

2.2.5. 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,

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

Al igual que en el caso del histograma, el estimador por kernels es un estimador consistente de f si $h \rightarrow 0$ y $nh \rightarrow \infty$. Además el MISE depende directamente de f'' .

2.2.6. Ancho de banda óptimo

Minimizando el MISE con respecto a h obtenemos

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

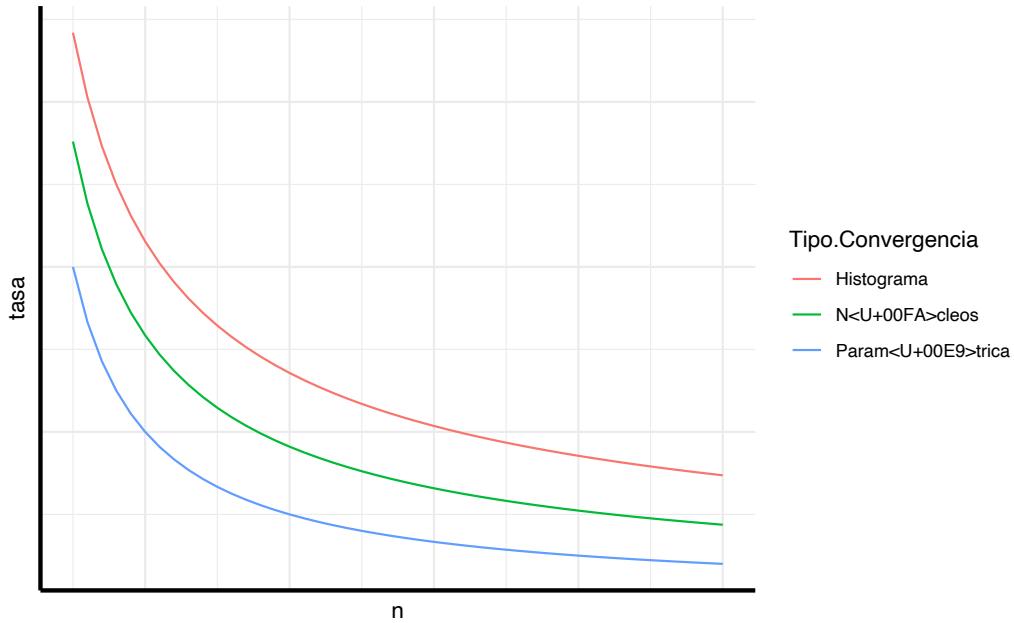
Nota. De forma práctica, h_{opt} no es un estimador útil de h porque depende de $\|f''\|_2^2$ que es desconocido. Más adelante veremos otra forma de encontrar este estimador.

Evaluando h_{opt} en el MISE tenemos que

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

y por lo tanto la tasa de convergencia del MISE a 0 es más rápida que para el caso del histograma:

32 CAPÍTULO 2. ESTIMACIÓN NO-PARAMÉTRICA DE DENSIDADES



Nota. Como se comentó anteriormente, el principal inconveniente del ancho de banda:

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

es que depende de f'' .

A continuación se explica dos posibles métodos para determinar para aproximar el ancho de banda óptimo:

2.2.6.1. 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 2.6. Pruebe que la ecuación anterior es verdadera. Utilice el hecho de que:

$$\|f''\|_2^2 = \sigma^{-5} \int \phi''(x)^2 dx$$

donde ϕ es la función de densidad de una $N(0, 1)$.

Nota. El principal inconveniente de \hat{h}_{rn} 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.

Para solucionar el problema anterior, se puede considerar una medida más robusta de variación, por ejemplo el rango intercuantil IQR:

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

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

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

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

pero esta aproximación es conveniente bajo el escenario de que la densidad f sea similar a una densidad normal.

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

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

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

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

Nota. El problema con esta expresión es que las observaciones que se usan para estimar la esperanza son las mismas 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 una estimación de $\int \hat{f}_h(x)f(x)dx$ a través de:

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

donde (estimador *leave-one-out*)

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

de esta forma nos aseguramos que las observaciones que se usan para calcular $\hat{f}_{h,-i}(x)$ son independientes de la observación que uno usa para definir el estimador de $E[\hat{f}_h(x)]$.

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

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

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

Finalmente tenemos la función,

Finalmente definimos la función objetivo del criterio de validación cruzada como:

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

Nota. Note que $\text{CV}(h)$ no depende de f o sus derivadas y además la función objetivo se adapta automáticamente a las características de la densidad f .

2.2.7. 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

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

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

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

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.

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 está definido para x fijo y no permite hacer inferencia sobre toda la función f . Una forma de determinar la banda de confianza de toda la función f es a través de la fórmula 3.52 en la página 62 de (Härdle y col. 2004).

2.3. Laboratorio

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

2.3.1. Efecto de distintos Kernels en la estimación

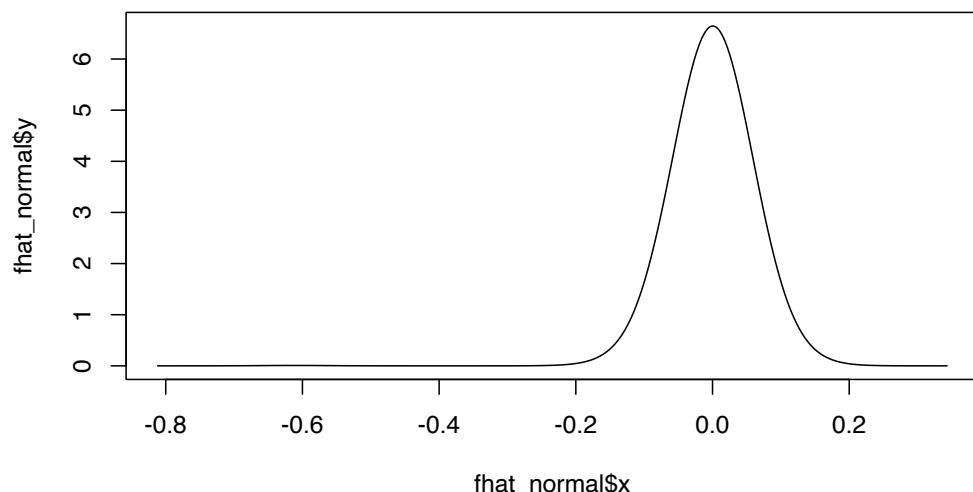
```
x <- read.csv("data/stockres.txt")
x <- unlist(x)

summary(x)

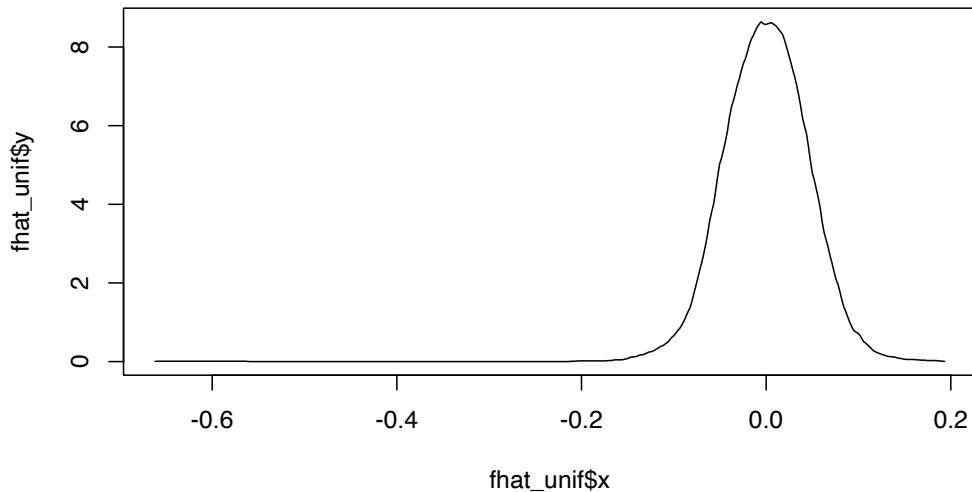
##      Min.   1st Qu.    Median     Mean   3rd Qu.   Max.
## -0.6118200 -0.0204085 -0.0010632 -0.0004988  0.0215999  0.1432286

library(KernSmooth)

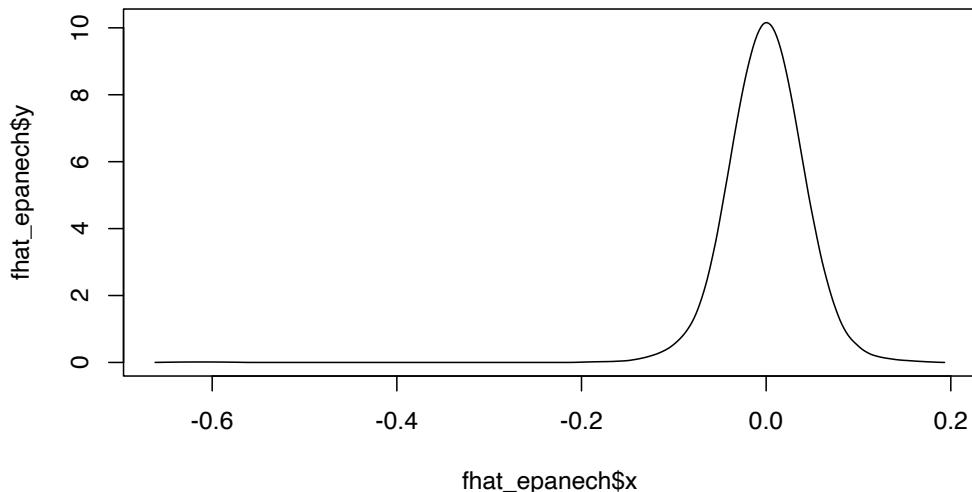
fhat_normal <- bkde(x, kernel = "normal", bandwidth = 0.05)
plot(fhat_normal, type = "l")
```



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

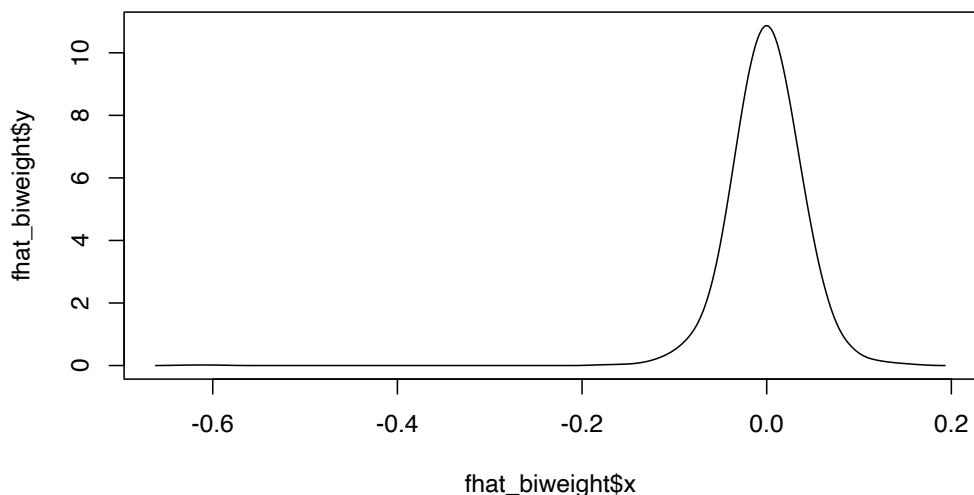


```
fhat_epanech <- bkde(x, kernel = "epanech", bandwidth = 0.05)
plot(fhat_epanech, type = "l")
```

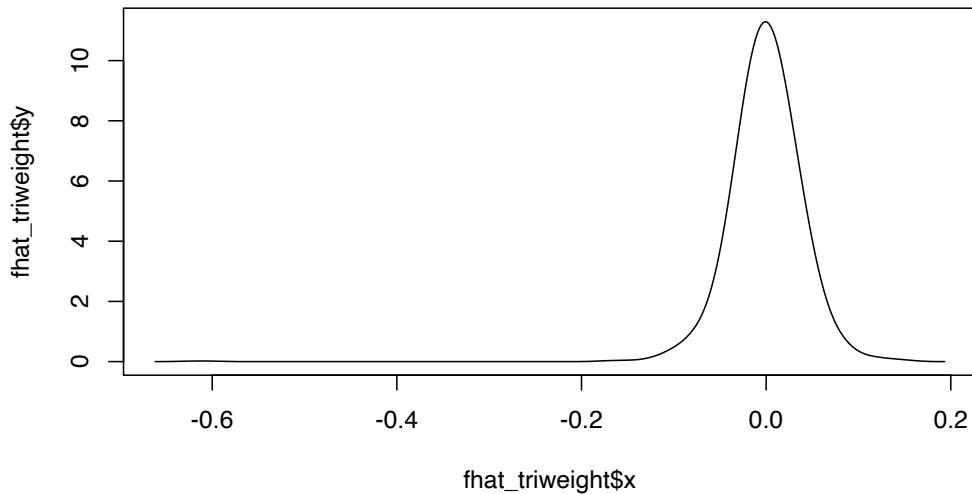


40 CAPÍTULO 2. ESTIMACIÓN NO-PARAMÉTRICA DE DENSIDADES

```
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")
```

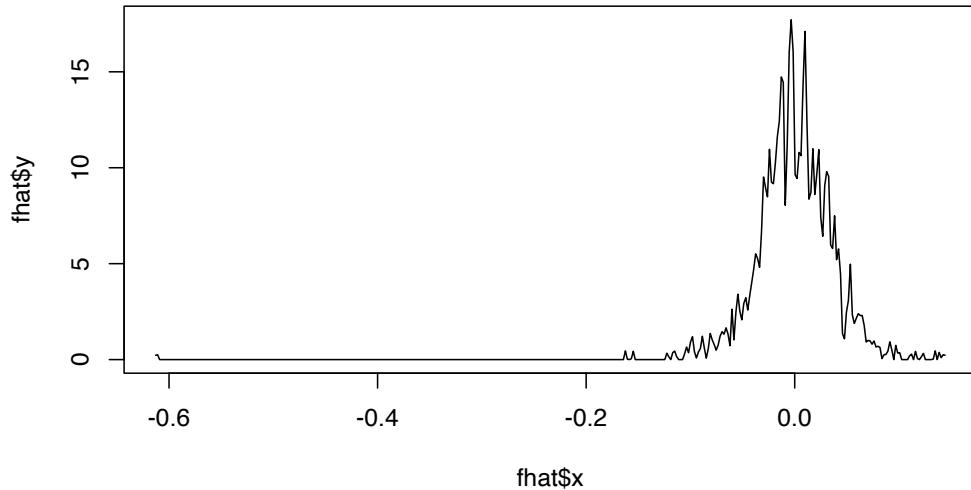


2.3.2. Efecto del ancho de banda en la estimación

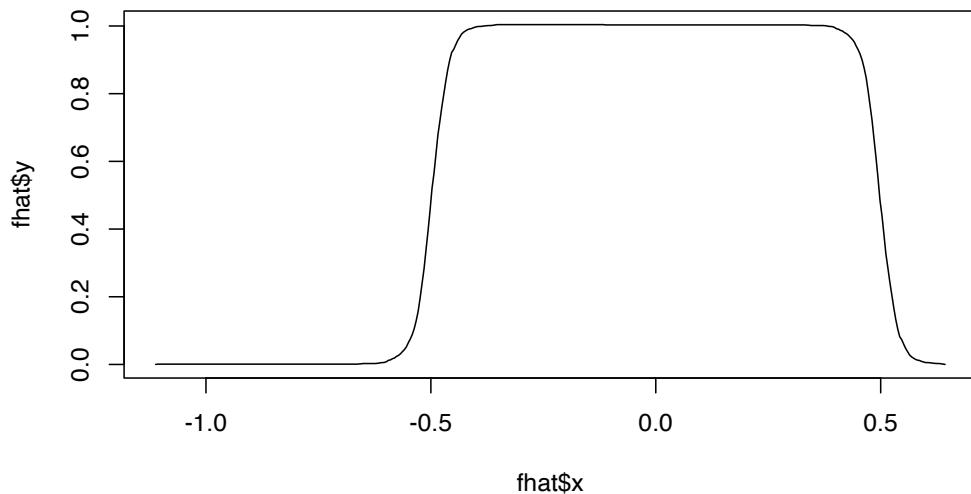
** Kernel uniforme **

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

42 CAPÍTULO 2. ESTIMACIÓN NO-PARAMÉTRICA DE DENSIDADES

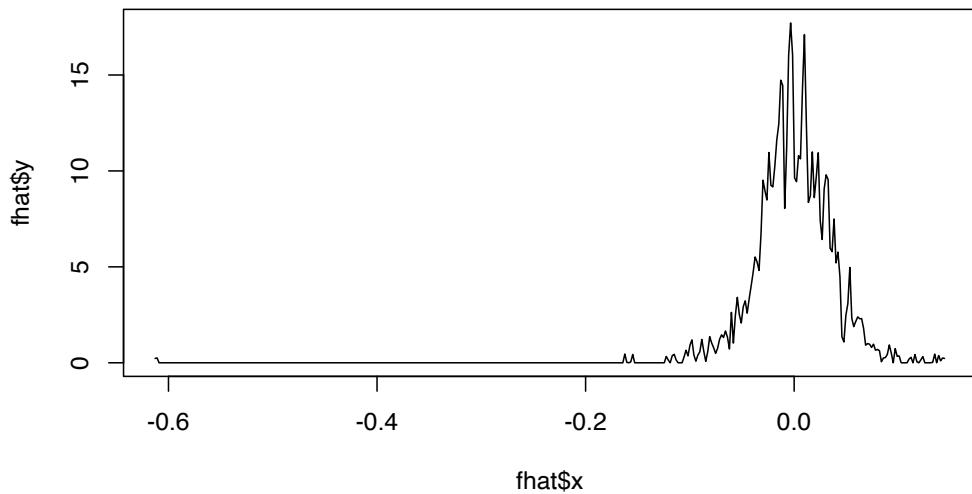


```
fhat <- bkde(x, kernel = "box", bandwidth = 0.5)
plot(fhat, type = "l")
```

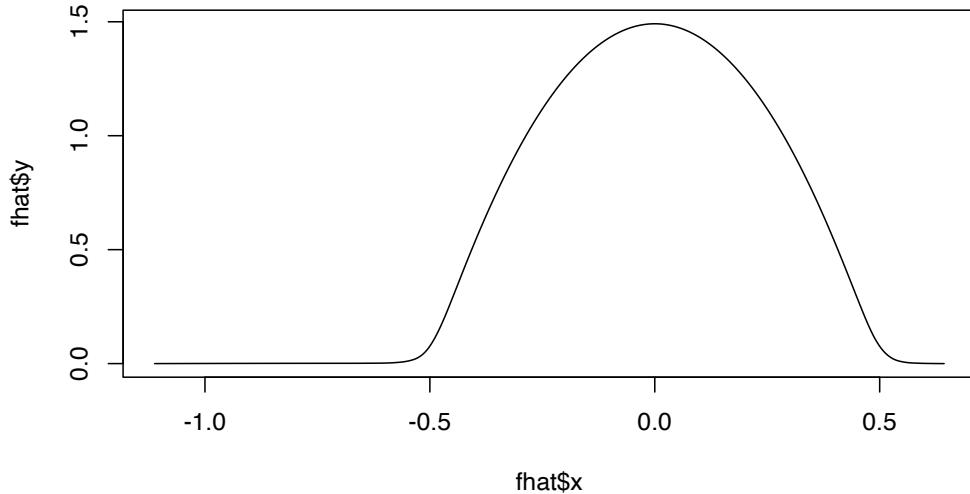


** Kernel Epanechnikov **

```
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)

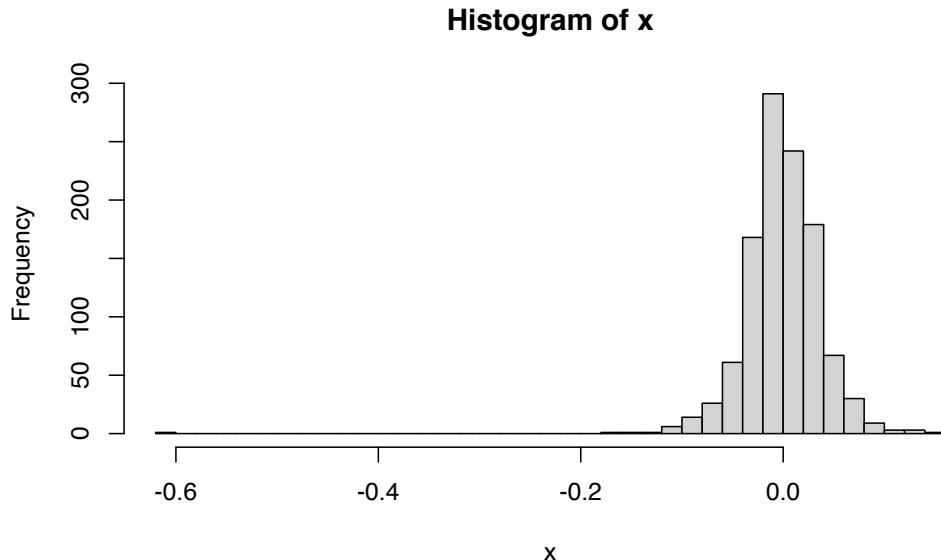
f_param <- dnorm(u, mean = mu, sd = sigma)

h <- 0.02

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

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

46 CAPÍTULO 2. ESTIMACIÓN NO-PARAMÉTRICA DE DENSIDADES

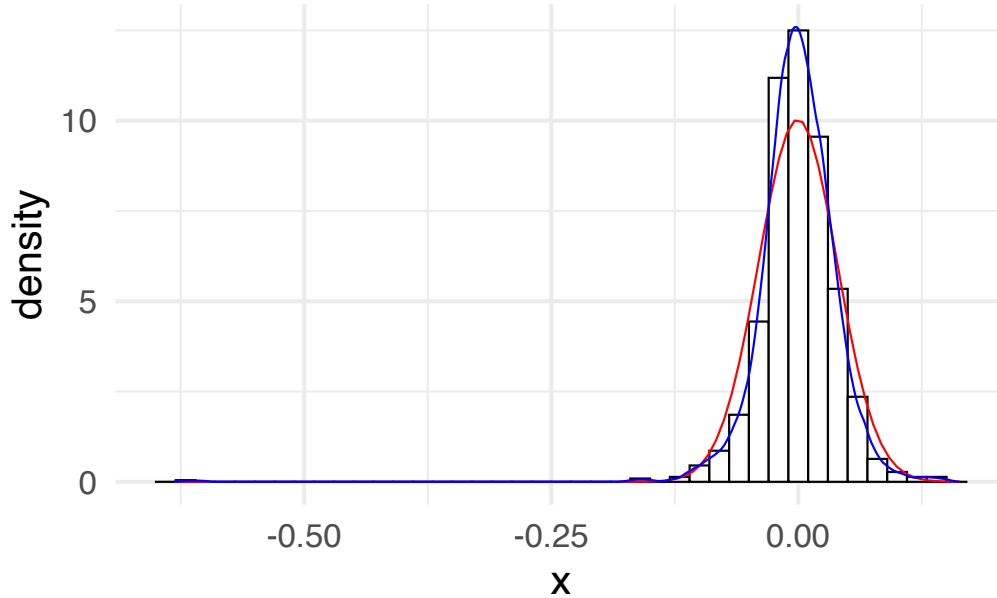


```
f_epa <- as.data.frame(bkde(x, kernel = "epa", bandwidth = h))

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)
```



2.3.3. 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 $IQR/1,34$?

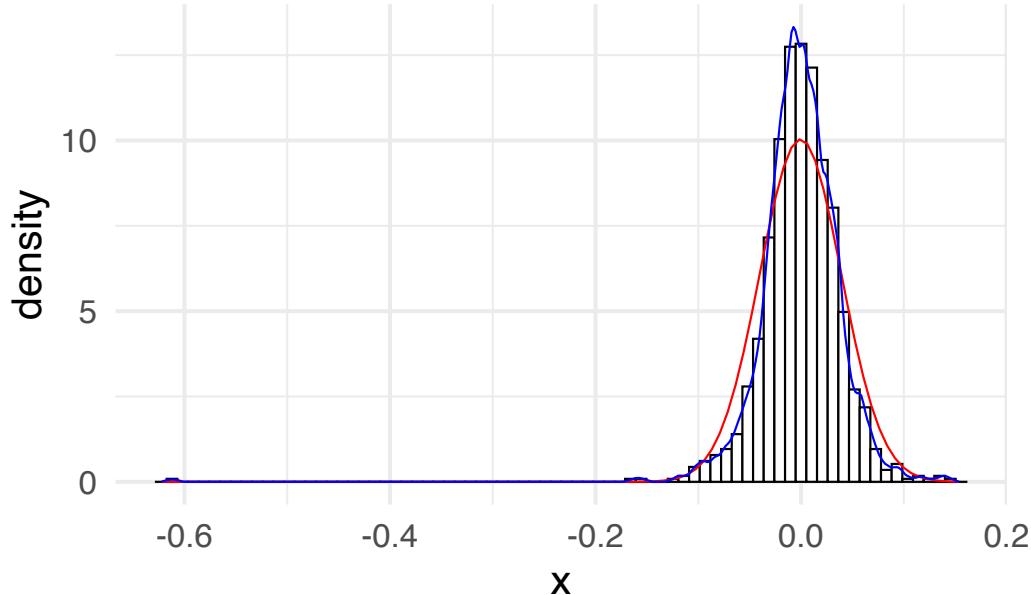
```
s <- sd(x)
n <- length(x)

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)
```



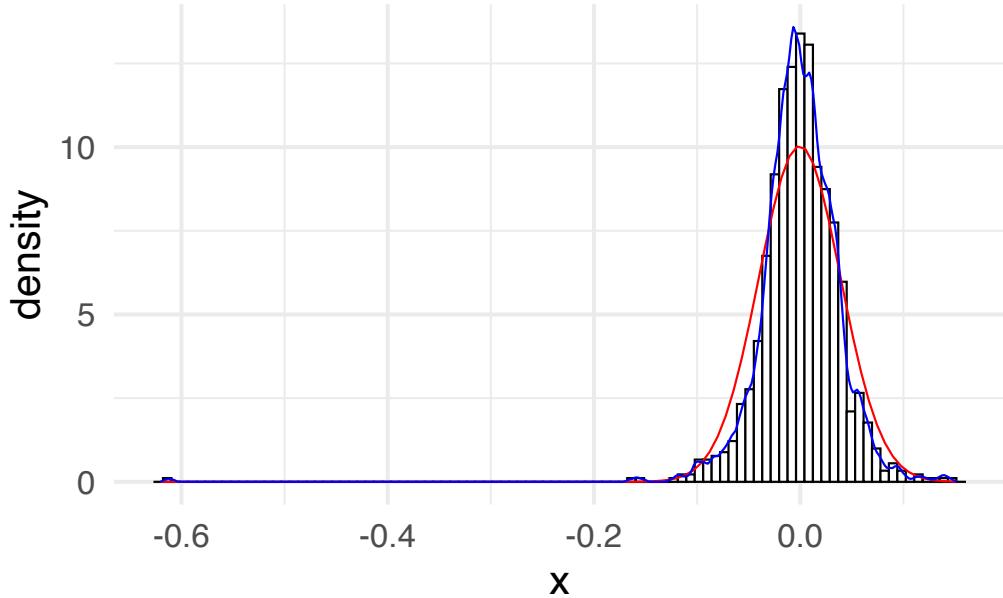
```
h_iqr <- 1.06 * min(s, IQR(x)/1.34) * n^(-1/5)

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)
```

50 CAPÍTULO 2. ESTIMACIÓN NO-PARAMÉTRICA DE DENSIDADES



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

```
library(np)

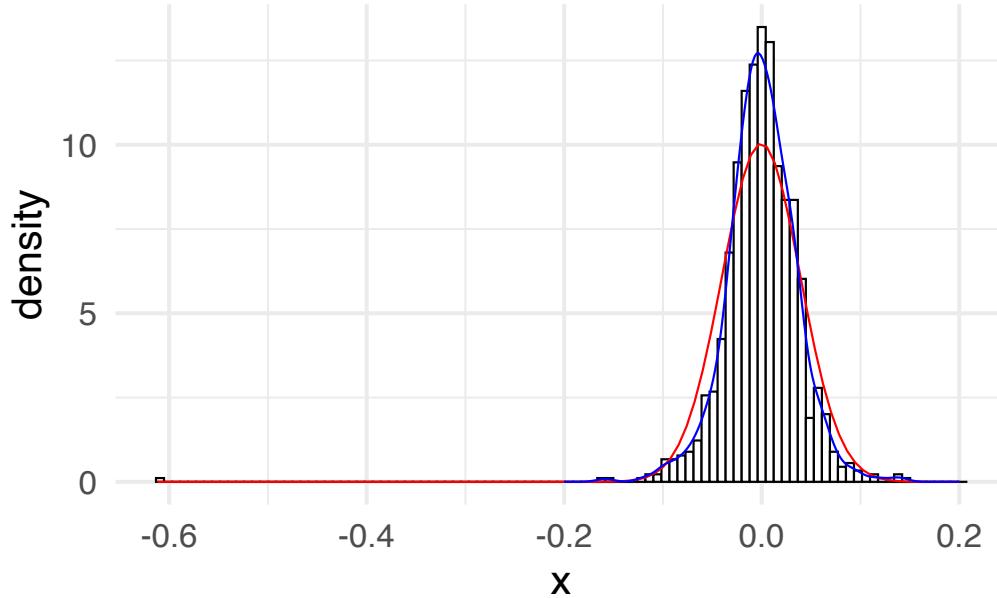
x.eval <- seq(-0.2, 0.2, length.out = 200)

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)
```



2.3.4. Validación cruzada

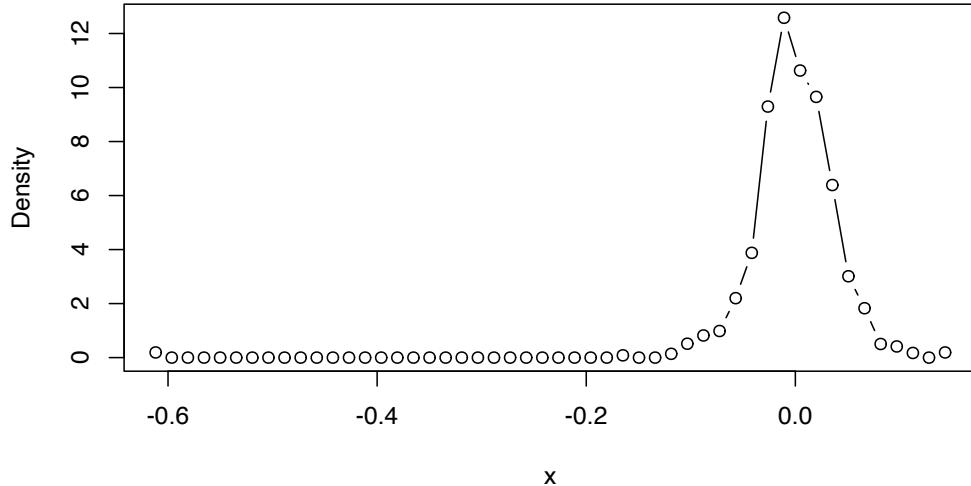
La forma que vimos en clase es la de validación cruzada por mínimos cuadrados “least-square cross validation” la cual se puede ejecutar con este comando.

```
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)

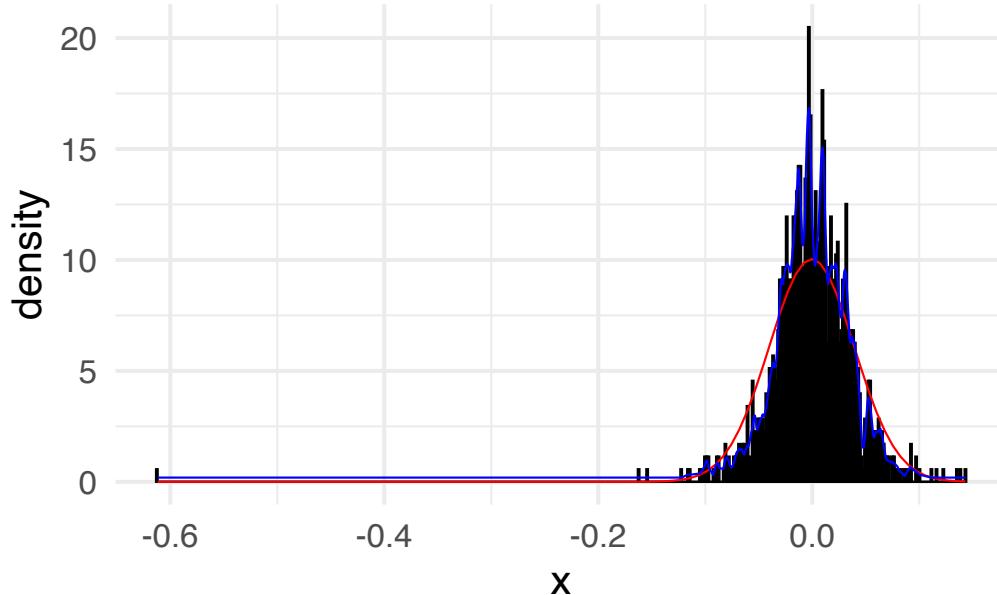
plot(dens.np, type = "b")
```

52 CAPÍTULO 2. ESTIMACIÓN NO-PARAMÉTRICA DE DENSIDADES



```
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)
```



2.3.5. 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.

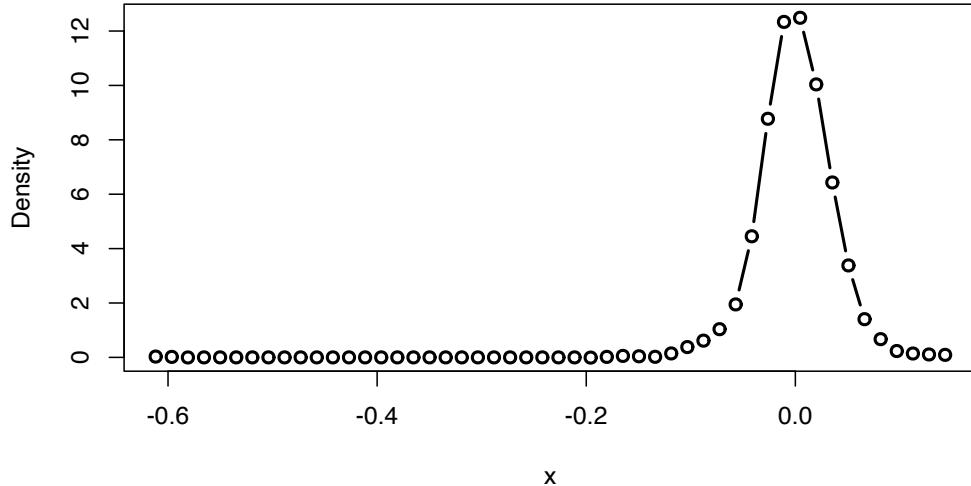
Esto se logra asumiendo que el Kernel es más veces diferenciable.

```
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)

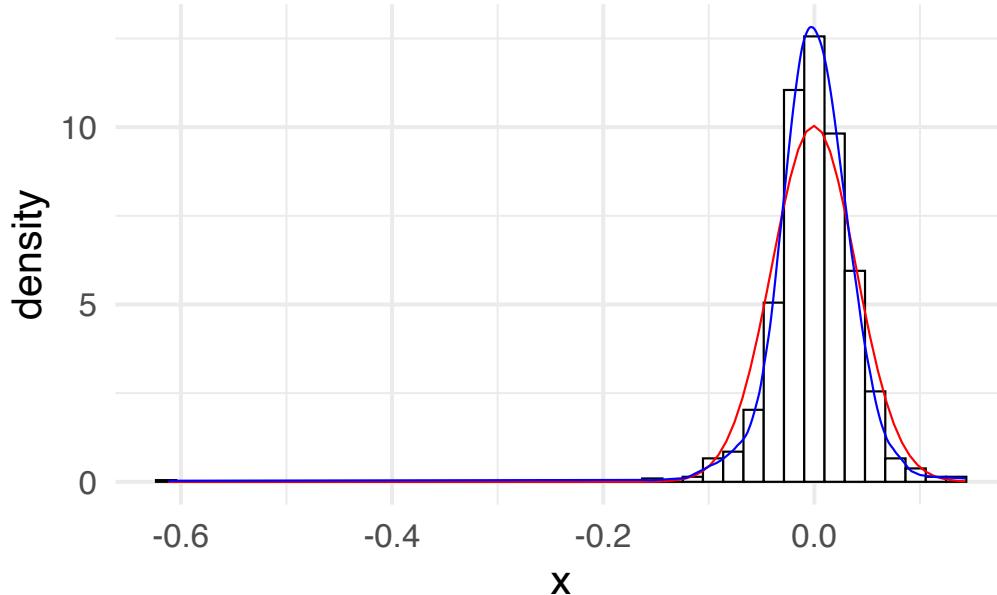
plot(dens.np, type = "b", lwd = 2)
```

54 CAPÍTULO 2. ESTIMACIÓN NO-PARAMÉTRICA DE DENSIDADES



```
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)
```

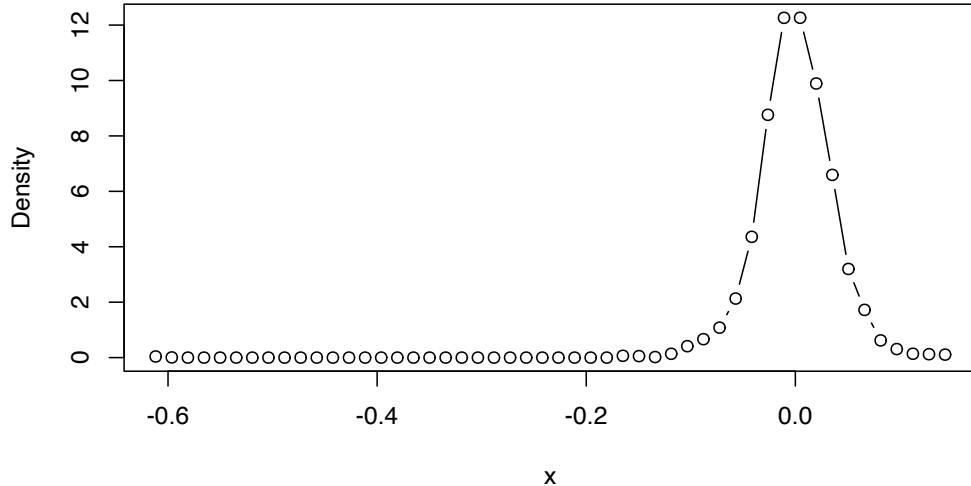


Otra forma de estimar el ancho de banda Otra forma de estimar ancho de bandas óptimos es usando máxima verosimilitud. Les dejo de tarea revisar la sección 1.1 del artículo de (Hall 1987) para entender su estructura.

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

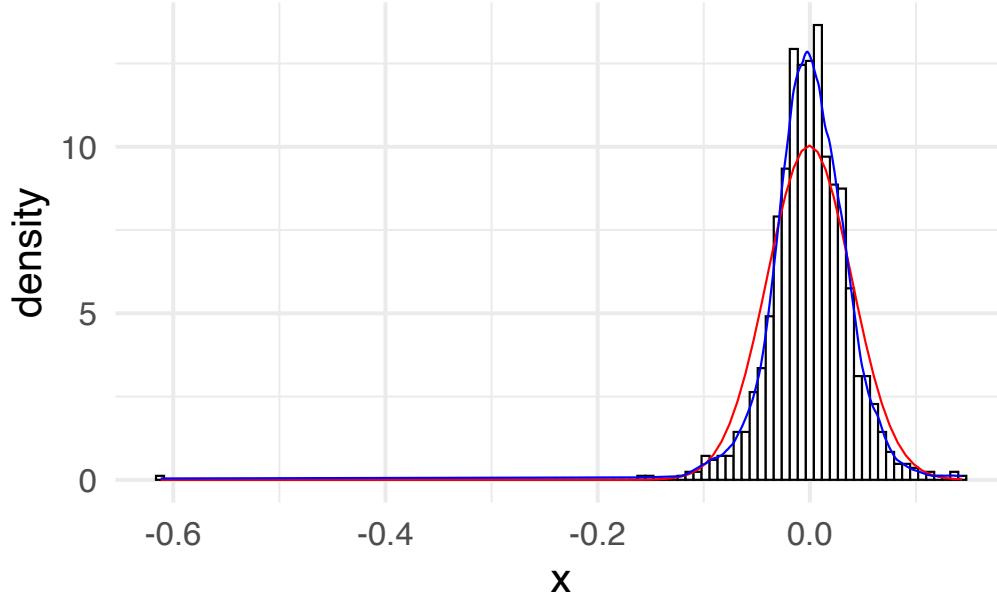
```
## Multistart 1 of 1 |Multistart 1 of 1 |Multistart 1 of 1 |Multistart 1 of 1 /Multis
dens.np <- npudens(h_cv_np_ml)
plot(dens.np, type = "b")
```

56 CAPÍTULO 2. ESTIMACIÓN NO-PARAMÉTRICA DE DENSIDADES



```
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)
```

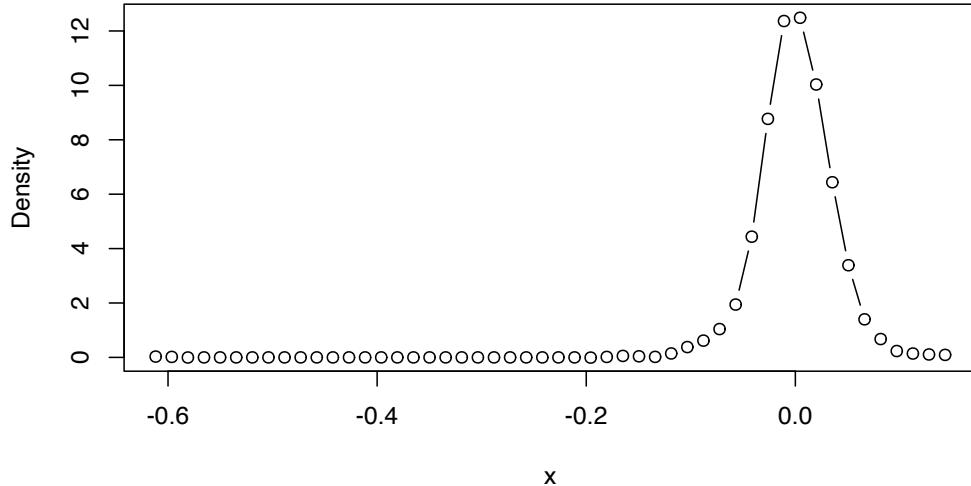


```
h_cv_np_ml <- npudensbw(dat = x, bwmethod = "cv.ml",
                           ckertype = "epanechnikov", ckerorder = 4)
```

```
## Multistart 1 of 1 |Multistart 1 of 1 |Multistart 1 of 1 |Multistart 1 of 1 /Multis
dens.np <- npudens(h_cv_np_ml)

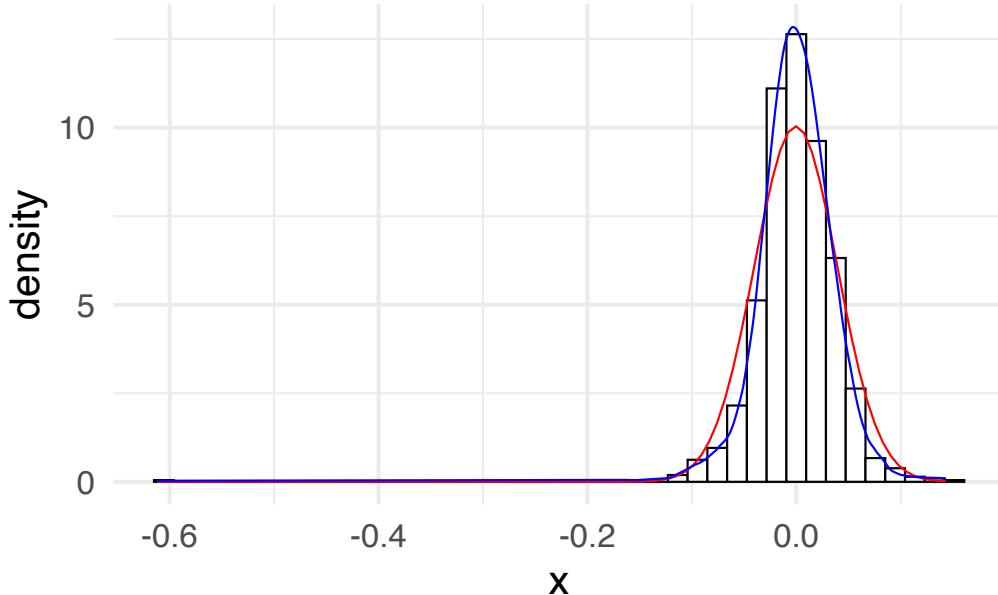
plot(dens.np, type = "b")
```

58 CAPÍTULO 2. ESTIMACIÓN NO-PARAMÉTRICA DE DENSIDADES



```
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 2.7. 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?

2.4. Ejercicios

Del libro de (Härdle y col. 2004) 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

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

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 dado que la distribución de $\hat{\mu}$ es normal.

Nota. ¿Qué pasaría si no conocemos la distribución de $\hat{\mu}$?

¿Cómo podemos encontrar ese intervalo de confianza?

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 varianza?

Note que esta pregunta es difícil ya que no tenemos ningún tipo de información adicional para inferir la variación de la varianza T_n .

Las dos técnicas que veremos a continuación nos permitirán extraer *información adicional* de la muestra para inferir propiedades distribucionales de T_n .

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

3.2. Jackknife

Esta técnica fue propuesta por (Quenouille 1949). Primero que todo se puede probar que existen estimadores que cumplen la siguiente propiedad:

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

para algún a and b .

Por ejemplo sea $\sigma^2 = \text{Var}(X_i)$ 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 elemento de la muestra.

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

Definimos el estimador de sesgo *jackknife* de T_n como

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

donde

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

y el estimador corregido por sesgo es: $T_{jack} = T_n - b_{jack}$.

En nuestro caso tendríamos lo siguiente:

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

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

Es decir, el sesgo aproximado (jackknife) del estimador T_n es 0.

Si se asume que T_n es un estimador del parámetro θ entonces se puede comprobar que b_{jack} cumple:

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

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 a través del siguiente estimador:

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

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

Ejercicio 3.2. Usando 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 3.3. Los pseudo-valores se estiman de forma directa como,

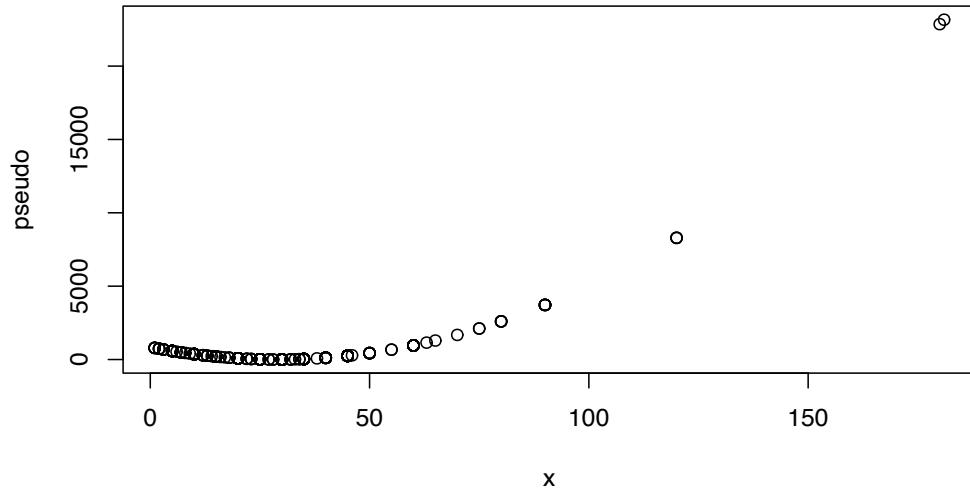
```
pseudo <- n * Tn - (n - 1) * Ti
```

```
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 asociación o correspondencia 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 los siguientes estimadores respectivos:

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

y

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

Nota. Sin embargo, se puede demostrar fácilmente que se pueden usar pseu-dovalores para construir una prueba normal de hipótesis.

Como los pseu-dovalores son idénticamente distribuidos entonces su promedio se ajusta de forma aproximada a una distribución normal a medida que el tamaño de la muestra aumenta. Por lo tanto, tenemos que

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

```
(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. Este fue propuesto por Bradley Efron en (Efron 1979).

Primero recordemos que estamos estimando la variabilidad propia de un estadístico a partir de una muestra. Asuma que este estadístico tiene la forma

$T_n = g(X_1, \dots, X_n)$ donde g es cualquier función (media, varianza, quantiles, etc).

Supongamos que conocemos la distribución real de los X 's, llamada $F(x)$ y asumamos que $T_n = \bar{X}_n$. Si uno quisiera estimar la varianza de T_n basta con hacer

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

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 utilizar el estimador empírico \hat{F}_n como estimador plug-in en la formulación de la varianza de T_n .

De manera sencilla se puede resumir la técnica de bootstrap como una simulación iid de la distribución \hat{F}_n de modo que se pueda conocer la varianza del estadístico T_n .

En simples 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,

$$\hat{s}_{\text{boot}} = \sqrt{v_{\text{boot}}}$$

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

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

producto de la ley de grandes números en ambos casos.

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 3.4. En este ejemplo podemos tomar $B = 1000$ y construir esa cantidad de veces nuestro estimador de varianza:

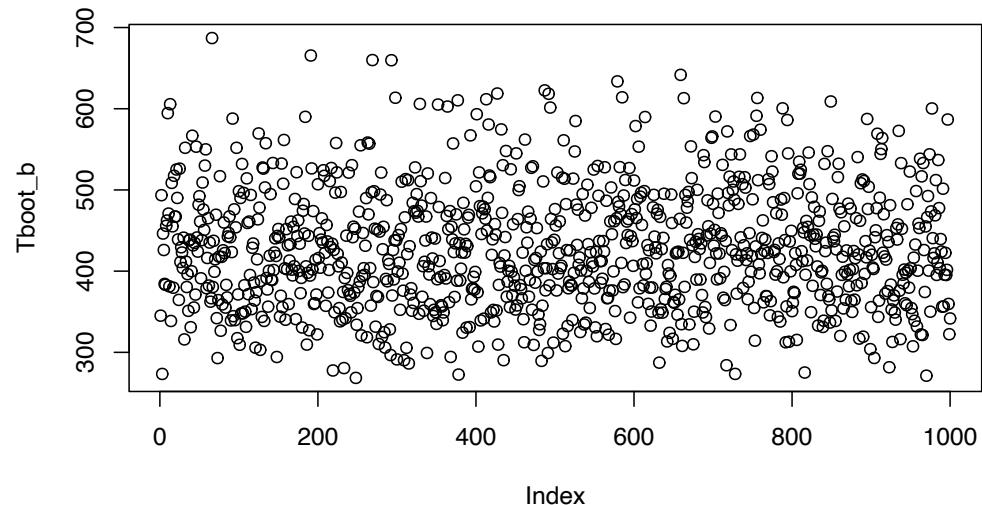
```
B <- 1000
Tboot_b <- NULL

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

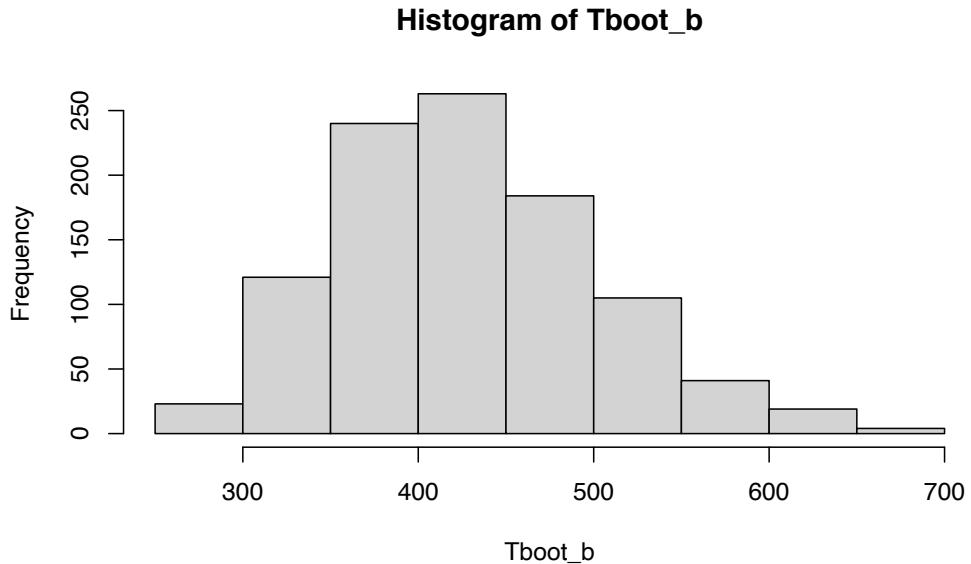
Tboot_b[1:10]

## [1] 345.1819 493.5279 273.3998 446.3071 426.0340 384.2662 383.2132 455.8139
## [9] 462.3363 594.5774
```

```
plot(Tboot_b)
```



```
hist(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
```

Nota. Si $\hat{\theta}$ es un estimador de θ (bajo cualquier método) entonces podemos sustituir el paso 1 en el algoritmo de Bootstrap por lo siguiente:

1. Seleccione $X_1^*, \dots, X_n^* \sim F_{\hat{\theta}}$

A este algoritmo modificado le llamamos Bootstrap paramétrico.

3.3.1. Intervalos de confianza

3.3.1.1. Intervalo Normal

Este es el más sencillo y se escribe como

$$T_n \pm z_{\alpha/2} \widehat{S}_{\text{boot}} \quad (3.4)$$

Nota. Este intervalo solo funciona si la distribución de T_n es normal.

El cálculo de este intervalo es

```
c(Tn - z * sdboot, Tn + z * sdboot)
```

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

3.3.1.2. Intervalo pivotal

Sea $\theta = T(F)$ y $\widehat{\theta}_n = T(\widehat{F}_n)$ y defina la cantidad pivotal $R_n = \widehat{\theta}_n - \theta$.

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

$$H(r) = \mathbb{P}_F(R_n \leq r).$$

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

$$a = \widehat{\theta}_n - H^{-1}\left(1 - \frac{\alpha}{2}\right) \quad \text{y} \quad b = \widehat{\theta}_n - H^{-1}\left(\frac{\alpha}{2}\right).$$

Se sigue que

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

Nota. $C_n^* = (a, b)$ es un intervalo de confianza al $(1 - \alpha)\%$.

El problema es que este intervalo depende de H desconocido.

Para resolver este problema, se puede construir una versión *bootstrap* de H usando lo que sabemos hasta ahora:

$$\widehat{H}(r) = \frac{1}{B} \sum_{b=1}^B I(R_{n,b}^* \leq r)$$

donde $R_{n,b}^* = \widehat{\theta}_{n,b}^* - \widehat{\theta}_n$.

Sea r_β^* el cuantil muestral de tamaño β de $(R_{n,1}^*, \dots, R_{n,B}^*)$ y sea θ_β^* el cuantil muestral de tamaño β de $(\theta_{n,1}^*, \dots, \theta_{n,B}^*)$.

Nota. Según la notación anterior se cumple que:

$$r_\beta^* = \theta_\beta^* - \widehat{\theta}_n$$

A partir de los estadísticos anteriores se puede construir un intervalo de confianza aproximado $C_n = (\widehat{a}, \widehat{b})$ al $(1 - \alpha)\%$ donde:

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

Nota. El intervalo de confianza pivotal de tamaño $1 - \alpha$ es

$$C_n = (2\widehat{\theta}_n - \widehat{\theta}_{((1-\alpha/2)B)}^*, 2\widehat{\theta}_n - \widehat{\theta}_{((\alpha/2)B)}^*)$$

El intervalo anterior para un nivel de 95% se estima de la siguiente forma

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

```
##      97.5%      2.5%
## 267.1250 552.9294
```

3.3.1.3. Intervalo pivotal studentizado

Una versión mejorada del intervalo pivotal sería a través de la normalización de los estimadores de T_n :

$$Z_n = \frac{T_n - \theta}{\widehat{\text{se}}_{\text{boot}}}.$$

Como θ 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 $T_{n,b}^*$ no de T_n .

Nota. Para calcular $Z_{n,b}^*$ requerimos estimar la varianza de $T_{n,b}^*$ para cada b .

Con esto se puede obtener cantidades $Z_{n,1}^*, \dots, Z_{n,B}^*$ que debería ser próximos a Z_n . (Bootstrap de los estadísticos normalizados)

Sea z_α^* el α -cuantil de $Z_{n,1}^*, \dots, Z_{n,B}^*$, entonces $\mathbb{P}(Z_n \leq z_\alpha^*) \approx \alpha$.

Define el intervalo

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

Justificado por el siguiente cálculo:

$$\begin{aligned} \mathbb{P}(\theta \in C_n) &= \mathbb{P}\left(T_n - z_{1-\alpha/2}^* \widehat{\text{se}}_{\text{boot}} \leq \theta \leq T_n - z_{\alpha/2}^* \widehat{\text{se}}_{\text{boot}}\right) \\ &= \mathbb{P}\left(z_{\alpha/2}^* \leq \frac{T_n - \theta}{\widehat{\text{se}}_{\text{boot}}} \leq z_{1-\alpha/2}^*\right) \\ &= \mathbb{P}\left(z_{\alpha/2}^* \leq Z_n \leq z_{1-\alpha/2}^*\right) \\ &\approx 1 - \alpha \end{aligned}$$

Note que para este caso tenemos que hacer bootstrap para cada estimador bootstrap calculado.

```

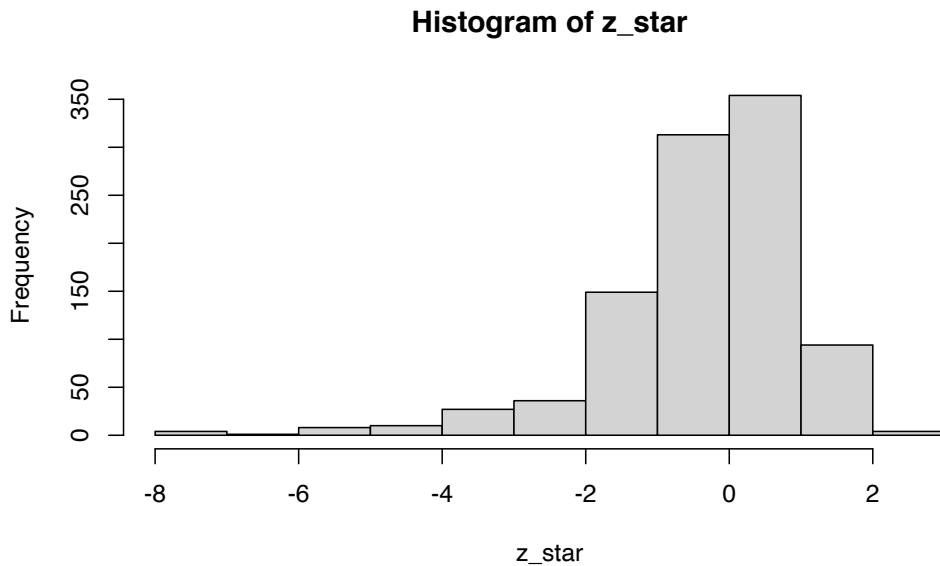
B <- 1000
Tboot_b <- NULL
Tboot_bm <- NULL
sdboot_b <- NULL

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
```

3.3.2. Resumiendo

Resumiendo todos los métodos de cálculo de intervalos obtenemos

```
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

Métodos lineales de regresión

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

4.1. Introducción al Aprendizaje Estadístico.

Supongamos que tenemos p variables de entrada que provocan una respuesta Y (variable dependiente) a través de la siguiente relación:

$$Y = f(X_1, \dots, X_p) + \varepsilon \quad (4.1)$$

donde f es desconocida, las variables X 's son las variables de entrada (covariables o predictores) y ε representa un error aditivo a la relación definida por f .

Hay dos motivos por los que estimamos f :

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

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

asumiendo que el valor medio del error ε es cero. Si tuvieramos valores nuevos de los X 's entonces podríamos estimar el valor que el corresponde a Y .

En este caso obtener una estructura óptima o precisa de la función \hat{f} no es importante, siempre y cuando sea posible obtener buenas predicciones de Y . Para entender mejor esta idea se puede definir:

- Error reducible:** Error de \hat{f} alrededor de f , el cual es propio de la escogencia del modelo.
- Error irreducible:** Error que escapa a una estimación perfecta de f . Puede venir de covariables no consideradas en el problema, fuentes de error que no se pueden cuantificar, etc.

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

asumiendo que f y X son conocidas y determinísticas.

2. **Inferencia:** Entender la relación entre X y Y , es decir entender cómo Y cambia como función de las covariables. En este caso sí nos interesa obtener un estimador preciso e interpretable de la función f . Las siguientes preguntas son de interés:

- ¿Cuáles covariables están asociadas con la variable respuesta o dependiente?
- ¿Cuál es la relación entre cada variable predictora y la respuesta?
- ¿La relación entre covariables y variable dependiente es lineal? o ¿la relación es más compleja?

4.1.1. Formas de estimar f

El proceso de estimación de f a través de \hat{f} se realiza sobre un subconjunto de los datos disponibles. A este conjunto se le llama *datos de entrenamiento*. El resto de los datos se puede utilizar para probar la capacidad predictiva del modelo seleccionado.

Existen varias clasificaciones de modelos para estimar f :

- Modelos paramétricos vs modelos no parámetricos. Los modelos pueden tener parámetros que facilitan el proceso de estimación, pero el número

de parámetros debe ser conservador para evitar situaciones de *sobreajuste*. Los modelos no-paramétricos requieren de mucha información para dar un buen ajuste, sea a través de una muestra grande o a través de manipular parámetros generales de suavidad (ancho de banda).

- Modelos predictivos vs modelos interpretativos. Entre más flexible (complejo) sea un modelo, más difícil es su interpretación, por lo tanto más difícil es hacer inferencia. Hay modelos muy flexibles que permiten hacer muy buena predicción, pero fácilmente se puede caer en sobreajuste.
- Modelos supervisados vs no supervisados. ¿La variable Y está disponible en la muestra?
- Modelos de regresión vs modelos de clasificación. ¿La variable Y es continua o es una variable categórica?

4.1.2. Medidas de bondad de ajuste

En el caso de regresión, la medida más utilizada es el Error Cuadrático Medio (MSE):

$$MSE = \frac{1}{n} \sum_{i=1}^n (y_i - \hat{f}(x_i))^2$$

calculada sobre la base de entrenamiento del modelo para evaluar la capacidad de ajuste de \hat{f} . Para evaluar la capacidad predictiva del modelo se puede usar el mismo concepto sobre la *base de prueba*. La diferencia entre la magnitud del MSE en los dos conjuntos de datos, puede ser un indicador de sobreajuste.

Para el caso de un problema de aprendizaje estadístico hay interpretaciones de los componentes de sesgo y varianza:

- *Varianza*: variación de \hat{f} ante cambios en los datos de entrenamiento. Modelos más flexibles tienen mayor varianza.
- *Sesgo*: error al aproximar la realidad complicada con un modelo más simple. Modelos más flexibles tienen menor sesgo.

Estrategia de búsqueda de modelos: conforme aumenta la flexibilidad de un modelo el sesgo disminuye, y la varianza no aumenta en el mismo ritmo. A partir de un cierto momento la disminución del sesgo no es lo suficientemente fuerte como para contrarrestar el crecimiento en varianza.

Conclusión: un modelo parsimonioso posiblemente garantizará un valor óptimo en MSE.

4.2. Regresión lineal

El caso más sencillo es cuando se asume que la relación es lineal y se describe de la siguiente forma:

$$Y = \beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p + \varepsilon.$$

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

Las covariables en un modelo de regresión pueden ser:

1. Cuantitativas: variables continuas.
2. Categóricas: variables tipo factor que admiten un número de niveles. Estas variables pueden ser ordinales o nominales, dependiendo si hay un orden natural en la escala de los niveles. Para incorporarla en el modelo de regresión debemos *codificar* la variable:

Ejemplo 4.1. Se tiene la variable G codificada con Casado (1), Soltero (2), Divorciado (3) y Unión Libre (4). Si queremos incorporar esta variable en una regresión podríamos usar la siguiente codificación:

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

que resulta en la matriz

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

Existen otras formas de codificar este tipo de variables, pero esta es una de las más usuales.

4.2.1. Forma matricial

Podemos escribir el modelo de regresión en forma matricial:

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

donde

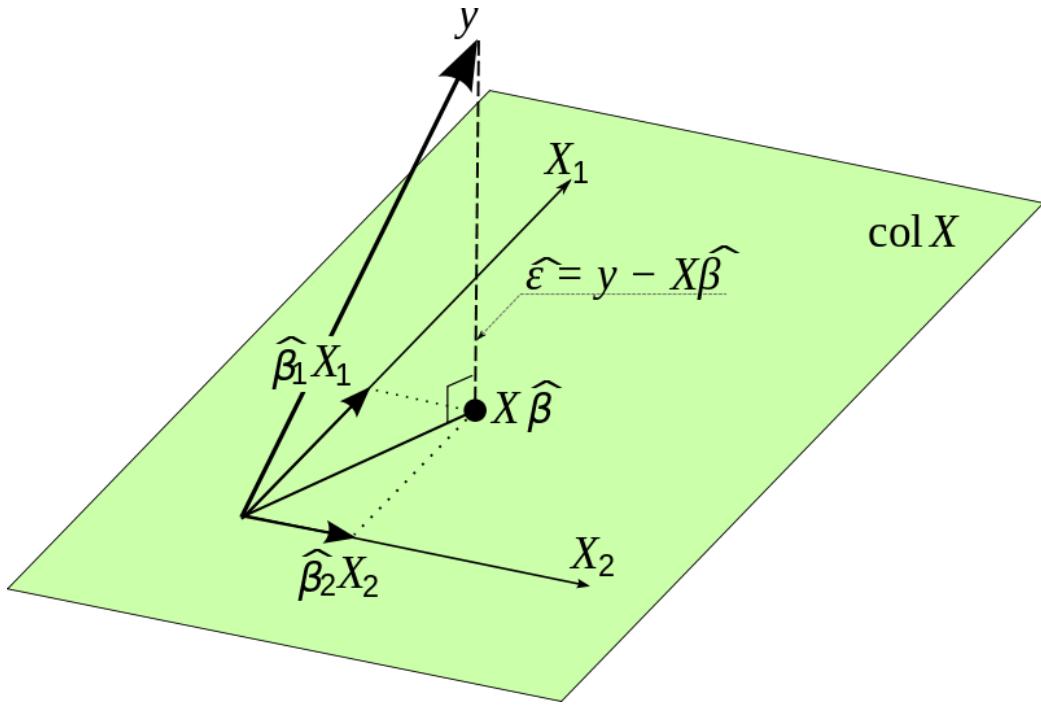
$$\begin{aligned} \mathbf{Y} &= \begin{pmatrix} Y_1 \\ \vdots \\ Y_n \end{pmatrix}_{n \times 1} & \mathbf{X} &= \begin{pmatrix} 1 & X_{1,1} & \cdots & X_{p,1} \\ \vdots & \vdots & \cdots & \vdots \\ 1 & X_{1,n} & \cdots & X_{p,n} \end{pmatrix}_{n \times (p+1)} \\ \boldsymbol{\varepsilon} &= \begin{pmatrix}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

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

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

$$\hat{\boldsymbol{\beta}} = \operatorname{argmin}_{\boldsymbol{\beta}} (\mathbf{Y} - \mathbf{X}\boldsymbol{\beta})^\top (\mathbf{Y} - \mathbf{X}\boldsymbol{\beta}) \quad (4.2)$$

$$= \operatorname{argmin}_{\boldsymbol{\beta}} \sum_{i=1}^n \left(Y_i - \beta_0 - \sum_{j=1}^p X_{j,i} \beta_j \right)^2 \quad (4.3)$$



Por lo tanto buscaríamos minimizar la suma de *residuos* al cuadrado.

Suponga que γ es un vector cualquiera en \mathbb{R}^{p+1} y defina $V := \{\mathbf{X}\gamma, \gamma \in \mathbb{R}^{p+1}\}$, es decir el espacio lineal generado por las columnas (covariables) de \mathbf{X} . Buscamos entonces un vector β que cumpla:

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

Entonces dado que $\mathbf{Y} - \mathbf{X}\beta \perp V$, es decir $\mathbf{Y} - \mathbf{X}\beta \perp \mathbf{X}\gamma, \forall \gamma \in \mathbb{R}^{p+1}$ entonces:

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

Donde se asume que $\mathbf{X}^\top \mathbf{X}$ 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 $\hat{\beta} \neq \tilde{\beta}$ tal que $\mathbf{X}\hat{\beta} = \mathbf{X}\tilde{\beta}$. A $\hat{\beta}$ se le llama estimador por mínimos cuadrados de β .

En el caso de predicción tenemos que

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

Donde H es la matriz “techo” o “hat”. La matriz H es la matriz de proyección de \mathbf{Y} al espacio de las columnas de X .

Ejercicio 4.1. Suponga que tenemos la regresión simple

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

Verifique que los estimadores de mínimos cuadrados de β_0 y β_1 son:

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

usando los siguiente métodos:

1. El método de proyecciones.
2. Aplicando el criterio de mínimos cuadrados. Ecuación (4.3).

4.2.2. 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)

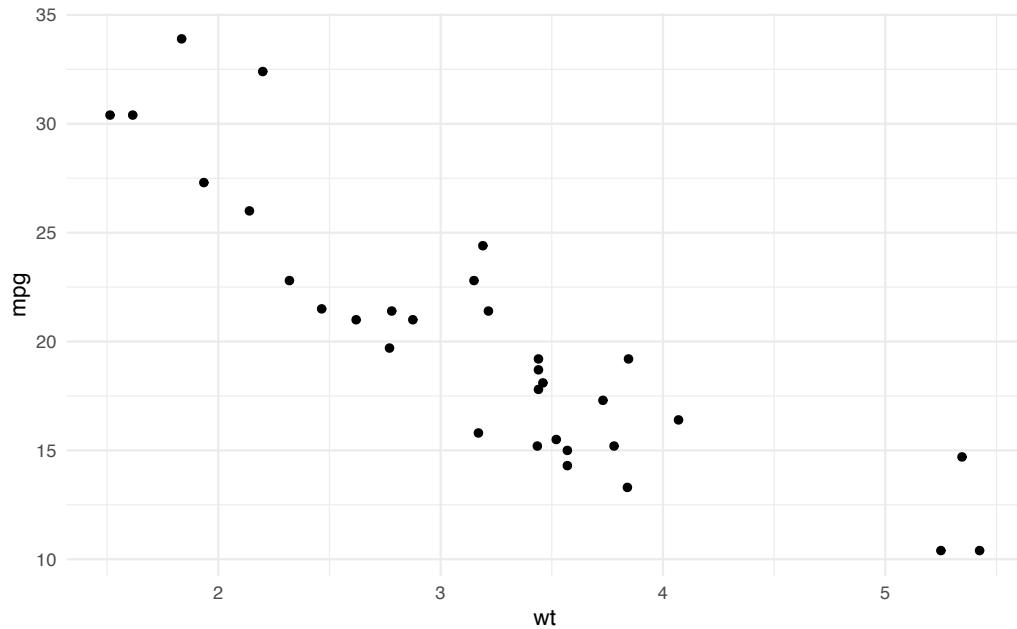
##      mpg          cyl          disp          hp          drat
##  Min.   :10.40   4:11   Min.   :71.1   Min.   :52.0   Min.   :2.760
##  1st Qu.:15.43   6: 7   1st Qu.:120.8  1st Qu.:96.5   1st Qu.:3.080
##  Median :19.20   8:14   Median :196.3   Median :123.0  Median :3.695
##  Mean   :20.09                    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)

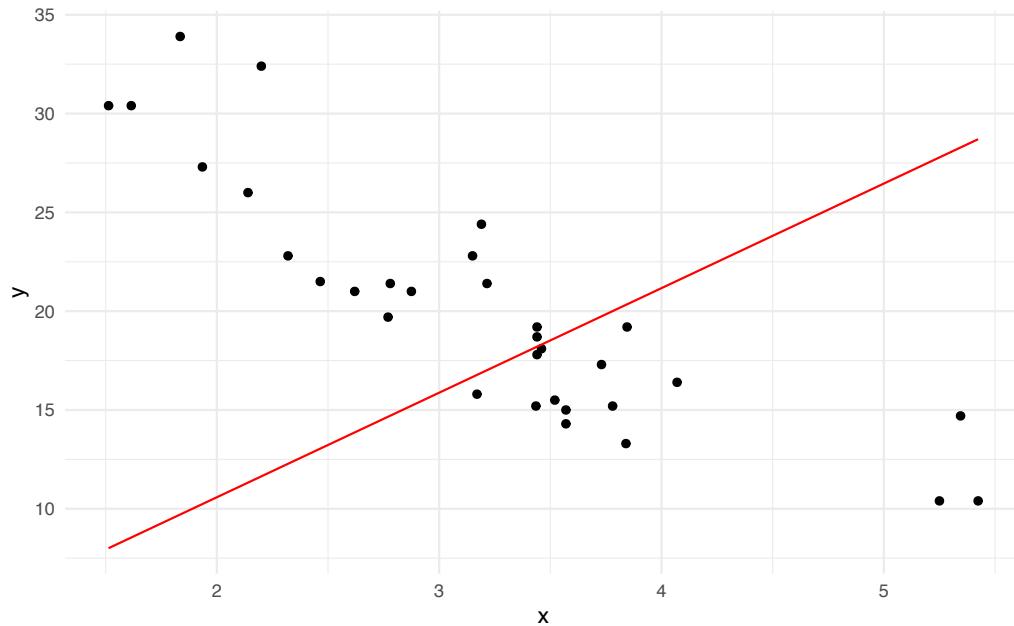
## [1] 21.0 21.0 22.8 21.4 18.7 18.1

(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()
```



en donde podemos concluir que la relación lineal no modela de manera apropiada la relación observada en los datos. Por lo tanto es necesario incluir el intercepto β_0 al modelo lineal:

```
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)
```

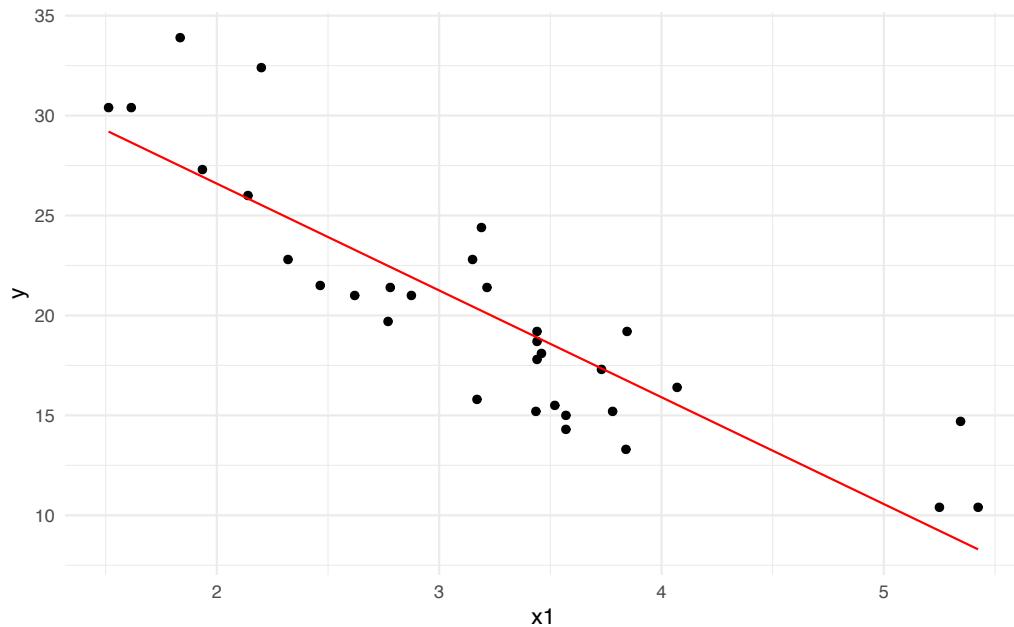
```
## [1] 21.0 21.0 22.8 21.4 18.7 18.1
```

```
(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()
```



El mismo resultado se puede obtener a través del comando `lm`:

```
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 trabajar con la variable categorica `cyl` (Número de cilindros) como única covariante. Lo que se debe hacer es codificar la variable categórica:

```
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

```

4.3. Propiedades estadísticas

Hasta ahora se han hecho pocos supuestos acerca de la distribución de los datos. Si asumimos que las observaciones Y_i son no correlacionadas y que tienen varianza constante σ^2 y además las covariables son fijas (no aleatorias), entonces:

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

Note que σ^2 puede ser estimado a través de:

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

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 $\mathbb{E}[\hat{\sigma}^2] = \sigma^2$ (estimador insesgado).

Para poder hacer inferencia sobre β y σ^2 se puede asumir además que los errores son gaussianos:

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

y de esta forma se obtiene:

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

Y además:

$$\hat{\beta} \sim \mathcal{N}(\beta, \sigma^2 (X^\top X)^{-1})$$

Por otro lado se puede comprobar que:

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

y además se puede comprobar que $\hat{\beta}$ y $\hat{\sigma}^2$ son independientes.

Ejercicio 4.2. Encuentre la varianza para $\hat{\beta}_0$ y $\hat{\beta}_1$ para el caso de la regresión simple.

4.3.1. Prueba t

La significancia de los parámetros β_j se puede verificar a través de la siguiente prueba de hipótesis:

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

En donde el estadístico de prueba es:

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

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

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

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

4.3.2. Prueba F

Si uno busca medir la significancia de todos los parámetros β_j de forma simultánea, excepto el intercepto. En este caso podemos definir la siguiente hipótesis nula:

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

Lo cual es equivalente a comparar el modelo nulo $Y = \beta_0 + \varepsilon$ contra el modelo completo $Y = \beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p + \varepsilon$.

Defina la suma total de cuadrados (TSS) y la suma de residuos al cuadrado (RSS) como:

$$TSS = \sum_{i=1}^n (Y_i - \bar{Y})^2$$

$$RSS = \sum_{i=1}^n (Y_i - \hat{Y}_i)^2$$

Entonces el estadístico de prueba es:

$$F = \frac{\frac{TSS - RSS}{p}}{\frac{RSS}{n-p-1}} \stackrel{H_0}{\sim} \frac{\chi_p^2}{\chi_{n-p-1}^2}.$$

y rechazaríamos H_0 al nivel α si:

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

Si por otro lado queremos probar que un conjunto de q covariables son no-significativas entonces probamos (sin pérdida de generalidad):

$$H_0 : \beta_{p-q+1} = \beta_{p-q+2} = \cdots = \beta_p = 0$$

a través de la comparación de un modelo completo y uno reducido:

$$Y = \beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p + \varepsilon \quad \text{Modelo completo}$$

$$Y = \beta_0 + \beta_1 X_1 + \cdots + \beta_{p-q} X_{p-q} + \varepsilon \quad \text{Modelo reducido}$$

usando el estadístico de prueba:

$$F = \frac{\frac{RSS_0 - RSS}{q}}{\frac{RSS}{n-p-1}} \stackrel{H_0}{\sim} \frac{\chi_q^2}{\chi_{n-p-1}^2}.$$

donde RSS_0 es la suma de residuos al cuadrado del modelo reducido. En este caso se rechazaría H_0 al nivel α si $F > F_{q,n-p-1,1-\alpha}$.

4.3.3. Laboratorio

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

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

##
## 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 < 2e-16 ***
## wt          -5.3445    0.5591 -9.559 1.29e-10 ***
## ---
## 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: 1.294e-10

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

##
## 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) 37.2851    1.8776 19.858 < 2e-16 ***
## wt          -5.3445    0.5591 -9.559 1.29e-10 ***
## cyl         -1.8473    0.9851 -1.870  0.8507    
## ---
```

```

## (Intercept) 33.9908     1.8878   18.006 < 2e-16 ***
## 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: 3.594e-11
anova(modelo_wt, modelo_wt_cyl)

## Analysis of Variance Table
##
## Model 1: mpg ~ wt
## Model 2: mpg ~ wt + cyl
##   Res.Df   RSS Df Sum of Sq    F    Pr(>F)
## 1     30 278.32
## 2     28 183.06  2     95.263 7.2856 0.002835 **
## ---
## Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ',' 1
modelo_nulo <- lm(mpg ~ 1, data = mtcars)
summary(modelo_nulo)

##
## Call:
## lm(formula = mpg ~ 1, data = mtcars)
##
## Residuals:
##       Min     1Q Median     3Q    Max 
## -9.6906 -4.6656 -0.8906  2.7094 13.8094 
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)    
## (Intercept) 20.091     1.065   18.86 <2e-16 ***
## ---
## Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ',' 1

```

```

## 
## Residual standard error: 6.027 on 31 degrees of freedom
anova(modelo_nulo, modelo_wt_cyl)

## Analysis of Variance Table
##
## Model 1: mpg ~ 1
## Model 2: mpg ~ wt + cyl
##   Res.Df   RSS Df Sum of Sq    F    Pr(>F)
## 1     31 1126.05
## 2     28  183.06  3    942.99 48.079 3.594e-11 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

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

```

4.4. Medida de bondad de ajuste

A través de la prueba F uno puede concluir si un modelo es significativo o no bajo un cierto nivel de confianza, o bien puede comparar si un modelo reducido es más significativo que uno completo, pero no nos da herramientas para decidir si un modelo 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**.

Interpretación: proporción de variabilidad en Y que es explicada a través de las covariables en X . Ya que $\text{TSS} - \text{RSS}$ representa la variabilidad explicada a través del modelo de regresión.

Limitación: puede tener un valor alto bajo un número grande de covariables, ya que RSS tiende a ser bajo conforme aumenta la complejidad del modelo (sobreajuste).

Estadístico R^2 ajustado

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

4.4.1. 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)
```

4.4.1.1. R^2

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

## [1] 1404.421

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

## [1] 256.8679

1 - RSS/TSS

## [1] 0.8171005
```

Otra forma de entender el R^2 es notando que

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

## [1] 0.8171005
```

4.4.1.2. R^2 ajustado

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

## [1] 1.405827

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

## [1] 0.2576408

1 - RSS_adj/TSS_adj

## [1] 0.8167336
```

4.4.1.3. `summary`

```
summary(fit)

##
## Call:
## lm(formula = y ~ x1 + x2)
##
## Residuals:
##    Min      1Q  Median      3Q     Max
## -1.73583 -0.35052  0.01175  0.33270  1.42618
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 1.05443   0.03252  32.42   <2e-16 ***
## x1          1.02131   0.01573  64.92   <2e-16 ***
## x2          0.91189   0.05655  16.13   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.5076 on 997 degrees of freedom
## Multiple R-squared:  0.8171, Adjusted R-squared:  0.8167
## F-statistic: 2227 on 2 and 997 DF, p-value: < 2.2e-16
```

4.5. Predicción

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

1. **Error Reducible:** Recuerde que $\hat{Y} = X\hat{\beta}$ 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

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

Por lo tanto un **intervalo de confianza** al $1 - \alpha$ para $X_0 \beta$ es

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

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

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

4.5.1. 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

```

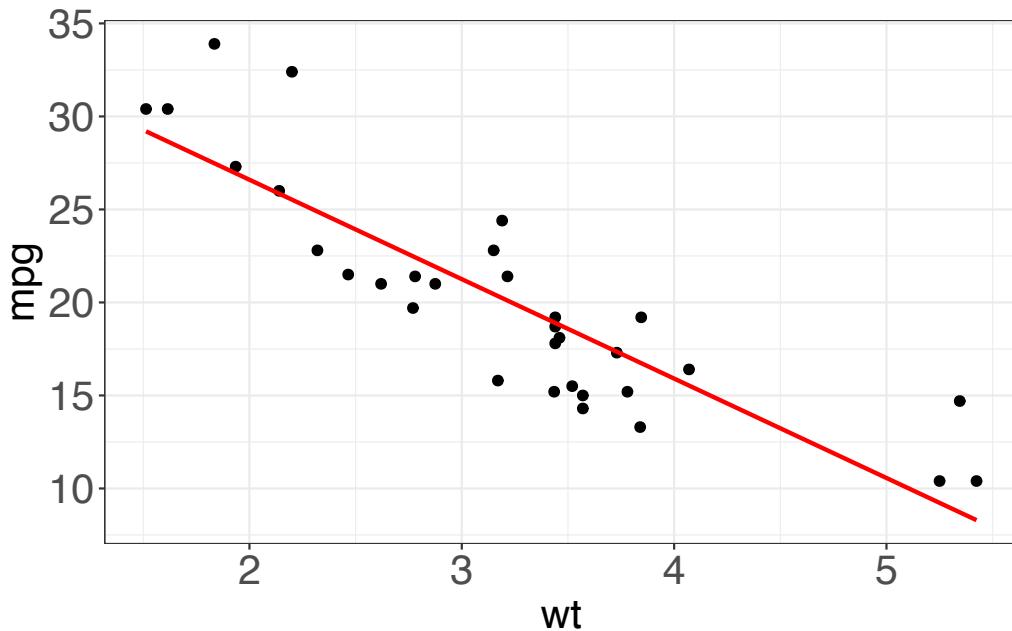
4.5.1.1 *Áinsta de la regresión sin intervalos de confianza*

```

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

```

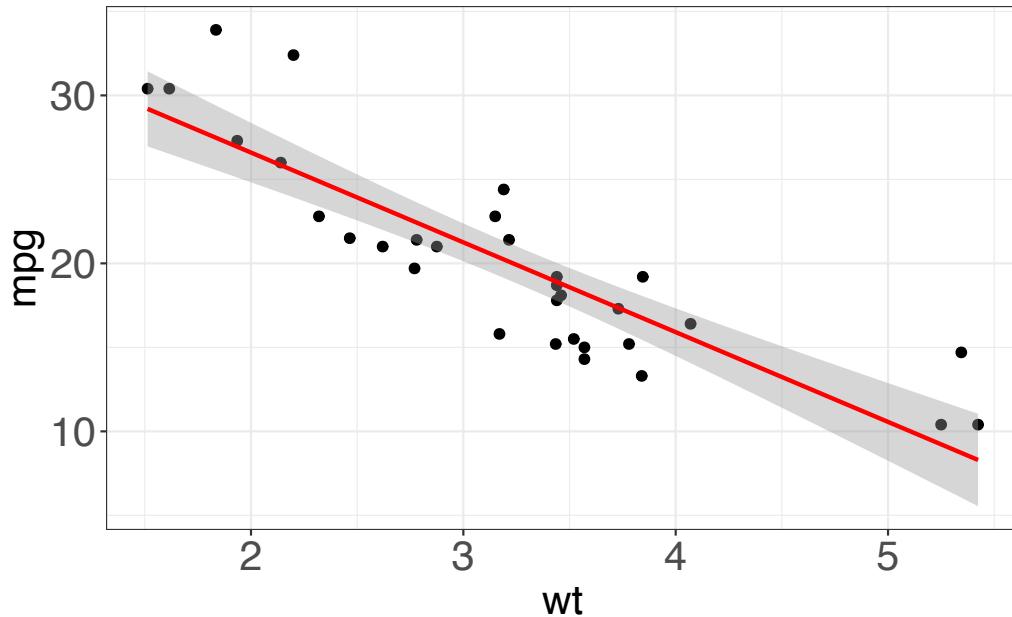


```
# # Guardar el gráfico en un archivo pdf
# ggsave(filename = 'linear_reg_sin_IC.pdf') #
```

4.5.1.2. 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,
                      se = TRUE,          # Agregar la línea de regresión
                      size = 1,            # Incluir el intervalo de confianza
                      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') #
```

4.5.1.3. 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"))

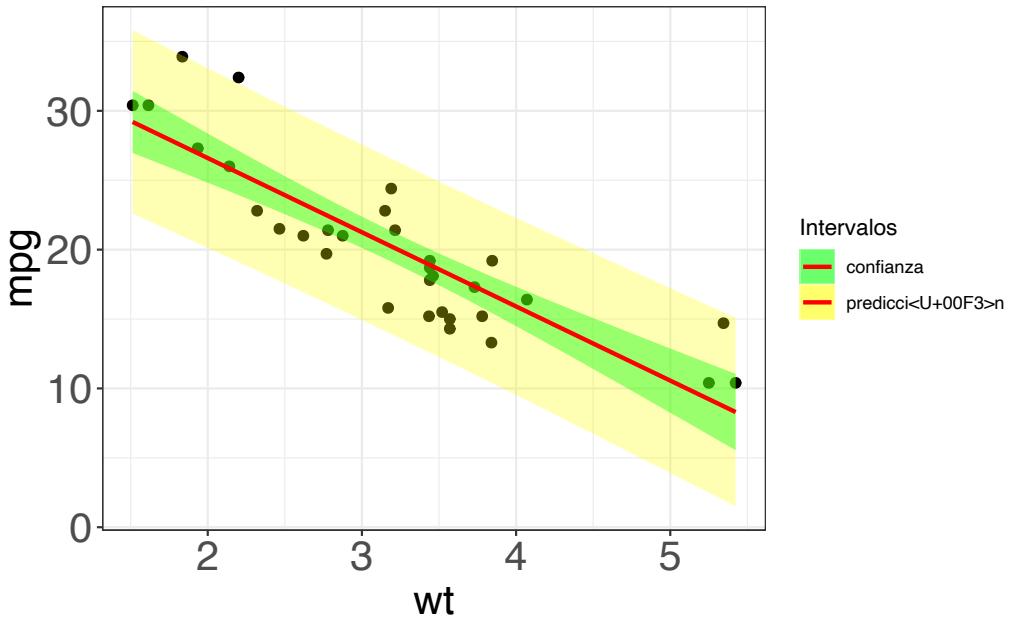
p <- ggplot(mtcars.pred, aes(x = wt, y = mpg))
# Use círculos de tamaño 2
p <- p + geom_point(size = 2)
# Agregue una banda de tamaño [lwr, upr]
# para cada punto y llámela 'predicción'
p <- p + geom_ribbon(aes(ymin = lwr, ymax = upr, fill = "predicción"),
                      alpha = 0.3)
# Agregue 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') #

```

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

```

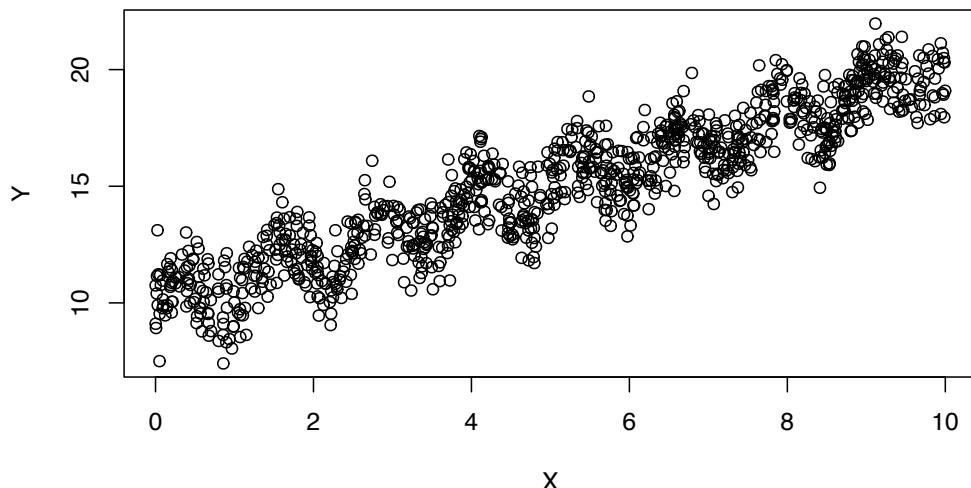
n <- 1000

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 
## -3.4587 -0.8232  0.0468  0.8709  3.4115 
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)    
## (Intercept)  11.0000    0.0500 220.000  <2e-16 ***
## X            1.0000    0.0000  100.000  <2e-16 ***
```

```

## (Intercept) 10.01402    0.07847 127.61   <2e-16 ***
## X           0.98895    0.01340  73.81   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.208 on 998 degrees of freedom
## Multiple R-squared:  0.8452, Adjusted R-squared:  0.845
## F-statistic:  5448 on 1 and 998 DF,  p-value: < 2.2e-16
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 = "predicci<U+00F3>n",
), 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)

```

4.6. Interacciones

Suponga un modelo lineal con dos covariables:

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

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 teniendo la misma estructura aunque hayamos cambiado el X_1 . Este fenómeno ocurre siempre bajo transformaciones lineales de las variables.

Ahora suponga que tenemos el siguiente modelo:

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

y aumentamos en 1 el X_1 :

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

Note que en este caso no se logra mantener el mismo tipo de estructura. Una forma de arreglar el problema es incluir las *interacciones* junto con todos sus *efectos principales*.

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

Este modelo se le conoce como modelo lineal con interacciones (caso de 2 covariables). Este modelo considera la posible interacción entre las covariables X_1 y X_2 que permiten cambiar tanto el intercepto como las pendientes de los efectos principales.

Principio de jerarquía. Con el fin de mantener la estructura del modelo lineal, siempre es necesario incluir los efectos principales cuando se determina que una interacción entre ellos es significativa.

Ejercicio 4.3. Compruebe que para el caso anterior, si aumenta en una unidad X_1 , el modelo preserva su estructura.

4.6.1. Laboratorio

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

```
# La funci<U+00F3>n across y where solo funciona
# solo para dplyr 1.0 Si tienen otra
# versi<U+00F3>n, 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 4.91e-16 ***
## 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: 2.744e-10
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 6.12e-12 ***
## 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: 2.744e-10
```

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 < 2e-16 ***
## wt:disp     -0.0072897  0.0009721 -7.499 2.33e-08 ***
## ---
## 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: 2.329e-08

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 < 2e-16 ***
```

```

## wt:disp      -0.013127   0.002714  -4.837 3.69e-05 ***
## ---
## 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: 3.686e-05

```

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 2.96e-14 ***
## wt          -6.495680   1.313383 -4.946 3.22e-05 ***
## 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: 1.158e-11
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 5.87e-13 ***
## wt          -6.495680  1.313383 -4.946 3.22e-05 ***
## 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: 1.158e-11

```

Ejercicio 4.4. 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))

```

4.7. Supuestos

El modelo lineal tiene los siguientes supuestos:

Linealidad En la forma lineal de la relación variable dependiente-covariables.

Errores centrados $\mathbb{E}(\varepsilon_i) = 0$.

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 (**error irreducible**) no depende de las variables independientes u otro factor.

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 $t \neq s$: si para una observación dada existe un error, este no debe depender del error de otra observación.

Si este supuesto no se cumple puede provocar que los errores estándar en

intervalos de confianza y predicción sean subestimados. Es decir que un intervalo del 95 % tendrá un margen de error menor y se rechazaría más fácilmente la hipótesis nula de las pruebas t y F .

Multicolinealidad Se asume que la matriz $X^T X$ es invertible, es decir X es una matriz de rango completo. Para esto cada una las covariables no debe ser linealmente dependientes, es decir $X^T X$ de debe acercarse a ser a una matriz singular con determinante cercano a 0. Es decir que cada variable explica aproximadamente “un aspecto o característica” del modelo. Sin embargo puede pasar que varias variables expliquen la misma característica y el modelo se vuelve **inestable** por decidir entre las dos variables. Por ejemplo: la temperatura en grados centígrados y farenheit.

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

$$\text{Var}(\hat{\beta}) = \sigma^2(X^\top X)^{-1}$$

Más observaciones que predictores En caso contrario existen formas alternativas de definir el problema de regresión. (Volveremos a esto cuando veamos selección de modelos)

4.7.1. Chequeos básicos de las hipótesis de regresión lineal

4.7.1.1. Linealidad, 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.

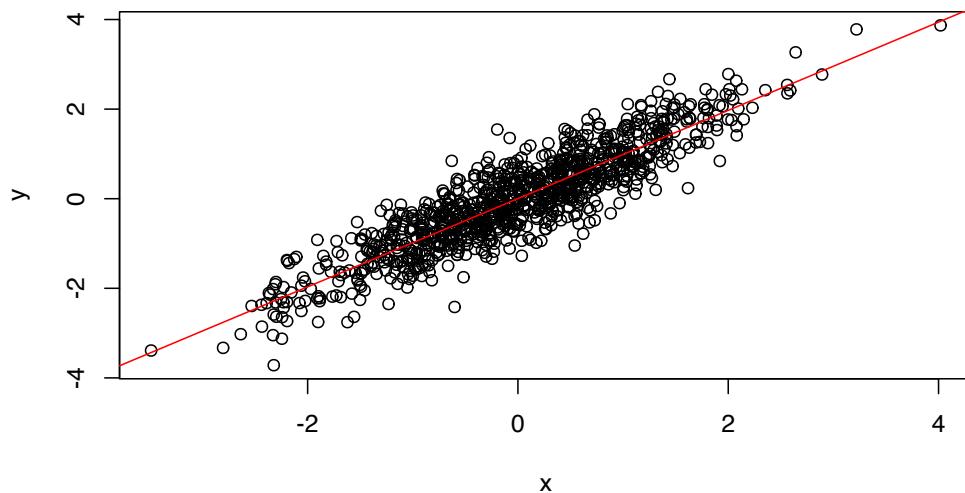
Se pueden aplicar transformaciones para resolver estos problemas. Normalmente se usan transformaciones como raíz cuadrada o logaritmos.

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")
```

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")
```

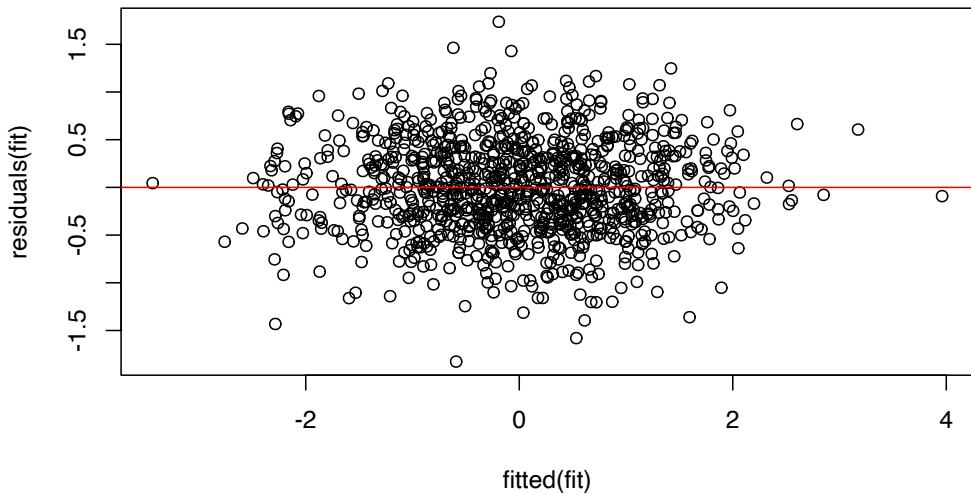
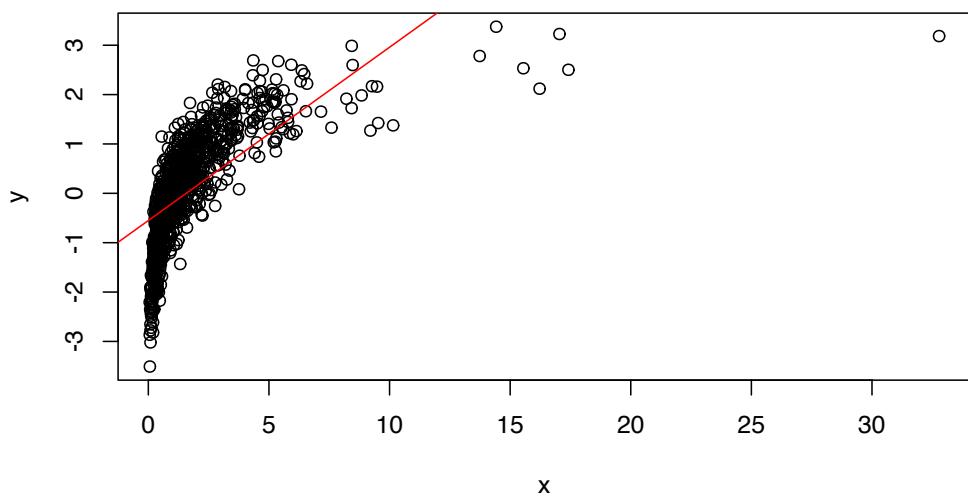


Figura 4.1: Gráfico de residuos caso lineal



```
plot(fitted(fit), residuals(fit))
abline(h = 0, col = "red")
```

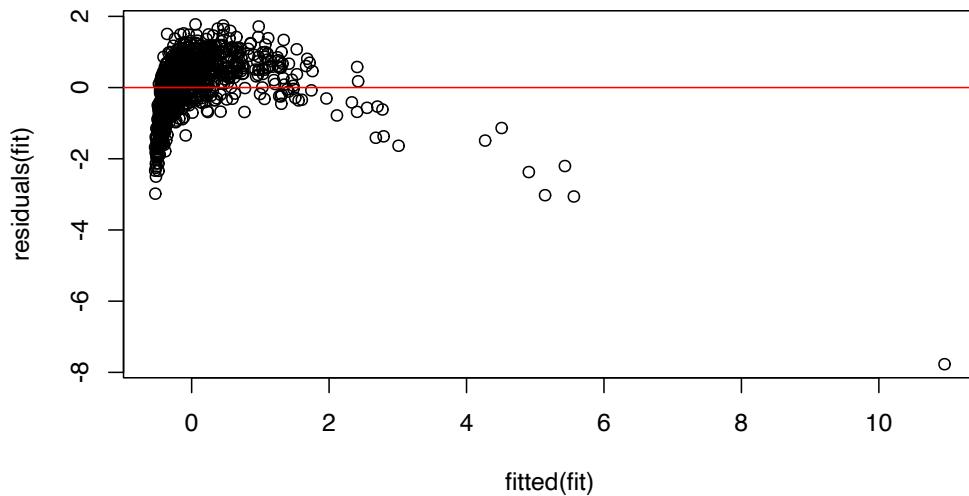
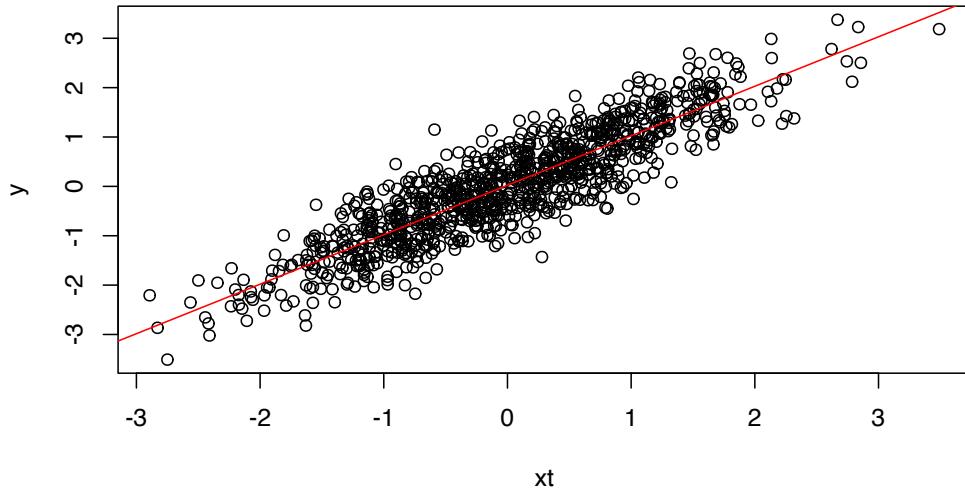


Figura 4.2: Gráfico de residuos caso no-lineal

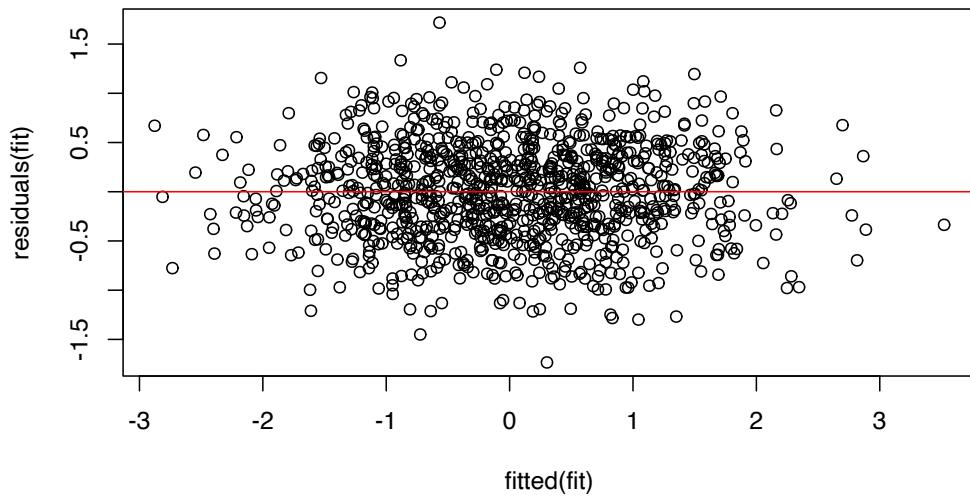
Caso no-lineal transformado

```
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")
```



4.7.1.2. 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

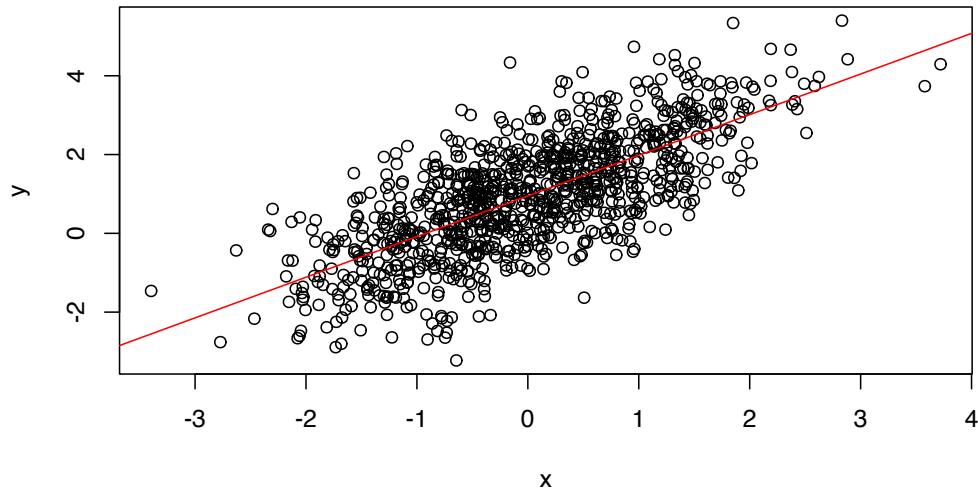
$$\rho(k) = \begin{cases} 1 & k = 0 \\ 0 & k \neq 0. \end{cases}$$

Se calcula la función de autocorrelación empírica y se grafica para analizar su comportamiento

Caso ideal

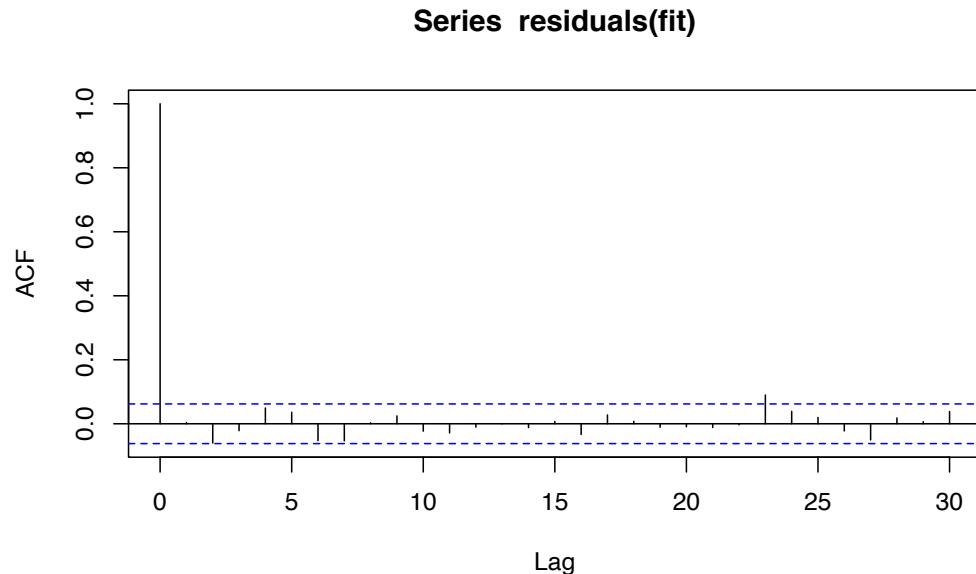
```
x <- rnorm(1000)
y <- 1 + x + rnorm(1000, sd = 1)

fit <- lm(y ~ x)
plot(x, y)
abline(a = coef(fit)[1], b = coef(fit)[2], col = "red")
```



```
summary(fit)
```

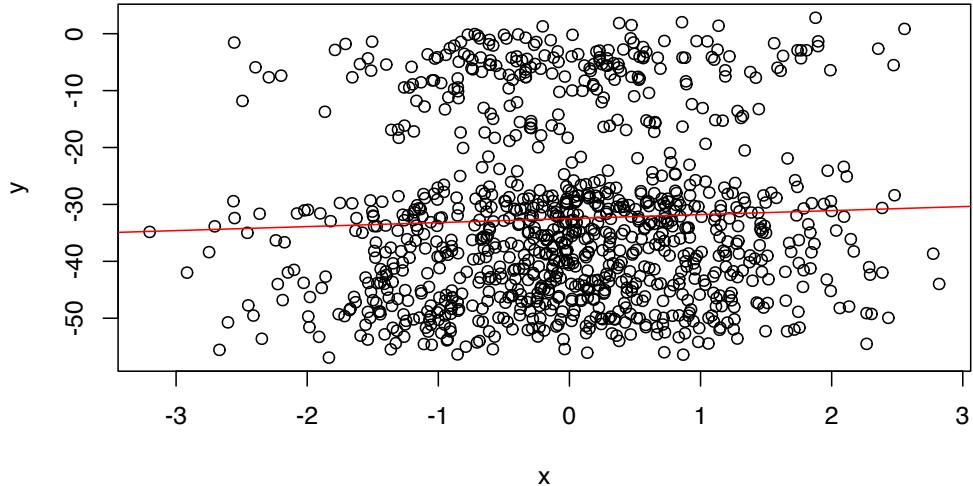
```
##  
## Call:  
## lm(formula = y ~ x)  
##  
## Residuals:  
##      Min       1Q   Median       3Q      Max  
## -3.5109 -0.6942  0.0465  0.6879  3.5523  
##  
## Coefficients:  
##                 Estimate Std. Error t value Pr(>|t|)  
## (Intercept)  0.94931    0.03237  29.33  <2e-16 ***  
## x           1.03167    0.03166  32.59  <2e-16 ***  
## ---  
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ',' 1  
##  
## Residual standard error: 1.024 on 998 degrees of freedom  
## Multiple R-squared:  0.5155, Adjusted R-squared:  0.515  
## F-statistic: 1062 on 1 and 998 DF,  p-value: < 2.2e-16  
acf(residuals(fit))
```



Caso errores auto-correlacionados

```
x <- rnorm(1000)
y <- 1 + x + diffinv(rnorm(999, sd = 1), lag = 1)

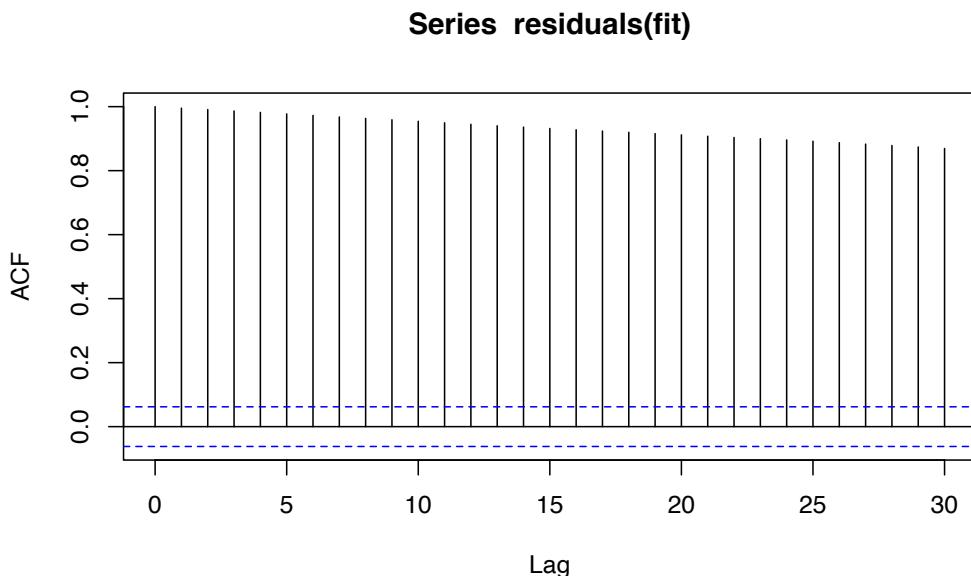
fit <- lm(y ~ x)
plot(x, y)
abline(a = coef(fit)[1], b = coef(fit)[2], col = "red")
```



```
summary(fit)

##
## Call:
## lm(formula = y ~ x)
##
## Residuals:
##    Min     1Q   Median     3Q    Max 
## -24.520 -11.918  -2.747   5.791  34.094 
## 
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)    
## (Intercept) -32.4938    0.4831 -67.268 <2e-16 ***
## x            0.7024    0.4731   1.485   0.138    
## ---        
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1 
## 
## Residual standard error: 15.27 on 998 degrees of freedom
## Multiple R-squared:  0.002204, Adjusted R-squared:  0.001204 
## F-statistic: 2.204 on 1 and 998 DF,  p-value: 0.1379
```

```
acf(residuals(fit))
```



4.7.1.3. Normalidad de los errores

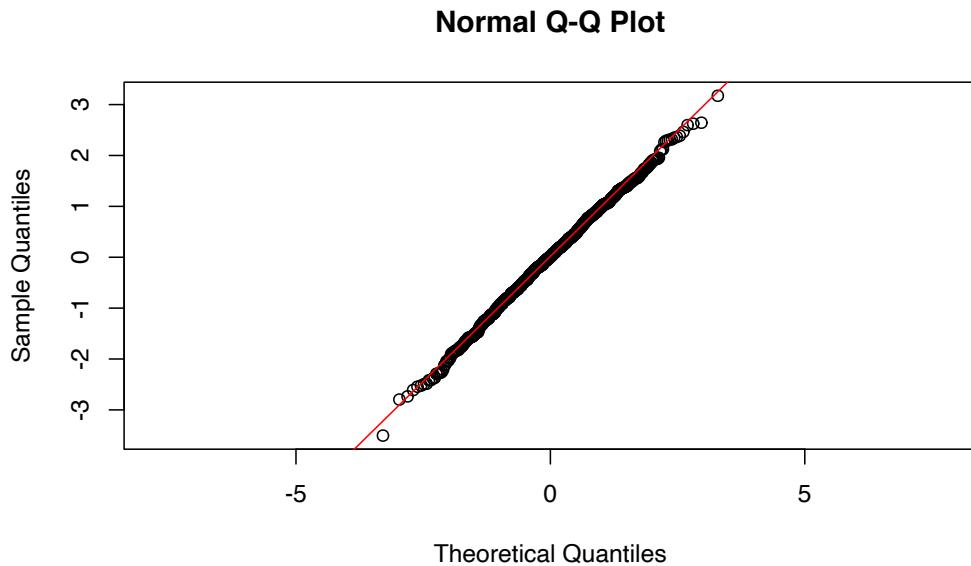
Este hipótesis es crucial para hacer las pruebas t y F que vimos anteriormente.

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)

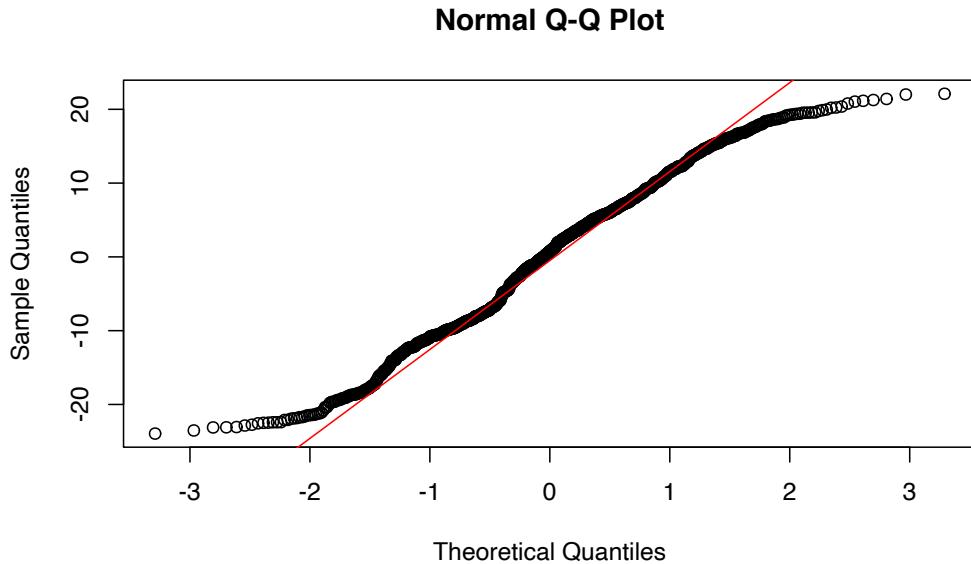
qqnorm(residuals(fit), asp = 1)
qqline(residuals(fit), col = "red")
```



Caso errores auto-correlacionados

```
x <- rnorm(1000)
y <- 1 + x + diffinv(rnorm(999, sd = 1), lag = 1)
fit <- lm(y ~ x)

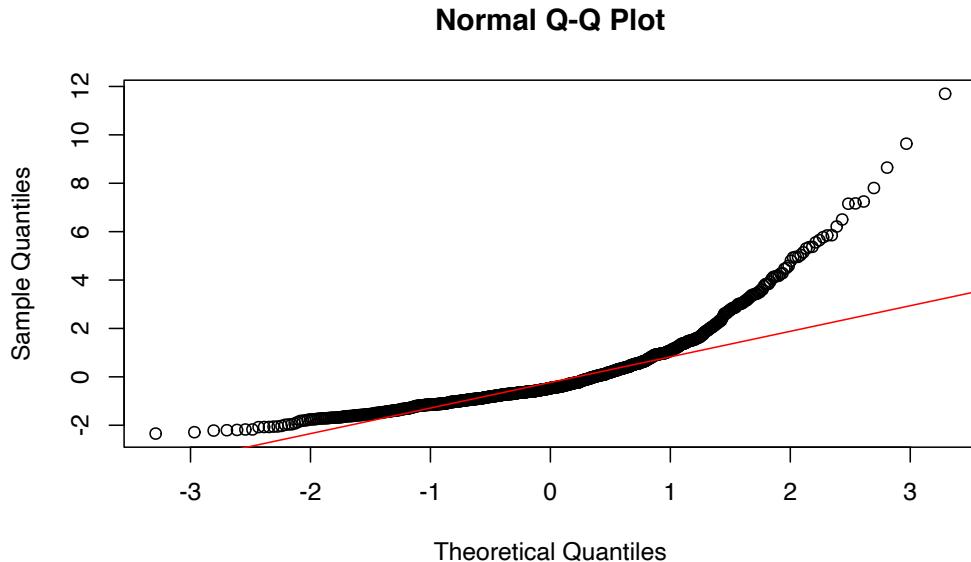
qqnorm(residuals(fit), asp = 0)
qqline(residuals(fit), col = "red")
```



Caso no-lineal

```
x <- rnorm(1000)
y <- x^2 + rnorm(1000, sd = 0.5)
fit <- lm(y ~ x)

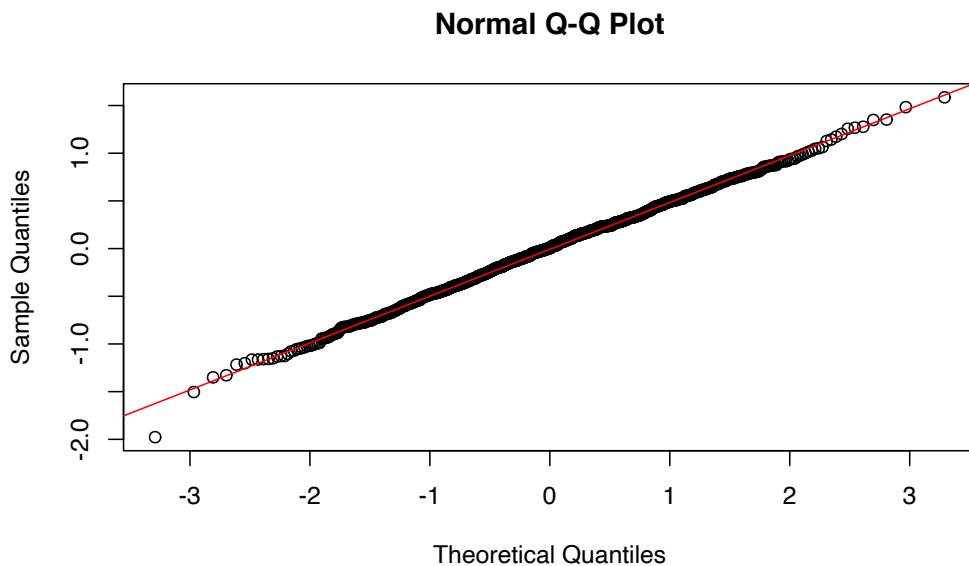
qqnorm(residuals(fit), asp = 0)
qqline(residuals(fit), col = "red")
```



```
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
## -1.97710 -0.33936  0.00447  0.32359  1.58603
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.02548    0.01877 -1.358   0.175
## x           -0.02164    0.01667 -1.298   0.195
## I(x^2)      0.99402    0.01222 81.335 <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ',' 1
```

```
##
## Residual standard error: 0.4902 on 997 degrees of freedom
## Multiple R-squared:  0.8692, Adjusted R-squared:  0.8689
## F-statistic:  3312 on 2 and 997 DF,  p-value: < 2.2e-16
qqnorm(residuals(fit), asp = 0)
qqline(residuals(fit), col = "red")
```



4.7.1.4. 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 $R^2_{X_j|X_{-j}}$ 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:

$$\text{VIF}(\hat{\beta}_j) = \frac{1}{1 - R_{X_j|X_{-j}}^2}$$

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

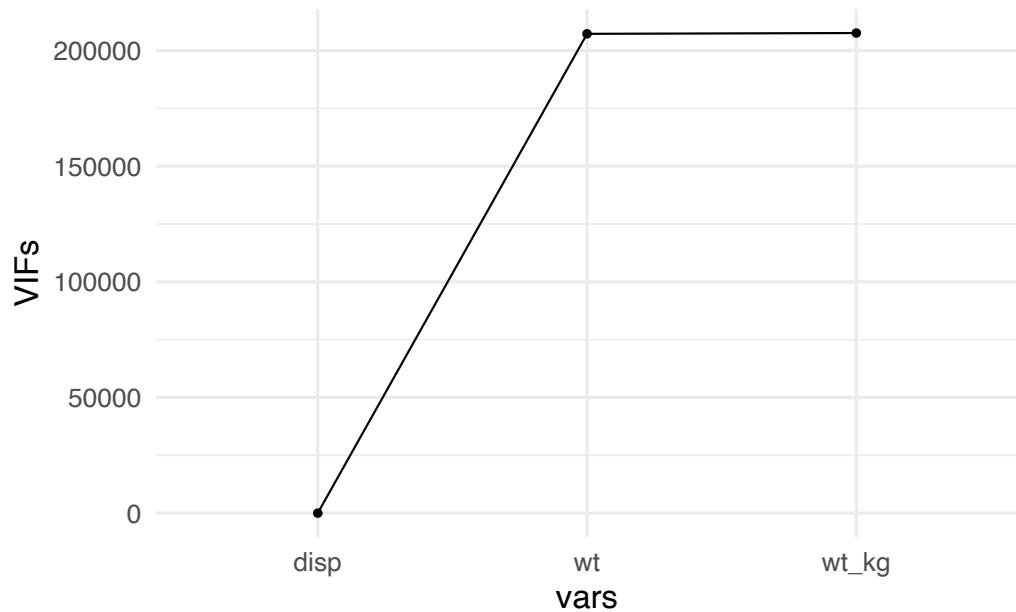
La variable `wt` está en unidades de 1000lb. La convertimos a Kilogramos.

```
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
## -3.567 -1.855 -1.014  1.655  6.686
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 36.202424   2.159407 16.765 3.92e-16 ***
## disp        -0.014316   0.008937 -1.602  0.1204
## wt          453.706761 232.492433  1.951  0.0611 .
## wt_kg       -1.009261   0.513377 -1.966  0.0593 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##  
## Residual standard error: 2.782 on 28 degrees of freedom  
## Multiple R-squared:  0.8075, Adjusted R-squared:  0.7869  
## F-statistic: 39.15 on 3 and 28 DF,  p-value: 3.766e-10  
  
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)
```



4.7.2. Otros chequeos importantes

4.7.2.1. Puntos extremos

Estos puntos son aquellos que Y_i esta lejos de \hat{Y}_i , es decir son puntos en donde los residuos son particularmente muy altos.

Se puede hacer un gráfico de los residuos vs los valores ajustados como en 4.1 y 4.2.

¿Qué tan grande deben ser los residuos?

Solución: Se debe escalar los residuos adecuadamente.

Se construyen los residuos semi-estandarizados

$$r_i^s = \frac{e_i}{\sqrt{\text{Var}(e_i)}}$$

donde $e_i = Y_i - \hat{Y}_i$. 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

$$\begin{aligned}\text{Var}(e) &= \text{Var}((I - H)Y) \\ &= (I - H)^2 \text{Var}(Y) \\ &= (I - H)\sigma^2\end{aligned}$$

ya que $I - H$ es idempotente. Por lo tanto

$$\text{Var}(e_i) = (1 - h_{ii})\sigma^2$$

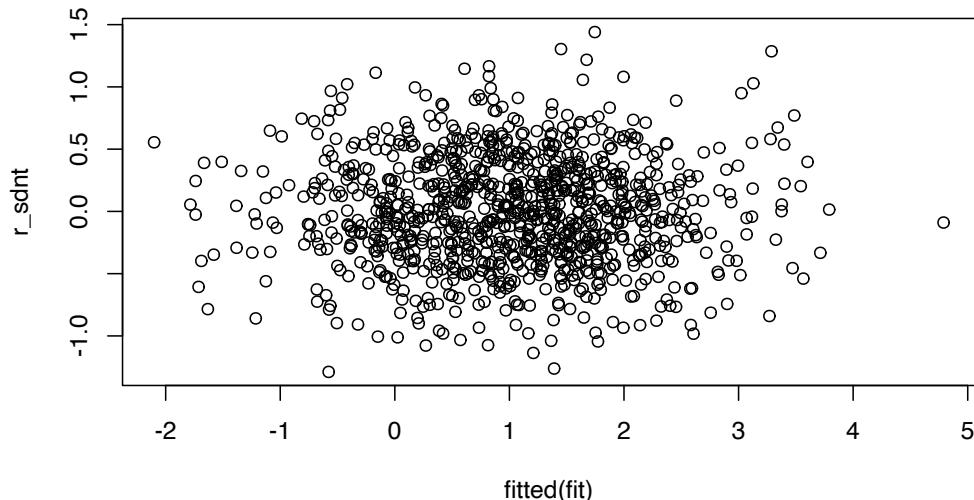
Para cada observación se estandarizan los residuos de siguiente forma

$$r_i^s = \frac{e_i}{\sqrt{(1 - h_{ii})\sigma^2}}$$

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

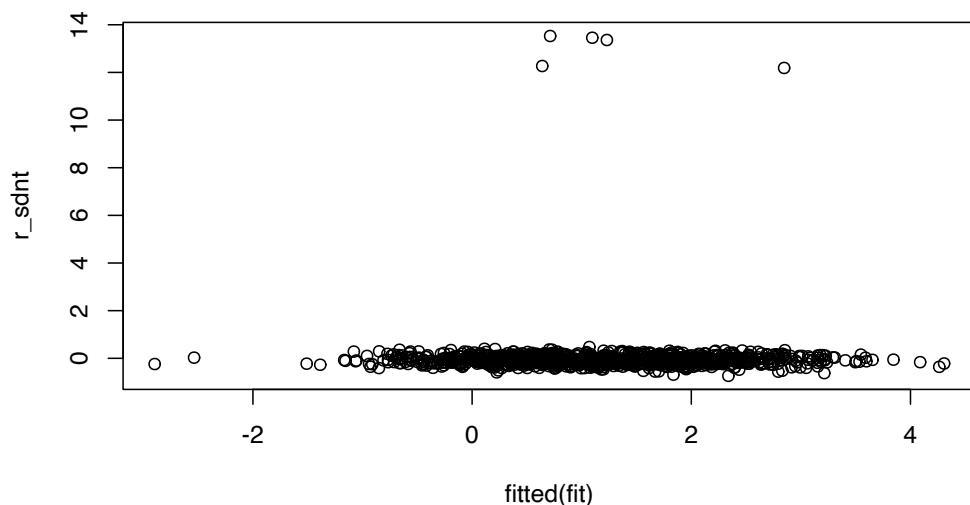
##
## Call:
```

```
## lm(formula = y ~ x)
##
## Coefficients:
## (Intercept)          x
##           0.9958      0.9682
```

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
##           1.1773      0.9884

```

4.7.2.2. 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**.

Como la matriz H es de idempotente y de rango completo:

$$\sum_{i=1}^n h_{ii} = p + 1 \text{ (Los predictores más el intercepto)}$$

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.

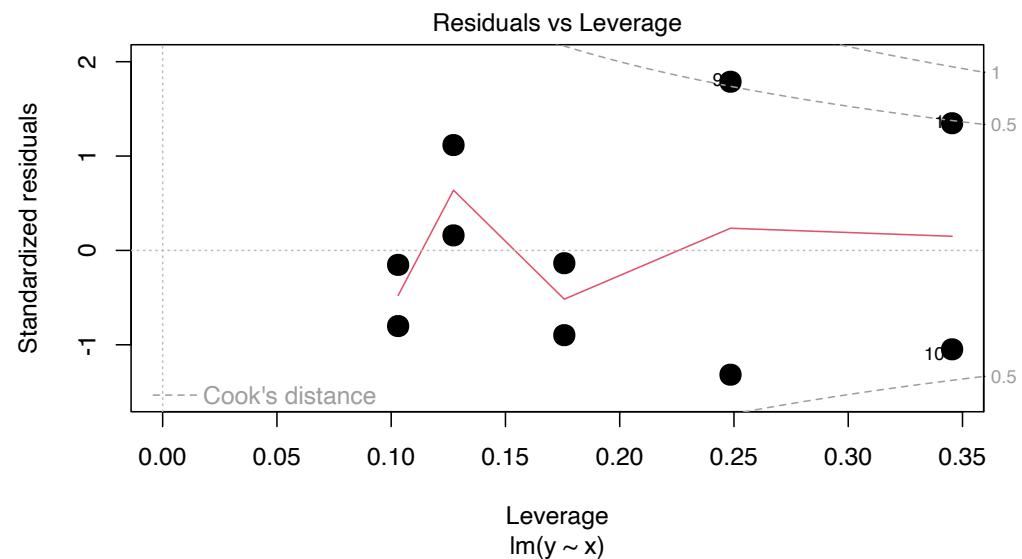
Caso base

```
set.seed(42)
apa_df = data.frame(x = 1:10, y = 10:1 + rnorm(n = 10))

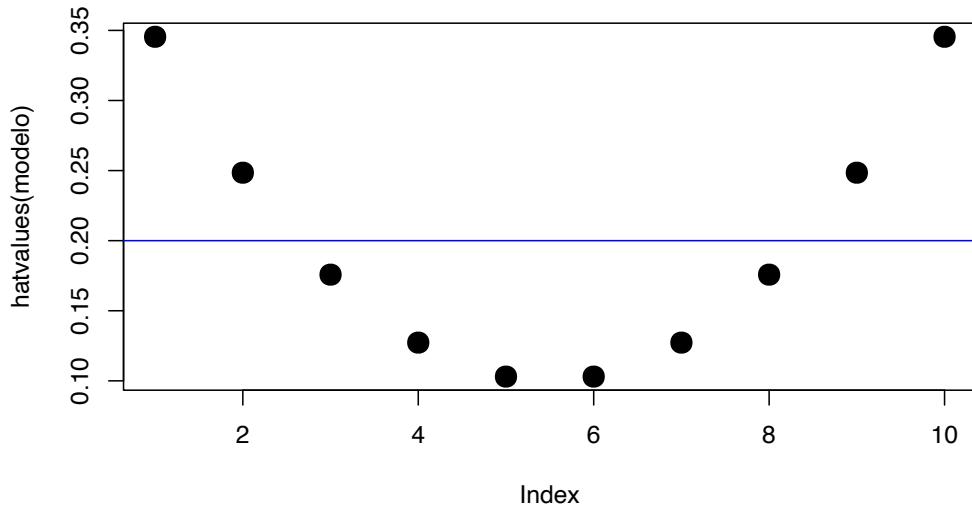
modelo <- lm(y ~ x, data = apa_df)
coef(modelo)

## (Intercept)           x
## 11.3801152 -0.9696033

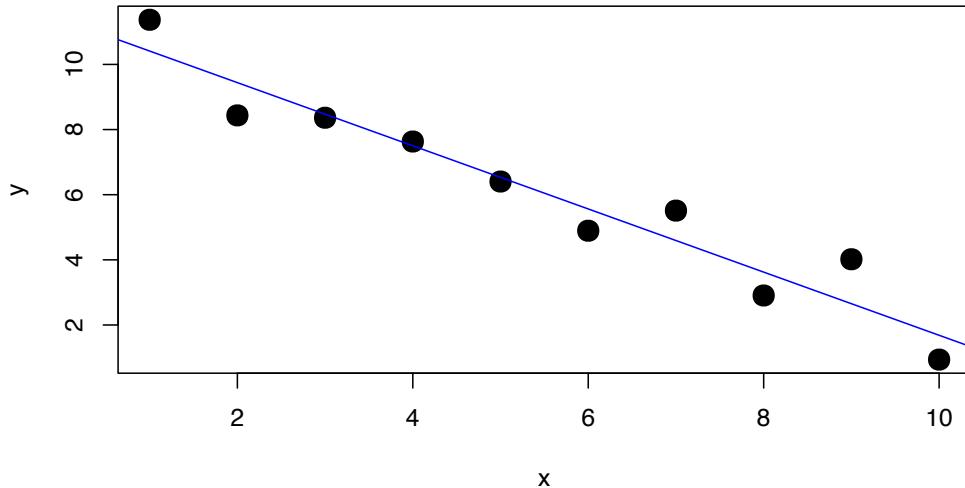
plot(modelo, 5, col = c(rep("black", 10), "red"), cex = 2,
     pch = 16)
```



```
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")
```

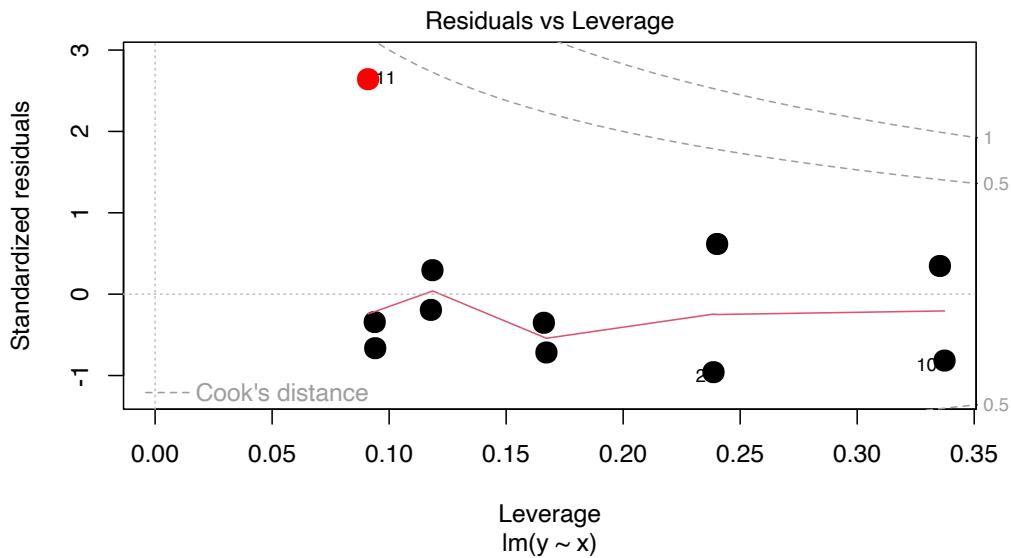


Bajo apalancamiento, residuos grandes, influencia pequeña

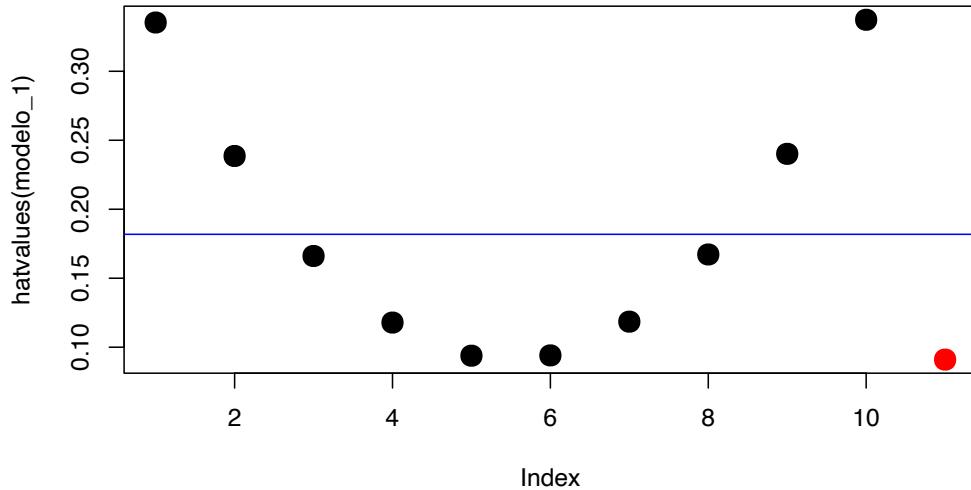
```
p_1 <- c(5.4, 11)
apa_df_1 <- rbind(apa_df, p_1)
modelo_1 <- lm(y ~ x, data = apa_df_1)
coef(modelo_1)
```

```
## (Intercept)          x
## 11.8509232 -0.9749534

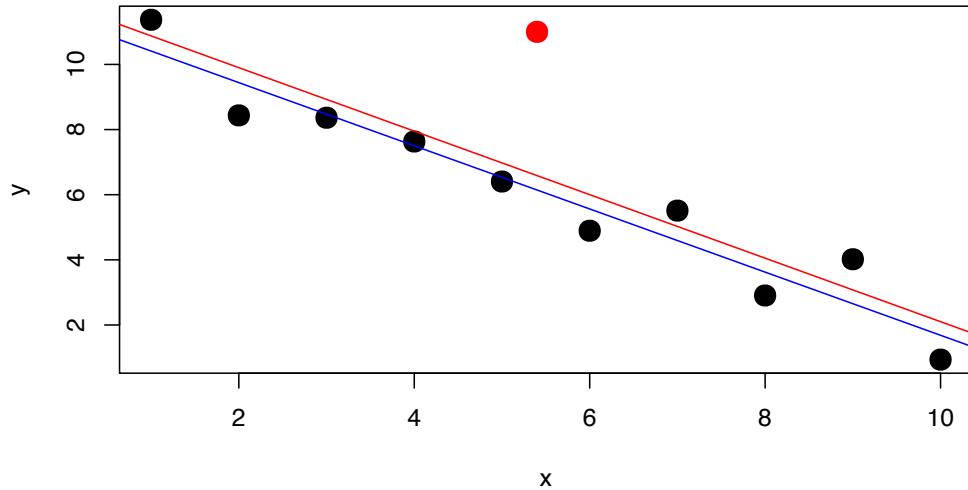
plot(modelo_1, 5, col = c(rep("black", 10), "red"),
     cex = 2, pch = 16)
```



```
plot(hatvalues(modelo_1), col = c(rep("black", 10),
  "red"), cex = 2, pch = 16)
abline(h = 2/11, col = "blue")
```



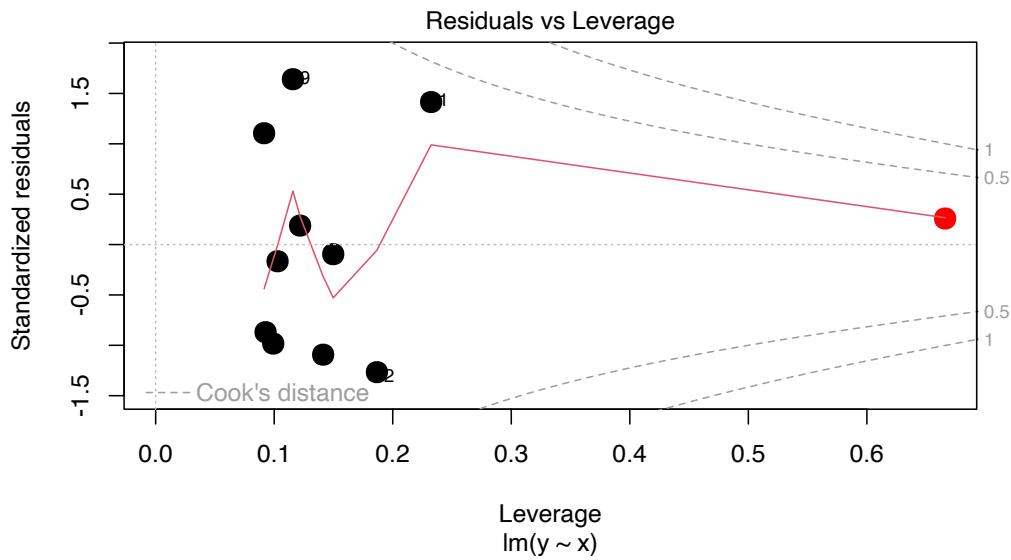
```
plot(apa_df_1, col = c(rep("black", 10), "red"), cex = 2,
     pch = 16)
abline(a = coef(modelo)[1], b = coef(modelo)[2], col = "blue")
abline(a = coef(modelo_1)[1], b = coef(modelo_1)[2],
       col = "red")
```



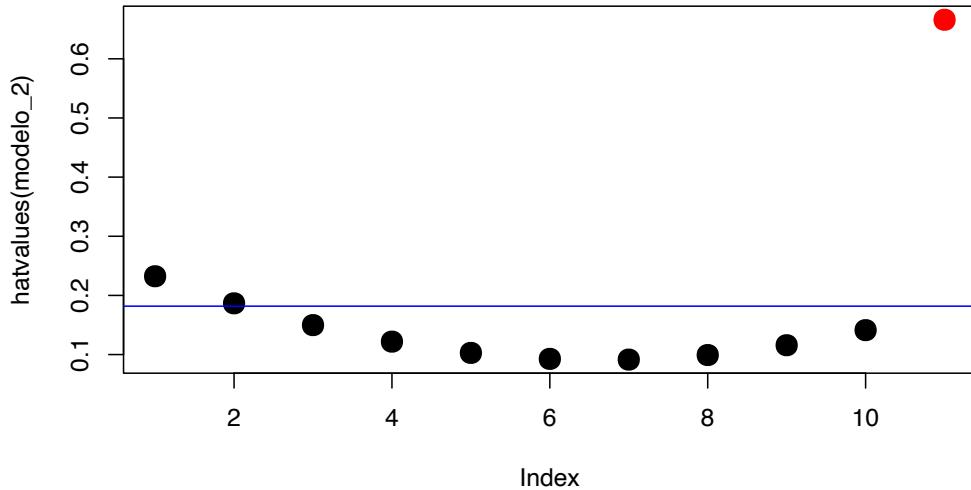
Alto apalancamiento, residuo pequeño, influencia pequeña

```
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)
```

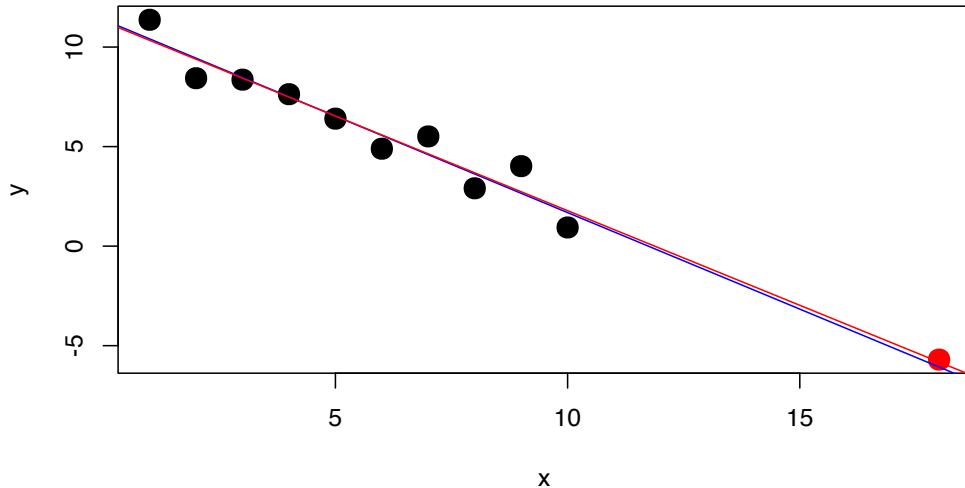
```
## (Intercept)          x
## 11.2888153 -0.9507397
plot(modelo_2, 5, col = c(rep("black", 10), "red"),
     cex = 2, pch = 16)
```



```
plot(hatvalues(modelo_2), col = c(rep("black", 10),
  "red"), cex = 2, pch = 16)
abline(h = 2/11, col = "blue")
```



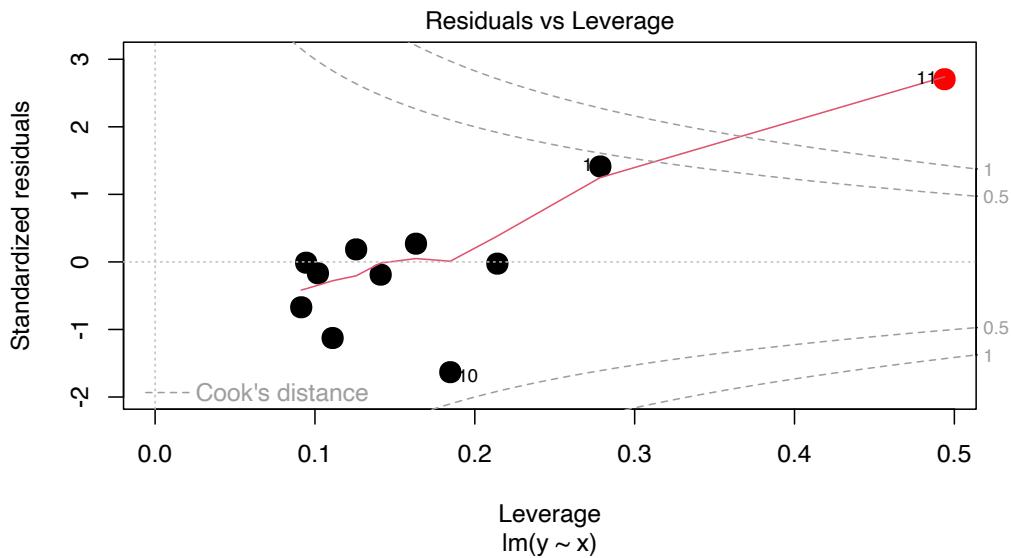
```
plot(apa_df_2, col = c(rep("black", 10), "red"), cex = 2,
      pch = 16)
abline(a = coef(modelo)[1], b = coef(modelo)[2], col = "blue")
abline(a = coef(modelo_2)[1], b = coef(modelo_2)[2],
       col = "red")
```



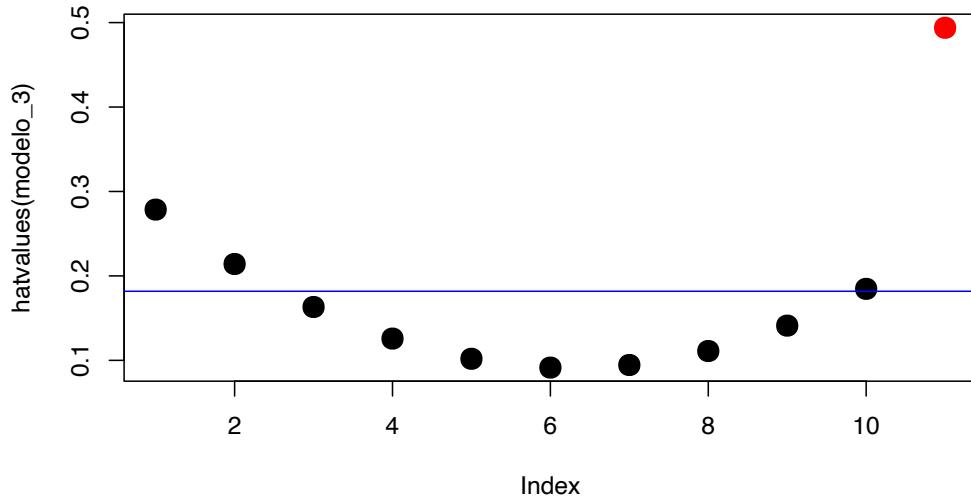
Alto apalancamiento, residuo alto, influencia grande

```
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)
```

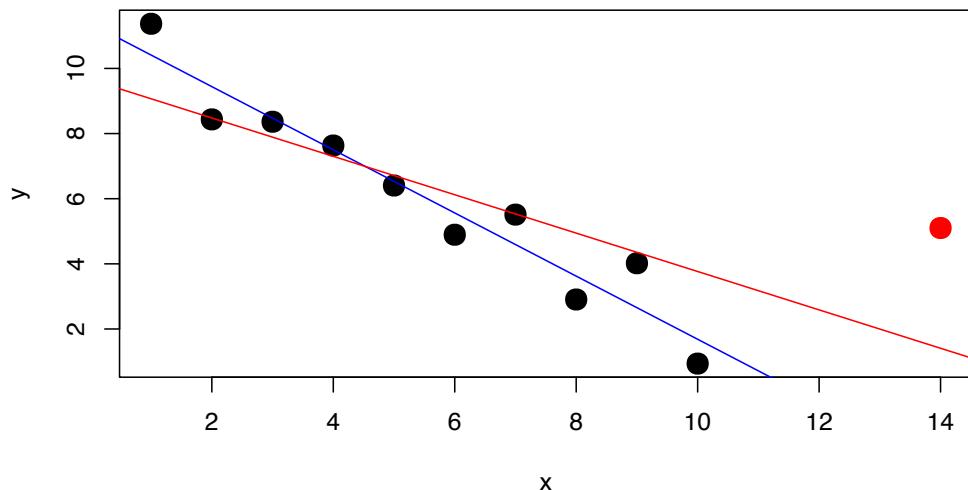
```
## (Intercept)          x
##   9.6572209  -0.5892241
plot(modelo_3, 5, col = c(rep("black", 10), "red"),
     cex = 2, pch = 16)
```



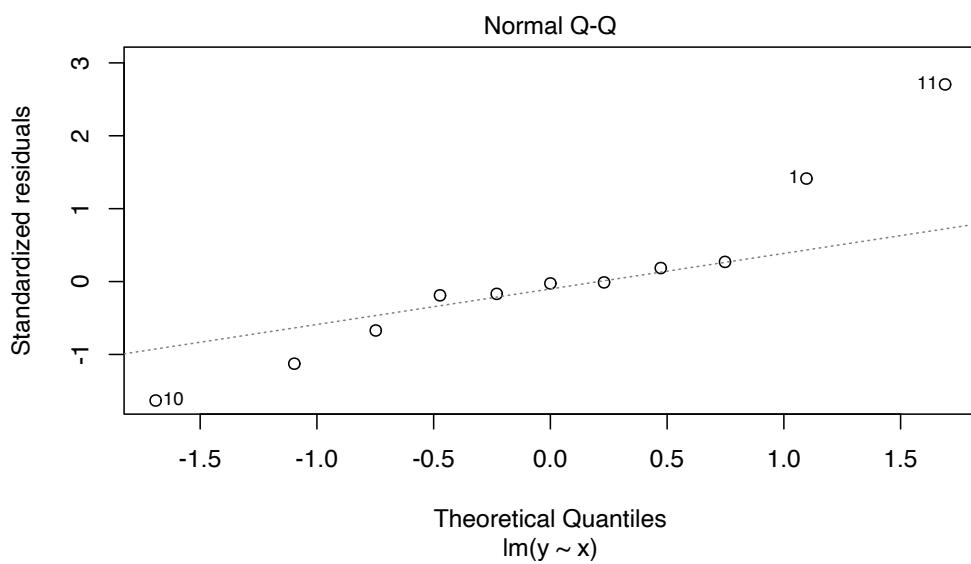
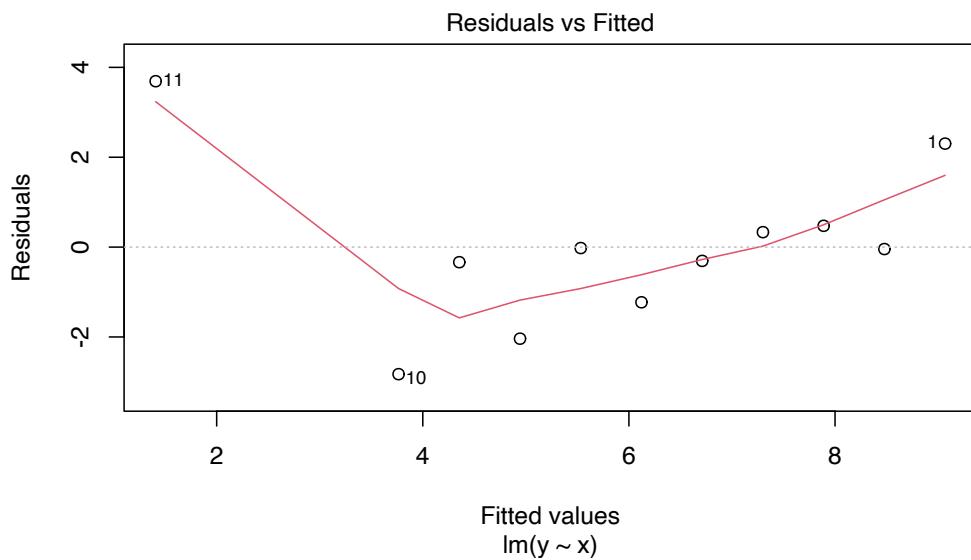
```
plot(hatvalues(modelo_3), col = c(rep("black", 10),
  "red"), cex = 2, pch = 16)
abline(h = 2/11, col = "blue")
```

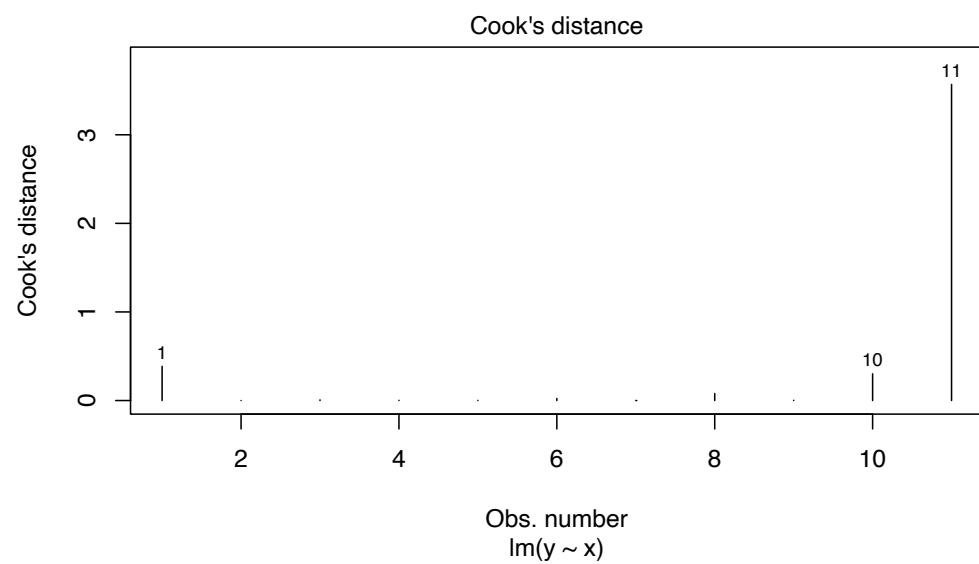
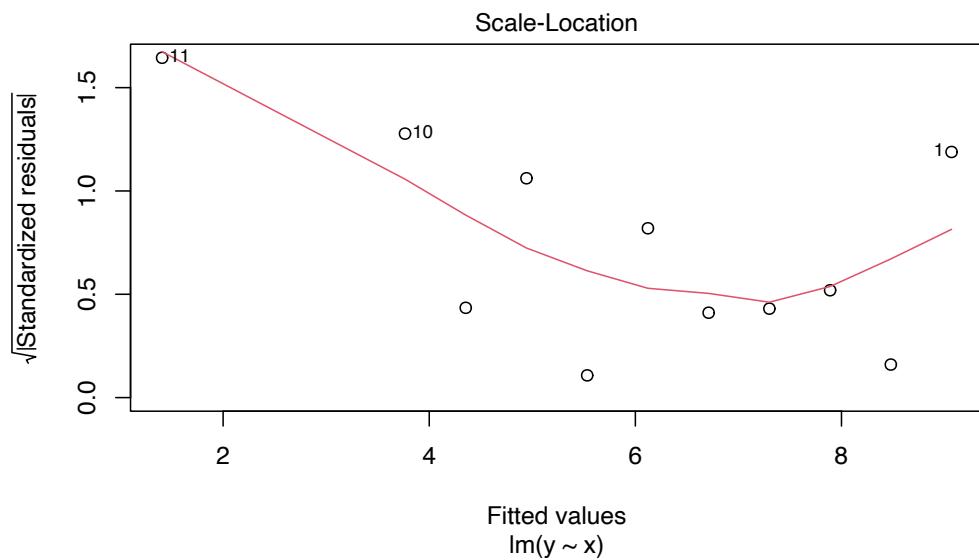


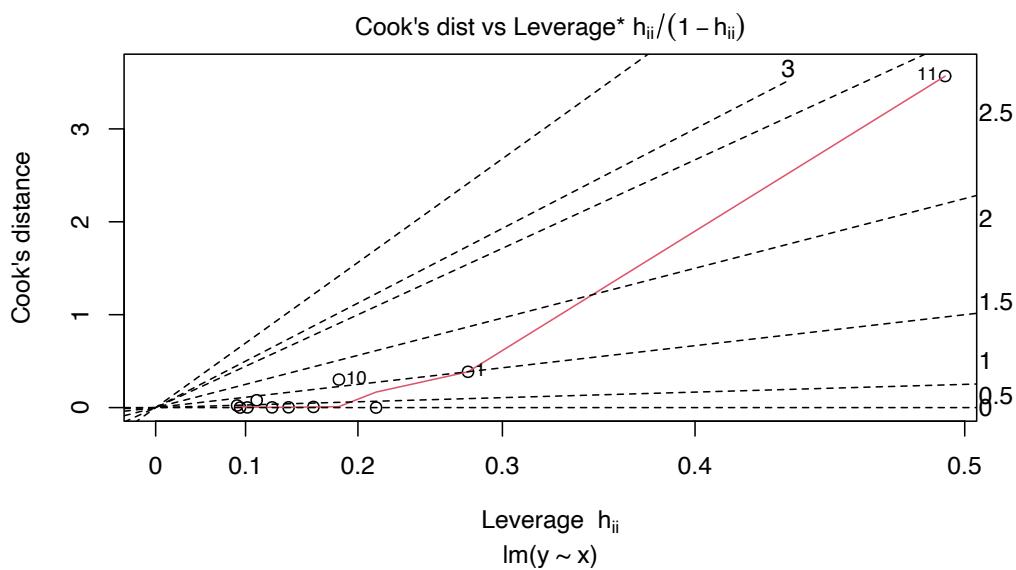
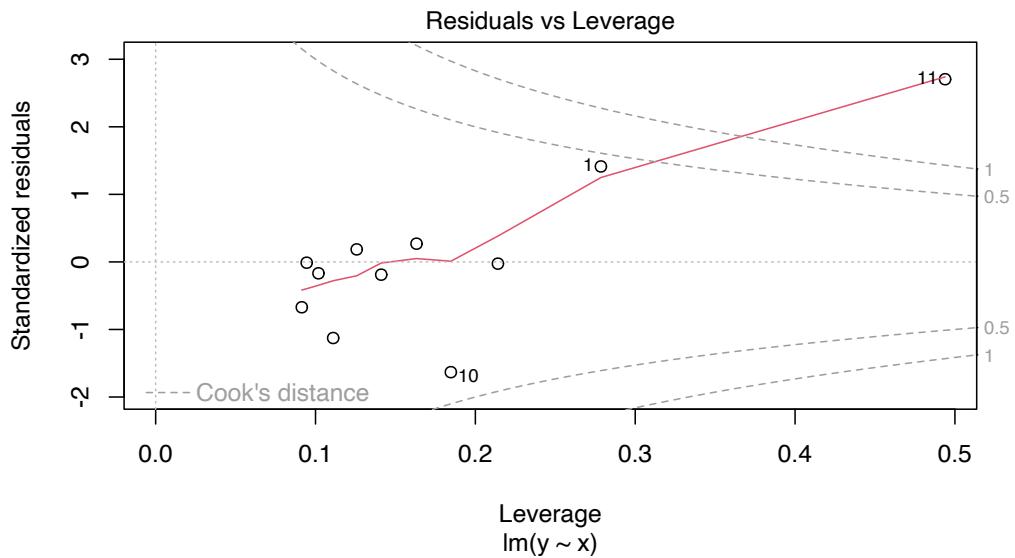
```
plot(apa_df_3, col = c(rep("black", 10), "red"), cex = 2,
      pch = 16)
abline(a = coef(modelo)[1], b = coef(modelo)[2], col = "blue")
abline(a = coef(modelo_3)[1], b = coef(modelo_3)[2],
       col = "red")
```



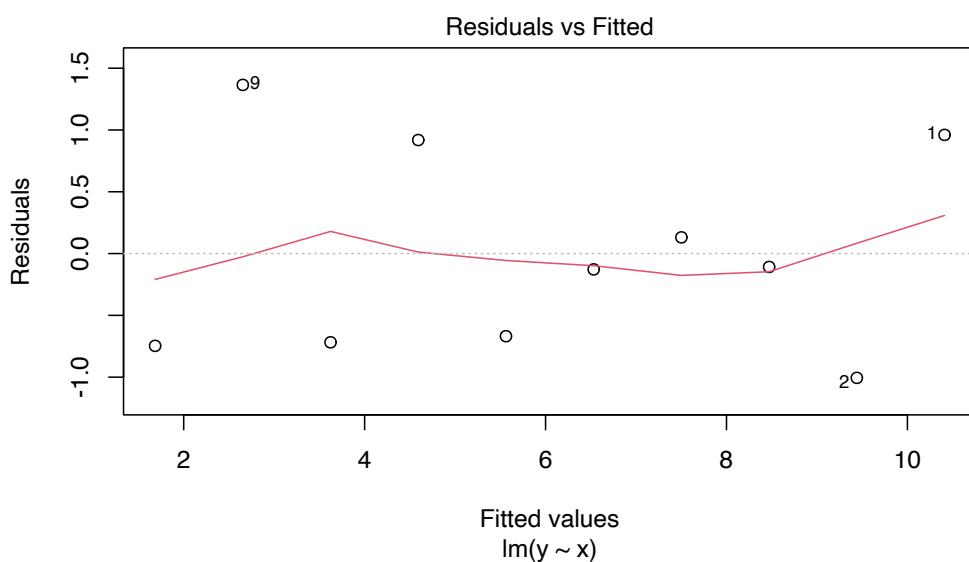
```
? (stats:::plot.lm)  
plot(modelo_3, which = 1:6)
```

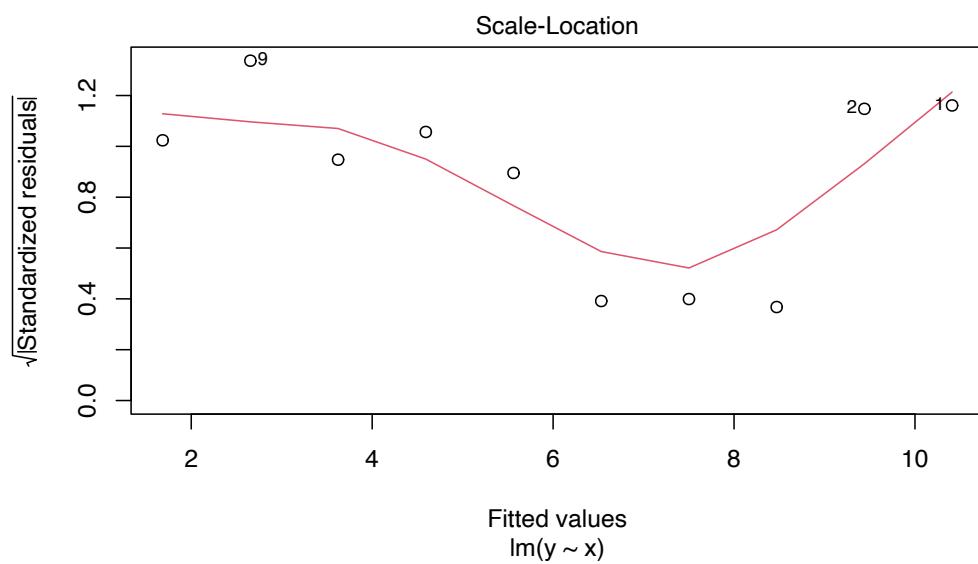
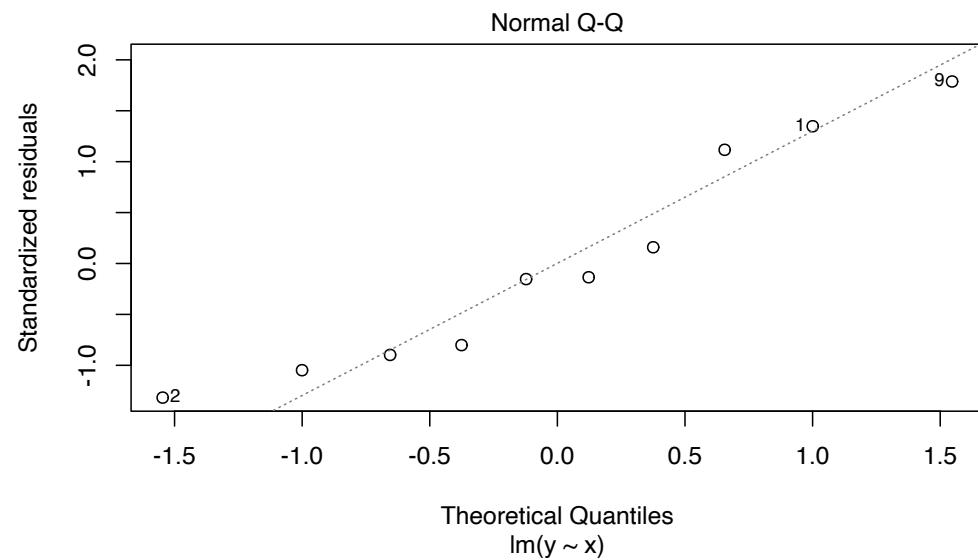


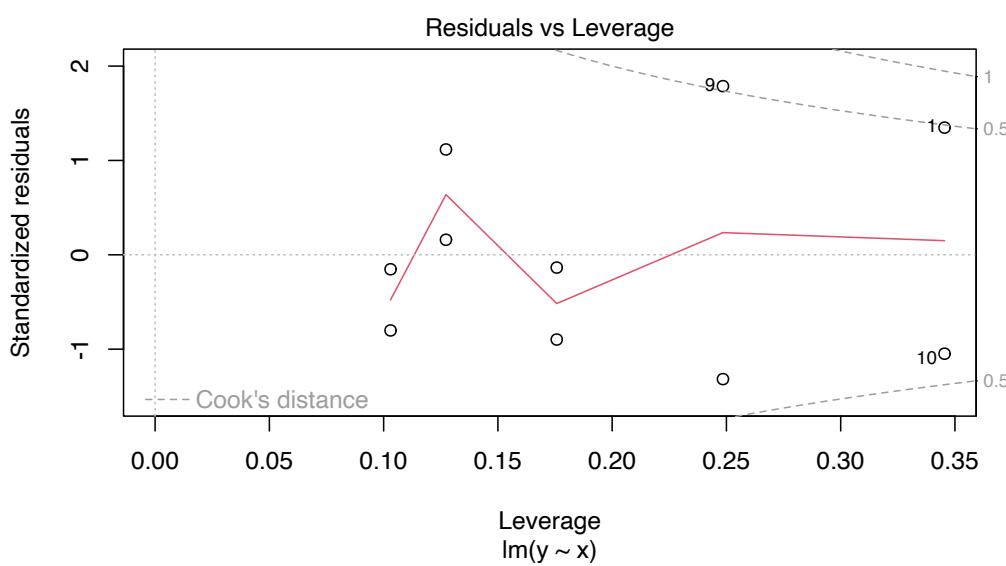
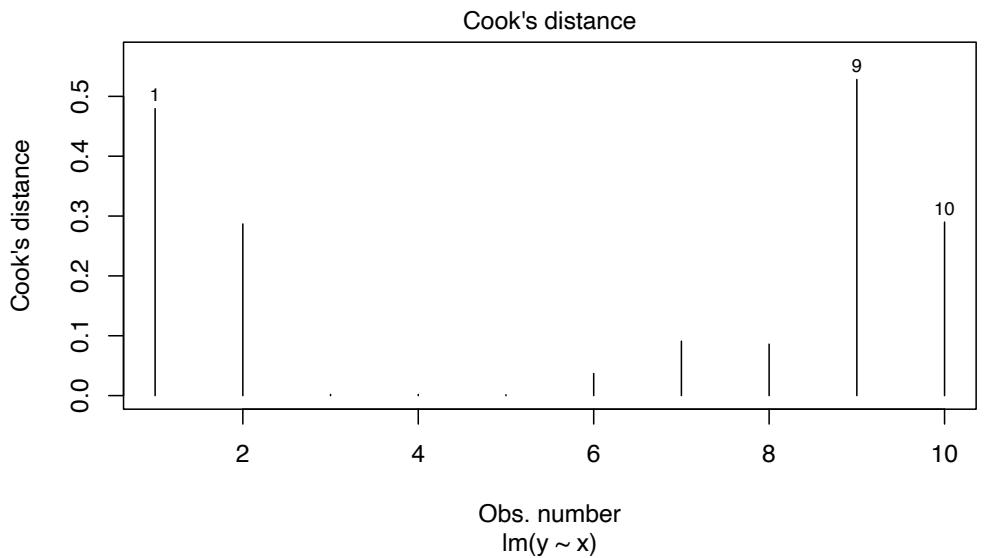


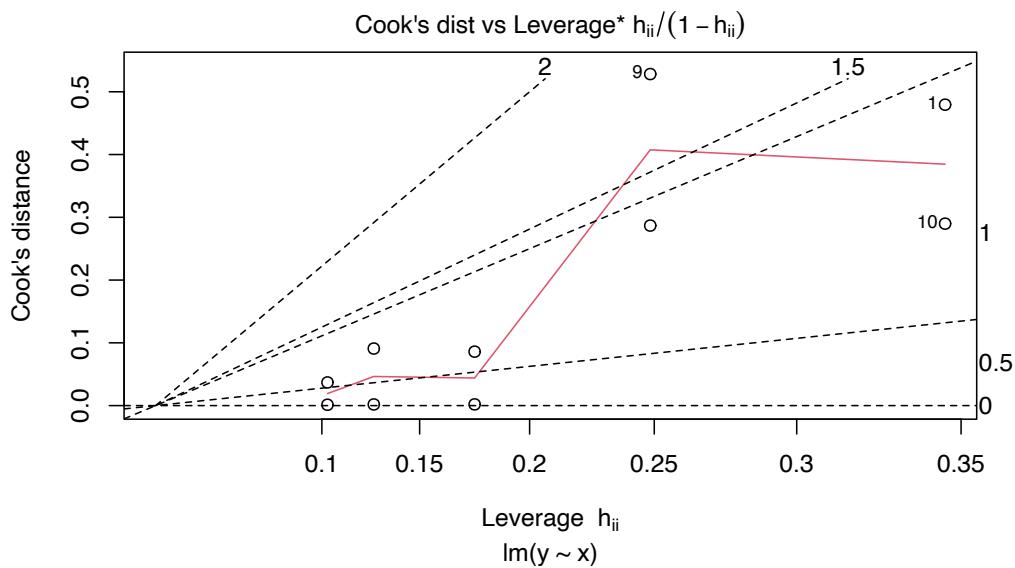


```
plot(modelo, which = 1:6)
```









4.8. Ejercicios

Del libro (James y col. 2013)

- Capítulo 3: 1, 3, 4, 5, 8, 9

Capítulo 5

Regresión Logística

5.1. Preliminares

Asuma que la variable dependiente 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: Cambiar Y por $g(Y) \in [0, 1]$, donde:

$$g(X) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p)}}$$

```
titanic <- read.csv("data/titanic.csv")
summary(titanic)

##   PassengerId      Survived       Pclass        Name
##   Min.    : 1.0   Min.    :0.0000   Min.    :1.000   Length:891
```

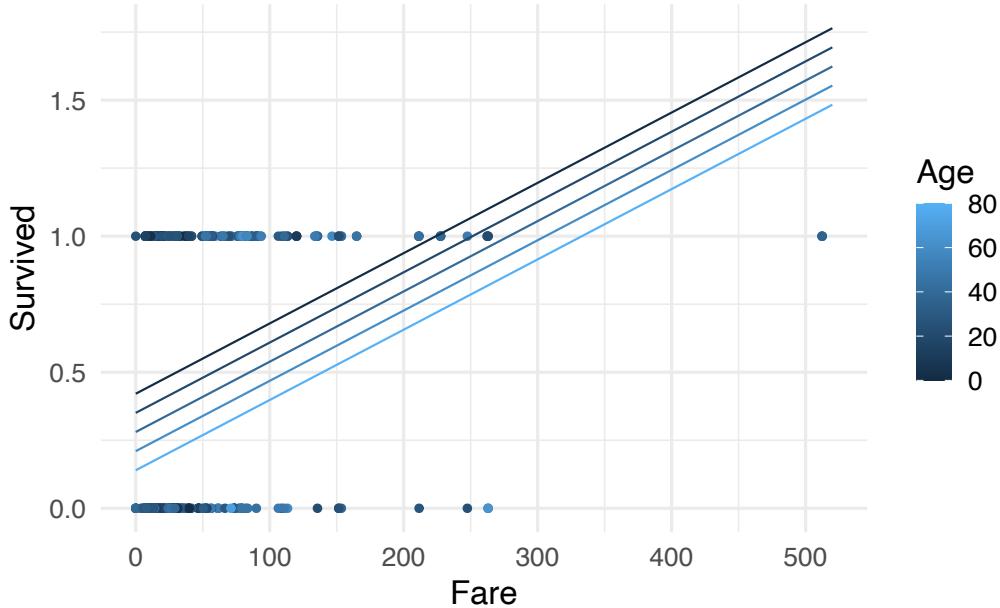
```

##   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()

## Error in select(., Survived, Fare, Age): unused arguments (Survived, Fare,
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`

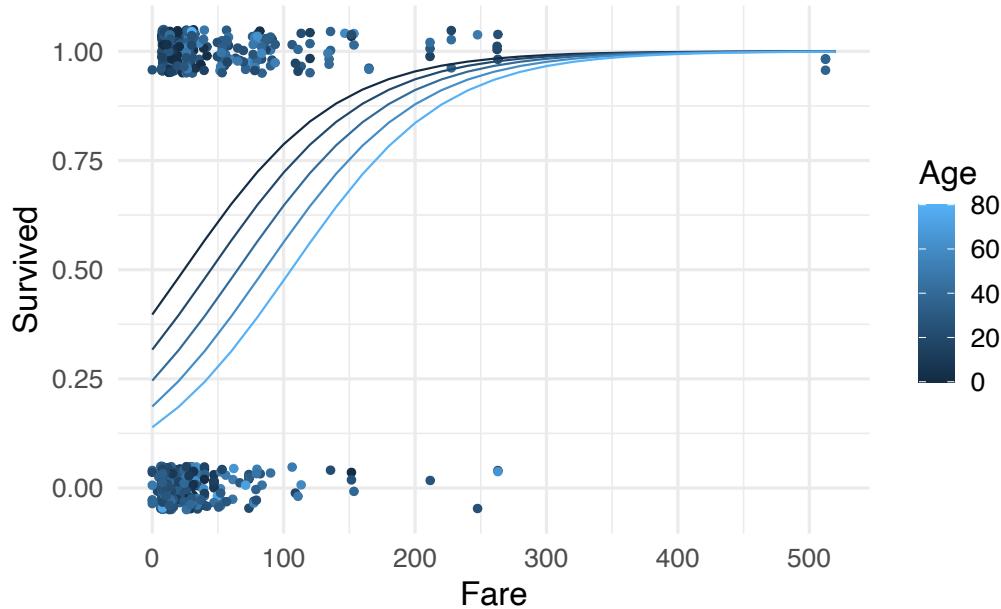
```
fit_glm <- glm(Survived ~ Fare + Age, data = titanic,
                 family = "binomial")
summary(fit_glm)
```

```
##
## 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
## (177 observations deleted due to missingness)
## 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".`

5.1.1. Oportunidad relativa (Odds Ratio)

Defina la oportunidad relativa:

$$O(X) = \frac{g(X)}{1 - g(X)} = e^{\beta_0 + \beta_1 X_1 + \dots + \beta_p X_p}.$$

como la razón de la probabilidad de obtener un 1 con respecto a la de obtener un 0.

Por ejemplo, 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.

Entonces la oportunidad relativa de pagar la tarjeta es $O(X) = \frac{0,8}{0,2} = \frac{4}{1}$, lo que se interpreta como que es 4 veces más probable de pagar que no pagar.

5.2. Máxima verosimilitud

Los valores de β se pueden encontrar por máxima verosimilitud.

Defina $p_\beta(\mathbf{X}) = \mathbb{P}(Y = 1 | \mathbf{X})$.

La verosimilitud es (donde asumimos sin pérdida de generalidad que $p(\mathbf{X}) := p_\beta(\mathbf{X})$):

$$L(\beta) = \prod_{i=1}^n p(\mathbf{X}_i)^{Y_i} (1 - p(\mathbf{X}_i))^{1-Y_i}$$

$$\begin{aligned}
\ell(\beta) &= \sum_{i=1}^n Y_i \log p(\mathbf{X}_i) + (1 - Y_i) \log (1 - p(\mathbf{X}_i)) \\
&= \sum_{i=1}^n \log (1 - p(\mathbf{X}_i)) + \sum_{i=1}^n Y_i \log \frac{p(\mathbf{X}_i)}{1 - p(\mathbf{X}_i)} \\
&= \sum_{i=1}^n \log (1 - p(\mathbf{X}_i)) + \sum_{i=1}^n Y_i (\mathbf{X}_i \cdot \beta) \\
&= \sum_{i=1}^n -\log (1 + e^{\mathbf{X}_i \cdot \beta}) + \sum_{i=1}^n Y_i (\mathbf{X}_i \cdot \beta)
\end{aligned}$$

$$\begin{aligned}
\frac{\partial \ell}{\partial \beta} &= -\sum_{i=1}^n \frac{1}{1 + e^{\mathbf{X}_i \cdot \beta}} e^{\mathbf{X}_i \cdot \beta} \mathbf{X}_i + \sum_{i=1}^n Y_i \mathbf{X}_i \\
&= \sum_{i=1}^n (Y_i - p(\mathbf{X}_i)) \mathbf{X}_i \\
&= X^\top (Y - p(\mathbf{X}))
\end{aligned}$$

Solución: Algoritmo de Netwon-Raphson.

Ejercicio 5.1. Muestre que

$$\frac{\partial^2 \ell}{\partial \beta^2} = -X^\top W X$$

donde $W_\beta = \text{diag}\{p(\mathbf{X}_i)(1 - p(\mathbf{X}_i))\}$.

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 5.2. Muestre que

$$\beta^{(t)} = (X^\top W_\beta X)^{-1} X^\top Z_\beta,$$

donde $Z_\beta = Z\beta + W_\beta^{-1}(Y - p(X))$ y $\beta = \beta^{(t-1)}$.

A esta técnica se le conoce como **mínimos cuadrados ponderados e iterados** o en inglés **Iteratively Re-Weighted Least Squares (IRLS)**.

5.2.1. Resultados adicionales

La suma al cuadrado de los residuos estandarizados se convierte en el estadístico de pearson:

$$\chi^2 = \sum_{i=1}^n \frac{(Y_i - \hat{p}(X_i))^2}{\hat{p}(X_i)}$$

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:

$$D_c - D_r \xrightarrow{H_0} \chi^2_{df_c - df_r}.$$

5.3. Diagnósticos del modelo

Advertencia: 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.

5.3.1. 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
## (177 observations deleted due to missingness)
## 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.predictores",
              -logit)

## Error in select(., Fare, Age): unused arguments (Fare, Age)

```

```
ggplot(df, aes(valores.predictores, logit)) + geom_point(size = 0.5,
  alpha = 0.5) + geom_smooth(method = "loess") +
  theme_bw() + facet_wrap(~predictores, scales = "free")
```

```
## Error in 'ggplot()':
## ! You're passing a function as global data.
## Have you misspelled the 'data' argument in 'ggplot()'
```

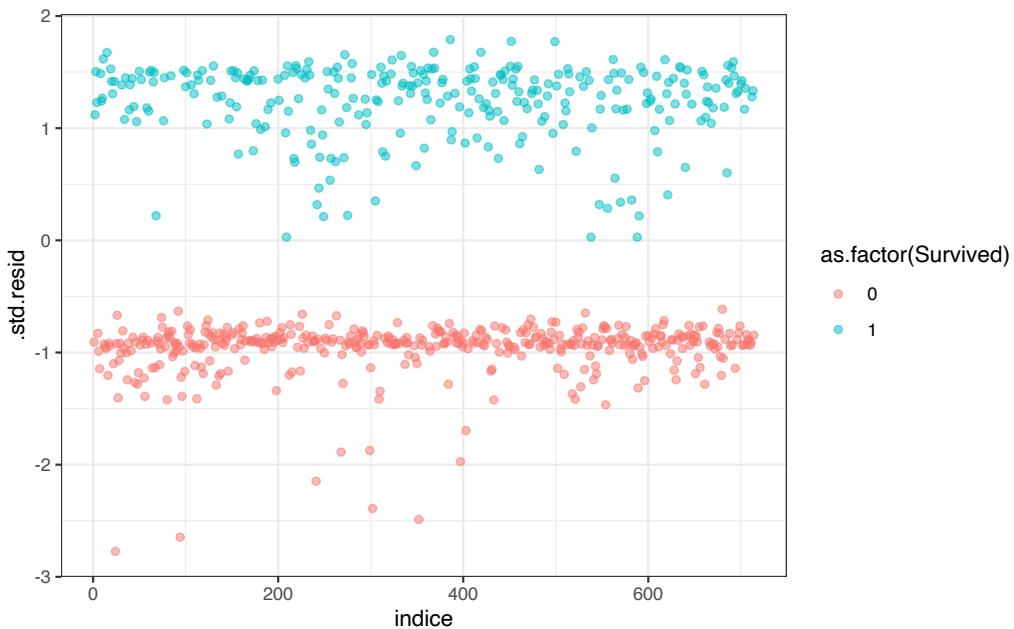
5.3.2 Valores de gran influencia

```
library(broom)
fit_data <- broom::augment(fit_glm) %>%
  mutate(indice = 1:n())

fit_data %>%
  top_n(3, .cooksdi)
```

```
## # A tibble: 3 x 11
##   .rownames Survived  Fare    Age .fitted .resid .std.resid     .hat .sigma
##   <chr>      <int> <dbl> <dbl>    <dbl>  <dbl>    <dbl> <dbl> <dbl>
## 1 28          0    263    19     3.79  -2.76    -2.77 0.00862  1.12
## 2 119         0    248.   24     3.43  -2.63    -2.65 0.0103   1.12
## 3 439         0    263    64     3.00  -2.47    -2.49 0.0171   1.12
## # ... with 2 more variables: .cooksdi <dbl>, 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: .rownames <chr>, Survived <int>, Fare <dbl>,
## #   Age <dbl>, .fitted <dbl>, .resid <dbl>, .std.resid <dbl>, .hat <dbl>,
## #   .sigma <dbl>, .cooksdi <dbl>, indice <int>
```

5.3.3. Multicolinealidad

```
car::vif(fit_glm)

##      Fare      Age
## 1.033878 1.033878
```

5.4. Predicción y poder de clasificación

La capacidad predictiva de un modelo de clasificación como el de regresión logística se debe medir conforme a la naturaleza de la variable dependiente. Primero recordemos que el modelo predictivo en este caso 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) para este conjunto de datos.

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

	Modelo = 0	Modelo = 1	
Real = 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)

## Error in table(titanic$Survived, predict_01): all arguments must have the same length
colnames(matriz_confusion) <- c("N", "P")

## Error in colnames(matriz_confusion) <- c("N", "P"): object 'matriz_confusion' not found
rownames(matriz_confusion) <- c("N", "P")

## Error in rownames(matriz_confusion) <- c("N", "P"): object 'matriz_confusion' not found

```

```
matriz_confusion
```

```
## Error in eval(expr, envir, enclos): object 'matriz_confusion' not found
```

Noten que se utilizó un umbral de .5 como separador de que un evento genera un 1 o un 0 a nivel de predicción. Para entender la siguiente tabla vamos a definir los siguientes términos:

Exactitud (Accuracy) Es la tasa de que un individuo esté bien identificado por el modelo de clasificación $(TP + TN)/(TP + TN + FN + FP)$.

Precisión Es la tasa de elementos identificados como 1 de forma correcta con respecto a los que fueron identificados con un valor de 1 $Precision = TP/P^*$

Sensibilidad (Exhaustividad) Es la tasa de elementos identificados como 1 de forma correcta con respecto a los que realmente son 1. $Sensitivity = TP/P$

F-Score Es la media armónica entre la precisión y la sensibilidad. $F-Score = 2 \times (Sensitivity * Precision) / (Sensitivity + Precision)$

Especificidad Es la tasa de elementos identificados correctamente como 0 que realmente estaban etiquetados como 0.

Entonces esto nos da las siguientes posibilidades.

Tipo	Cálculo	Sinónimos
Sensibilidad	TP/P	1 - Error tipo II, Poder, Exhaustividad, Exhaustividad.
Especificidad	TN/N	1- Error tipo I.
Valor de Predicción Positivos	TP/P^*	Positive predicted values (PPV), Precisión.
Valor de Predicción Negativos	TN/N^*	Negative predicted values (NPV)
F-Score	$\frac{2(TP/P^* \times TP/P)}{(TP/P^* + TP/P)}$	

Nota:

- La sensibilidad y especificidad son buenos indicadores cuando los datos son simétricos (igual número de FP y FN).
- El F-Score ayuda cuando los datos son asimétricos. El valor mayor de este valor es 1 indicado una precisión y exhaustividad perfectas.
- La sensibilidad nos permite describir la capacidad de categorizar los verdaderos positivos de forma correcta.
- En cambio, la especificidad lo hace para los verdaderos negativos.
- La precisión o PPV nos permite describir la capacidad del modelo de predecir verdaderos positivos.
- El NPV predice la capacidad del modelo de predecir los verdaderos negativos.

En un modelo se debe entender que significa cada uno de estos valores para entender los resultados.

- **Sensibilidad** es importante si la ocurrencia de **falsos negativos** es inaceptable. Supongamos que alguien tiene VIH y el examen le da negativo. En este caso es inaceptable el resultado, por lo que la prueba debe tener un alto valor de sensibilidad.
- **Especificidad** es importante para si la ocurrencia de **falsos positivos** es inaceptable. Es decir, supongamos que una persona tiene un tumor benigno y un examen lo clasifica como maligno. La persona sana, se declara enferma y debe pasar por un proceso de quimioterapia. La alta especificidad baja la probabilidad que esto ocurra.
- **PPV** es importante si se quiere estar más seguro de los **verdaderos positivos**. Por ejemplo detectar **spam** en correos electrónicos.
- **NPV** es importante si se desea predecir correctamente los **verdaderos negativos**. Por ejemplo, se está tratando una población con una droga experimental y se quiere saber cuál será la proporción de personas que se curaran de la enfermedad.

```
(TN <- matriz_confusion["N", "N"])
```

```
## Error in eval(expr, envir, enclos): object 'matriz_confusion' not found
(TP <- matriz_confusion["P", "P"])
```

```
## Error in eval(expr, envir, enclos): object 'matriz_confusion' not found
```

```
(FP <- matriz_confusion["N", "P"])

## Error in eval(expr, envir, enclos): object 'matriz_confusion' not found
(FN <- matriz_confusion["P", "N"])

## Error in eval(expr, envir, enclos): object 'matriz_confusion' not found
(exactitud <- (TP + TN)/(TP + TN + FP + FN))

## Error in eval(expr, envir, enclos): object 'TP' not found
(precision <- TP/(TP + FP))

## Error in eval(expr, envir, enclos): object 'TP' not found
(sensibilidad <- TP/(TP + FN))

## Error in eval(expr, envir, enclos): object 'TP' not found
(F_score <- 2 * (precision * sensibilidad)/(precision +
sensibilidad))

## Error in eval(expr, envir, enclos): object 'sensibilidad' not found
(especificidad <- TN/(TN + FP))

## Error in eval(expr, envir, enclos): object 'TN' not found
```

5.4.1. 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) grafica la Tasa Falsos Positivos vs Sensibilidad del modelo. Y el estadístico AUC mide el área bajo la curva ROC.

```
library(ROCR)

logist.pred.ROCR <- prediction(predict_numeric, titanic$Survived)
```

```
## Error in prediction(predict_numeric, titanic$Survived): Number of predictions in e
logist.perf <- performance(logist.pred.ROCR, "tpr",
  "fpr")
## Error in performance(logist.pred.ROCR, "tpr", "fpr"): object 'logist.pred.ROCR' no
plot(logist.perf)
## Error in h(simpleError(msg, call)): error in evaluating the argument 'x' in select
abline(0, 1, col = "red")
## Error in int_abline(a = a, b = b, h = h, v = v, untf = untf, ...): plot.new has no
auc <- performance(logist.pred.ROCR, measure = "auc")
## Error in performance(logist.pred.ROCR, measure = "auc"): object 'logist.pred.ROCR'
auc@y.values
## Error in eval(expr, envir, enclos): object 'auc' not found
```

Nota:

Hasta este momento estamos verificando el poder de clasificación del modelo con los mismos datos que usamos para ajustarlo. Es decir, le estamos diciendo al modelo que compruebe la veracidad de la clasificación que ya se hizo previamente.

Esto es incorrecto, ya que el modelo ya sabe “las respuestas” y no estamos midiendo su poder de clasificación sino más bien su capacidad predictiva dentro del conjunto de entrenamiento.

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 83  8
##      P 60 21

(TN <- matriz_confusion["N", "N"])

## [1] 83

(TP <- matriz_confusion["P", "P"])

## [1] 21

(FP <- matriz_confusion["N", "P"])

## [1] 8

(FN <- matriz_confusion["P", "N"])

## [1] 60

(exactitud <- (TP + TN)/(TP + TN + FP + FN))

## [1] 0.6046512

(precision <- TP/(TP + FP))

## [1] 0.7241379
```

```
(sensibilidad <- TP/(TP + FN))

## [1] 0.2592593

(F_score <- 2 * (precision * sensibilidad)/(precision +
sensibilidad))

## [1] 0.3818182

(especificidad <- TN/(TN + FP))

## [1] 0.9120879
```

Por defecto, la probabilidad para crear la matrix de confusión es 0.5. Si empezamos a variar esa probabilidad en el rango de 0 a 1, obtenemos la curva ROC.

```
logist.pred.ROCR <- prediction(predict_numeric, test$Survived)

## Error: 'predictions' contains NA.

logist.perf <- performance(logist.pred.ROCR, "tpr",
"fpr")

## Error in performance(logist.pred.ROCR, "tpr", "fpr"): object 'logist.pred.ROCR' no
plot(logist.perf)

## Error in h(simpleError(msg, call)): error in evaluating the argument 'x' in select
abline(0, 1, col = "red")

## Error in int_abline(a = a, b = b, h = h, v = v, untf = untf, ...): plot.new has no
auc <- performance(logist.pred.ROCR, measure = "auc")

## Error in performance(logist.pred.ROCR, measure = "auc"): object 'logist.pred.ROCR'
auc@y.values

## Error in eval(expr, envir, enclos): object 'auc' not found
```

5.5. 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 6

Métodos de selección de variables y regularización

6.1. Estimación del error de prueba

Recuerden que el error de prueba es el error promedio que se obtiene al usar un método de aprendizaje estadístico para predecir una observación que no está en el conjunto de calibración o entrenamiento. Si se tuviera un conjunto de prueba designado para medir el error de prueba, este último sería muy fácil de calcular. Cuando no se tiene un conjunto de prueba designado, entonces hay dos formas de estimar el error de prueba:

- a. Estimar indirectamente el error de prueba al hacerle un ajuste al error de entrenamiento. Es decir, agregarle artificialmente la variabilidad no observada, con el fin de corregir el exceso de sesgo producto de sobreajuste.
- b. Estimar el error de prueba usando técnicas de muestreo sobre el conjunto de entrenamiento.

Por ahora explicaremos la segunda solución:

6.1.1. Técnica de conjunto de validación

Dividir aleatoriamente los datos totales en 2 partes:

- Conjunto de entrenamiento: en donde se ajusta el modelo.

- Conjunto de validación o prueba: en donde el modelo ajustado se usa para realizar predicciones.

Se usa una medida de validación en cada uno de los dos conjuntos (MSE en el caso cuantitativo)

1. Puede que el estimador del error de prueba tenga una varianza muy alta, dependiendo de las observaciones que se incluyeron en la muestra de prueba.
2. El error de validación tiende a ser mayor que el error de entrenamiento debido a que se usa un número pequeño de observaciones en el conjunto de entrenamiento. (sobreestimación del error de prueba o validación)

6.1.2. 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. Este proceso se repite para $i = 1, \dots, n$.

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). Es decir el error de prueba no tiende a sobreestimarse.
2. No hay aleatoriedad en la aproximación del error de prueba producto de la separación conjunto de prueba-entrenamiento.

Desventaja: Puede ser lento, dependiendo de la cantidad de datos.

En el caso de un modelo lineal, se puede calcular CV_n en una sola iteración:

$$CV_n = \frac{1}{n} \sum_{i=1}^n \left(\frac{y_i - \hat{y}_i}{1 - h_{ii}} \right)^2$$

6.1.3. Validación cruzada k -veces

Se aplica el mismo principio que LOOCV, pero se divide la muestra en k distintas partes (folds). El estimador del error de prueba es:

$$CV_k = \frac{1}{k} \sum_{i=1}^k MSE_i$$

donde en este caso MSE_i es el MSE obtenido al utilizar la i -ésima parte de la muestra como conjunto de prueba y las restantes $k - 1$ partes como conjunto de entrenamiento. Los valores usuales de k son 5 o 10. Note que LOOCV es un caso especial de esta técnica de validación tomando $k = n$.

Ventaja: Es más económico

Desventaja: Nivel intermedio de sesgo respecto a las dos técnicas anteriores.

- Recuerden que entre más grande sea el conjunto de entrenamiento, menos sesgo hay en la estimación de error de prueba.
- El nivel de correlación entre los MSE's de los conjuntos de prueba aumenta si los tamaños de entrenamiento aumentan. Por lo tanto la varianza del error de prueba estimado es mayor en estos casos.

$$\frac{n}{2} < \frac{(k-1)n}{k} < n-1$$

6.1.4. 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 de manera análoga.

6.1.5. Otras medidas de error de prueba

6.1.5.1. R^2 ajustado

Recuerde que $R^2 = 1 - \frac{RSS}{TSS}$. Como RSS decrece si se le agrega más variables, entonces $R^2 \nearrow 1$. Recuerden que $RSS = \sum(y_i - \hat{y}_i)^2$ y $TSS = \sum(y_i - \bar{y}_i)^2$. En este caso definimos:

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

6.1.5.2. C_p de Mallows

$$C_p = \frac{1}{n} [RSS + 2p\hat{\sigma}^2]$$

donde p es el número de predictores y $\hat{\sigma}^2$ es el estimador de la varianza de los errores ϵ . Si $\hat{\sigma}^2$ es insesgado de σ^2 , entonces C_p es un estimador insesgado del MSE de prueba.

Definición alternativa:

$$C'_p = \frac{RSS}{\hat{\sigma}^2} + 2p - n \implies C_p = \frac{1}{n} \hat{\sigma}^2 [C'_p + n] \propto C'_p$$

6.1.5.3. Estimador de máxima verosimilitud (MLE)

Supongamos para el conjunto de datos disponible una medida de ajuste del modelo $2 \ln L(\hat{\beta}|x)$. Así si el valor es grande, entonces los parámetros β s serían los correctos. El problema con esta medida es que no penaliza el ingreso de nuevas variables al modelo.

6.1.5.4. Akaike Information Criterion (AIC).

El AIC se define de la siguiente manera:

$$AIC = -2 \log(L(\hat{\beta}|X)) + 2p$$

La idea del AIC es ajustar el riesgo empírico

Esta medida es derivada asintóticamente de $-2n\mathbb{E}[\log(L(\hat{\beta}|X))]$ cuando $n \rightarrow \infty$. En el caso gaussiano, el AIC se puede escribir como:

$$AIC = \frac{1}{n\hat{\sigma}^2}(RSS + 2p\hat{\sigma}^2)$$

- *BIC* (Bayesian Information Criterion).

$$BIC = -2 \log(L(\hat{\beta}|X)) + \log(n)p.$$

Representa asintóticamente hablando, el negativo del logaritmo de la distribución posterior. En el caso gaussiano:

$$BIC = \frac{1}{n}(RSS + 2 \log(n)p\hat{\sigma}^2).$$

6.1.5.5. C_p de Mallows

Este estadístico se define como

$$C_p = \frac{RSS}{\hat{\sigma}^2} + 2p - n$$

donde p es el número de predictores y $\hat{\sigma}^2$ es el estimador de la varianza de los errores ϵ . Si $\hat{\sigma}^2$ es insesgado de σ^2 , entonces C_p es un estimador insesgado del *MSE* de prueba.

Para regresión ordinaria sabemos que $\hat{\beta}_p = (\mathbf{X}'_p \mathbf{X}_p)^{-1} \mathbf{X}'_p \mathbf{Y}$. Nuestro caso ideal sería hacer el MSE lo más pequeño posible entre todos los posibles modelos,

$$\mathbb{E} [\hat{\beta}_p - \beta]^2.$$

Con esto en mente, calculemos el RSS del modelo,

$$\begin{aligned}
\text{RSS}(p) &= \sum_{n=1}^N (y_n - \mathbf{x}_n \hat{\boldsymbol{\beta}}_p)^2 \\
&= (\mathbf{Y} - \mathbf{X}_p \hat{\boldsymbol{\beta}}_p)' (\mathbf{Y} - \mathbf{X}_p \hat{\boldsymbol{\beta}}_p) \\
&= \mathbf{Y}' \left(\mathbf{I}_N - \mathbf{X}_p (\mathbf{X}'_p \mathbf{X}_p)^{-1} \mathbf{X}'_p \right) \mathbf{Y}
\end{aligned}$$

Usando este resultado para matrices

$$\mathbb{E} [\mathbf{Y}' \mathbf{A} \mathbf{Y}] = \mathbb{E} [\mathbf{Y}'] \mathbf{A} \mathbb{E} [\mathbf{Y}] + \text{tr}[\Sigma \mathbf{A}]$$

y donde Σ es la matriz de covarianza de \mathbf{Y} , encontramos que

$$\begin{aligned}
\mathbb{E}[\text{RSS}(p)] &= \mathbb{E} \left[\mathbf{Y}' \left(\mathbf{I}_N - \mathbf{X}_p (\mathbf{X}'_p \mathbf{X}_p)^{-1} \mathbf{X}'_p \right) \mathbf{Y} \right] \\
&= \mathbb{E} [\hat{\boldsymbol{\beta}}_p - \boldsymbol{\beta}]^2 + \text{tr} \left[\mathbf{I}_N - \mathbf{X}_p (\mathbf{X}'_p \mathbf{X}_p)^{-1} \mathbf{X}'_p \right] \sigma^2 \\
&= \mathbb{E} [\hat{\boldsymbol{\beta}}_p - \boldsymbol{\beta}]^2 + \sigma^2 \left(N - \text{tr} \left[(\mathbf{X}'_p \mathbf{X}_p) (\mathbf{X}'_p \mathbf{X}_p)^{-1} \right] \right) \\
&= \mathbb{E} [\hat{\boldsymbol{\beta}}_p - \boldsymbol{\beta}]^2 + \sigma^2(N - p)
\end{aligned}$$

Note que si el modelo verdadero tiene p parámetros, entonces $\mathbb{E} [C_p] = p$. Esto muestra por qué, si un modelo es correcto, C_p tenderá a estar cerca de p .

Un problema con el criterio C_p es que tenemos que encontrar una estimación apropiada de σ^2 para usar con todos los valores de p .

6.1.5.6. Estimador de máxima verosimilitud (MLE)

Supongamos para el conjunto de datos disponible una medida de ajuste del modelo $-\ln L(\hat{\boldsymbol{\beta}}|x) = \ell(\boldsymbol{\beta})$. Así si el valor es grande, entonces los parámetros $\boldsymbol{\beta}$ s serían los correctos. Si se define $\bar{\ell}(\boldsymbol{\beta}) = \mathbb{E}(\ell(\boldsymbol{\beta} | X))$ como la log verosimilitud poblacional.

El problema con esta medida es que no penaliza el ingreso de nuevas variables al modelo.

6.1.5.7. Akaike Information Criterion (AIC).

El AIC se define de la siguiente manera:

$$AIC = -2\ell(\hat{\beta}) + 2p$$

Explicación breve: La idea del AIC es ajustar el riesgo empírico $\hat{R}_n = -\ell(\hat{\beta})$ con respecto al riesgo verdadero $R(\hat{\beta}) = \mathbb{E}(-n\bar{\ell}(\hat{\beta}))$. Se puede probar que asintóticamente

$$\mathbb{E}(\hat{R}_n(\hat{\beta}_n)) - R(\hat{\beta}_n) = -n\mathbb{E}\left(\left(\hat{\beta}_n - \beta^*\right)^T I(\beta^*) \left(\hat{\beta}_n - \beta^*\right)\right)$$

Por el curso de estadística I se puede probar que

$$\sqrt{n}(\hat{\beta}_n - \beta^*) \approx N(0, I^{-1}(\beta^*))$$

Por lo tanto,

$$n(\hat{\beta}_n - \beta^*)^T I(\beta^*) (\hat{\beta}_n - \beta^*) \approx \chi_d^2$$

lo cual implica que

$$n\mathbb{E}\left(\left(\hat{\beta}_n - \beta^*\right)^T I(\beta^*) \left(\hat{\beta}_n - \beta^*\right)\right) = d$$

Entonces para asegurarnos de que el estimador sea insesgado asintóticamente, debemos redefinir nuestro riesgo empírico estimador sumándole un término p

$$\hat{R}_n(\hat{\beta}_n) + d = -\ell_n + d$$

6.1.5.8. Bayesian Information Criterion (BIC)

Este criterio se parece al AIC pero modificado con un $\log(n)$,

$$BIC = -2\ell(\hat{\beta}) + \log(n)p.$$

Para cada modelo m Escriba la regla de Bayes definida en el curso anterior

$$\pi(m | X_1, \dots, X_n) = \frac{\pi(m, X_1, \dots, X_n)}{L(X_1, \dots, X_n)} \propto L(X_1, \dots, X_n | m) \pi(m)$$

Se puede demostrar que

$$L(X_1, \dots, X_n | \theta, m) \approx e^{\ell_n - n(\theta - \theta^*)^T I(\theta^*)(\theta - \theta^*)}$$

Asumiendo que la variable respuesta Y es gaussiana, se puede simplificar la log-verosimilitud como

$$\log p(X_1, \dots, X_n | m) \approx \ell_n - \frac{d}{2} \log n + \frac{d}{2} \log(2\pi) + \log \det(I(\theta^*)) + \log \pi(\hat{\theta}_n | m)$$

Las únicas dos cantidades que incrementan con n son las dos primeras. Se multiplica por -2 para tener el BIC.

En el caso de datos gaussiano se tendría lo siguiente:

$$BIC = \frac{1}{n}(RSS + 2 \log(n)p\hat{\sigma}^2).$$

6.1.5.9. Notas adicionales

- 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).
- La validación cruzada LOOCV es asintóticamente equivalente al AIC para modelos de regresión lineal múltiple (Stone 1977).
- El AIC ajusta el modo que el riesgo o verosimilitud empírica o real sean insesgadas. Es decir, bajo la observación de nuevos datos, el error que se cometería debería ser cercano a 0.

- Aunque el BIC se parece al AIC, el razonamiento es algo diferente. En la construcción de BIC estamos seleccionando el modelo con mayor evidencia según los datos. Cuando los datos se generan de hecho a partir de uno de los modelos en la colección de modelos que estamos eligiendo, el posterior se concentrará en este modelo correcto.
- Una forma de interpretar ambos criterios es que AIC elige el mejor modelo predictivo, mientras que BIC intenta seleccionar el modelo verdadero si existe en el conjunto de modelos.

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).

6.2. Selección de variables

Cuando se construye un modelo de regresión (lineal o logística) existe la posibilidad de que existan más variables que datos disponibles. En este caso definitivamente la matriz de diseño X no sería de rango completo, y otros estadísticos no tendrían una definición clara, por ejemplo 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 simples y cómo hacer comparaciones entre ellos.

6.2.1. 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 un total de $\binom{p}{k}$ modelos diferentes que escoger. Por lo tanto existe un total de 2^p posibles modelos para escoger con cualquier número de covariables.

El algoritmo para este caso sería:

Algoritmo:

1. Sea M_0 el modelo nulo.
2. 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 .
3. Seleccione el mejor modelo entre M_0, M_1, \dots, M_p usando el 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. Esto puede provocar sobreajuste.

- El principal inconveniente de este método es que es computacionalmente ineficiente cuando p es relativamente grande.

6.2.2. Selección de modelos hacia adelante (Forward Stepwise Selection)

Algoritmo:

1. Sea M_0 el modelo nulo.
2. Para $k = 0, 1, \dots, p - 1$,
 - a. Considera los $p - k$ modelos que contenga los predictores en M_k con un predictor adicional.
 - b. Selecciona el mejor entre esos $p - k$ modelos usando el R^2 o RSS . Llámalo M_{k+1} .
3. Selecciona el mejor modelo entre M_0, \dots, M_p usando CV , C_p , AIC , BIC o R^2 ajustado.

Ejemplo: $Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_3$

- $M_0: Y = \beta_0$
- $M_1: Y = \beta_0 + \beta_1 X_1$, $Y = \beta_0 + \beta_2 X_2$ o $Y = \beta_0 + \beta_3 X_3$. De los tres se escoge el mejor (por ejemplo, el 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 el mejor.
- $M_3: M_2$ más la variable no incluida.

Nota:

- El número de modelos por calcular usando el mejor subconjunto es 2^p , mientras que usando Forward Selection es $1 + \sum_0^{p-1} p - k = \frac{1 + p(1 + p)}{2}$.
- No hay garantía de que el modelo seleccionado con este método sea el mejor de todos los posibles.

6.2.3. Selección de modelos hacia atrás (Backward Stepwise Selection)

Algoritmo:

1. Sea M_p el modelo completo.
2. Para $k = p, p-1, \dots, 1$,
 - a. Considere los k modelos que contienen todos excepto uno de los predictores en M_k para un total de $k-1$ predictores.
 - b. Seleccione el mejor entre esos k modelos usando el R^2 o RSS . Llámelo M_{k+1} .
3. Seleccione el mejor modelo entre M_0, \dots, M_p usando CV , 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.

Las dos notas del método de Forward Selection aplican para este caso también.

6.3. Métodos de regularización

Una forma alternativa de hacer selección de variables es a través de la restricción o regularización de los coeficientes de los modelos.

6.3.1. Regresión Ridge

Considere el problema de mínimos cuadrados en el modelo lineal:

$$RSS = \sum_{i=1}^n \left(y_i - \sum_{j=1}^p \beta_j X_{ij} \right)^2$$

y

$$\hat{\beta} = \underset{\beta}{\operatorname{argmin}} RSS$$

La regresión Ridge consiste en determinar

$$\hat{\beta}_\lambda^R = \operatorname{argmin}_{\beta} [RSS + \lambda \|\beta_{-0}\|_2^2]$$

donde:

$$\|\beta_{-0}\|_2^2 = \sum_{j=1}^p \beta_j^2$$

¿Cómo se debe seleccionar el λ ? El método para seleccionarlo es por validación cruzada

Nota:

- 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.
- Cuando p es cercano a n , la varianza a nivel de parámetros es alta (matriz X deja de comportarse como de rango completo). En ese caso la regresión ridge se convierte en una solución para disminuir esa varianza.
- Si $p > n$ (mayor cantidad de variables que observaciones), al realizar mínimos cuadrados, no existe una solución única, pero con la forma de regresión de Ridge es posible alcanzarla para un cierto valor de λ .
- La regresión ridge tiene ventajas computacionales comparado con el método de selección de todos los subconjuntos.
- Los estimadores de ridge no son invariantes a cambios de escala (transformaciones cX_i), a diferencia de los estimadores por mínimos cuadrados. Por lo tanto se recomienda estandarizar todas las predictores antes de aplicar ridge.

6.3.2. Regresión Lasso

La regresión ridge tiene el inconveniente de que incluye todas los predictores en el modelo, lo cual puede ser un inconveniente para efectos interpretativos (de parámetros) pero no tanto para efectos predictivos, especialmente cuando el número de predictores es alto. Para solucionar este problema se plantea:

$$\beta_\lambda^{LASSO} = \operatorname{argmin}_{\beta} (RSS + \lambda \|\beta_{-0}\|_1)$$

donde:

$$\|\beta_{-0}\|_1 = \sum_{j=1}^p |\beta_j|$$

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 $\sum_{j=1}^p |\beta_j| \leq s$.

Nota: la regresión lasso causa una selección de predictores debido a que RSS suele intersecar la región de penalización en esquinas.

6.3.3. Explicación gráfica

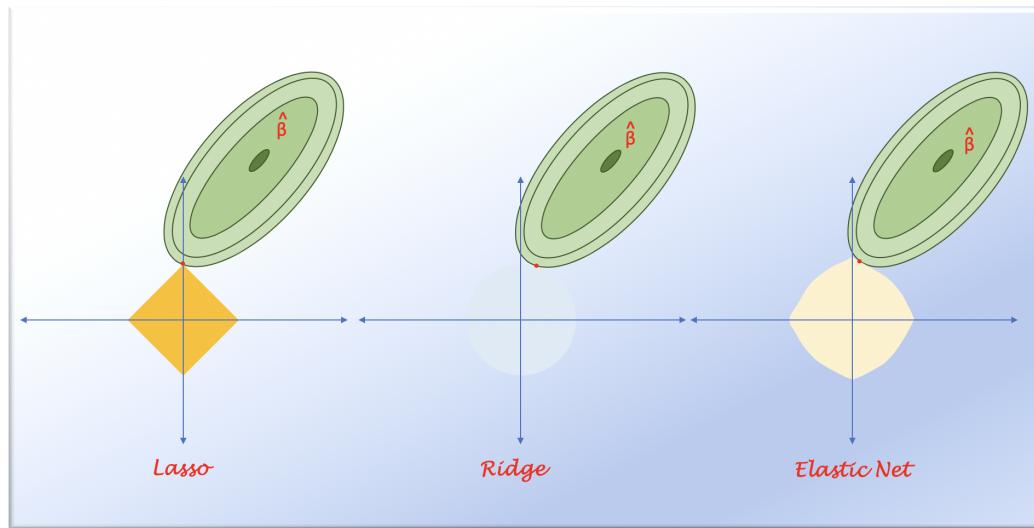
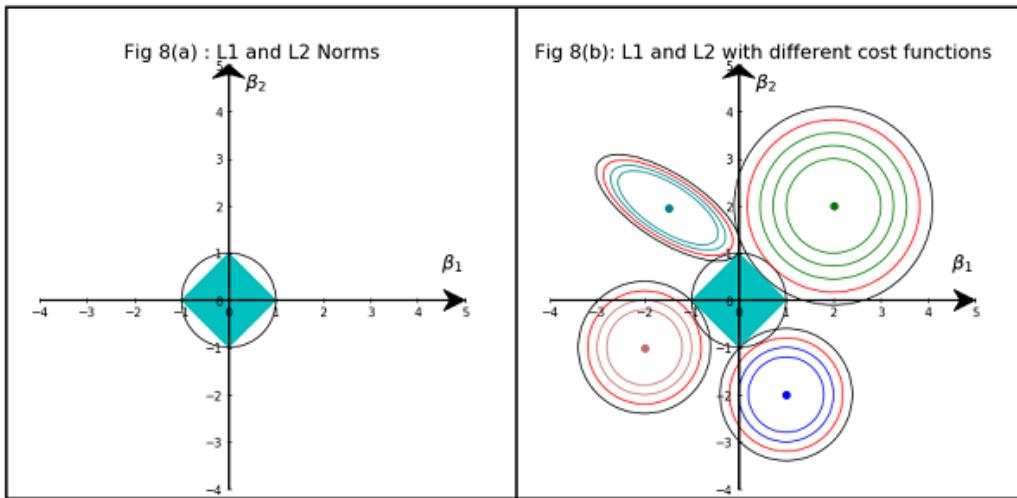


Figura 6.1: Tomado de [DataSklr](#)

Figura 6.2: Tomado de [Towards to Data Science](#)

6.4. Laboratorio

6.4.1. Cross-Validation

6.4.1.1. 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:
```

188 CAPÍTULO 6. MÉTODOS DE SELECCIÓN DE VARIABLES Y REGULARIZACIÓN

```

##      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 <0.0000000000000002 ***
## horsepower -0.157845   0.006446 -24.49 <0.0000000000000002 ***
## ---
## 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)
cv.err <- cv.glm(Auto, glm.fit)
cv.err$delta

```

```

## [1] 24.23151 24.23114

```

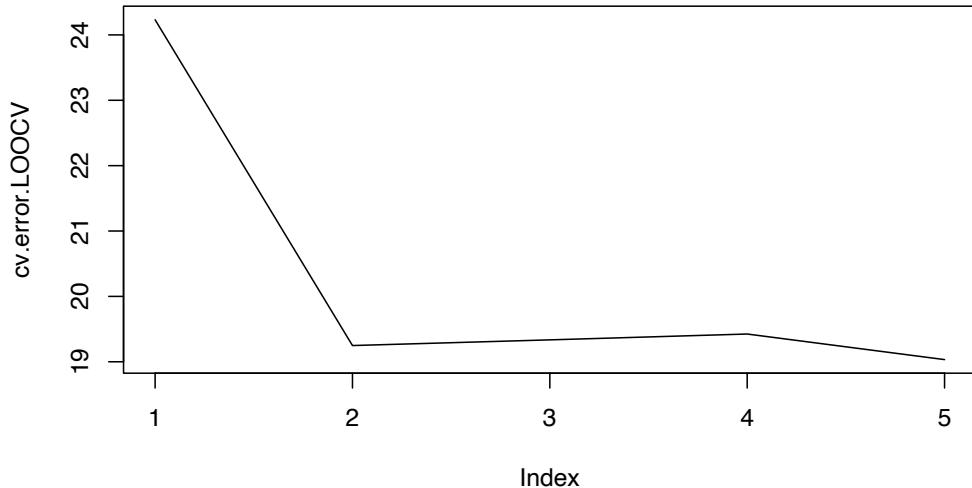
En particular se puede usar un `for` para aplicar este mismo procedimiento a múltiples modelos.

```

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")

```



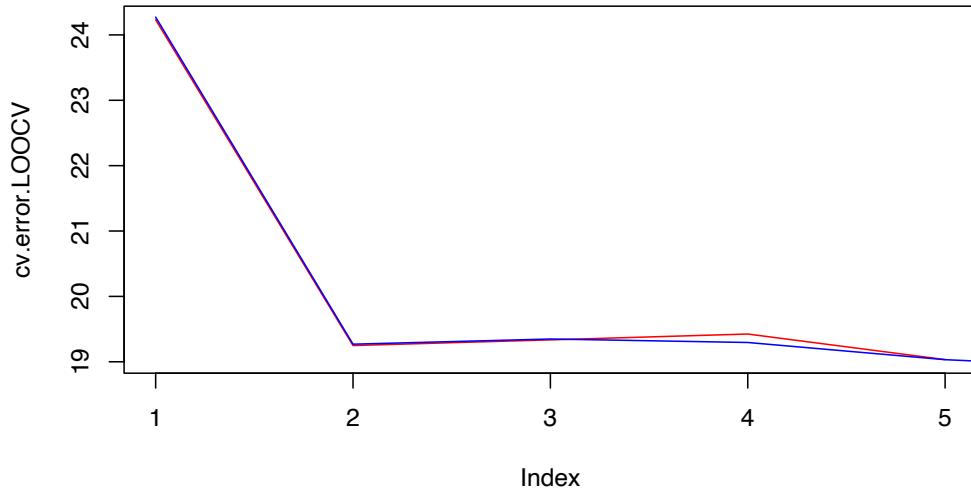
6.4.1.2. K-Fold Cross Validation

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 19.14666
## [9] 18.87013 20.95520

plot(cv.error.LOOCV, type = "l", col = "red")
lines(cv.error.10, type = "l", col = "blue")
```



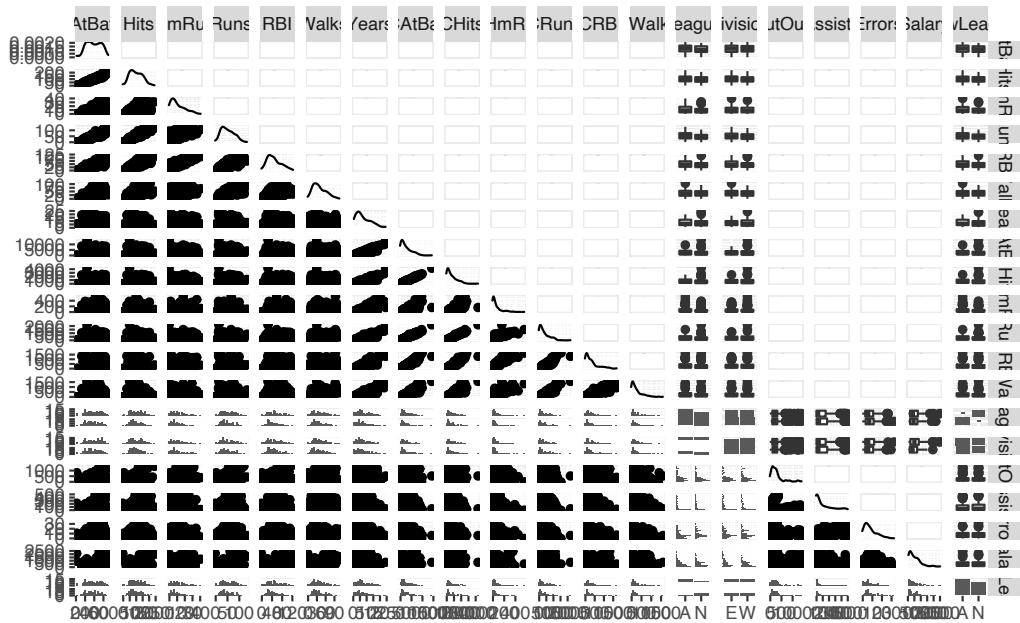
6.4.2. Selección de variables

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.).

6.4.2.1. 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
```

192CAPÍTULO 6. MÉTODOS DE SELECCIÓN DE VARIABLES Y REGULARIZACIÓN

```

##   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
##   Mean   : 260.24
##   3rd Qu.: 339.25
##   Max.    :1566.00
##          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()

```

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, 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 CRBI
## 1   ( 1 )   " "   " "   " "   " "   " "   " "   " "   " "   " "   " "   " "   " * "
## 2   ( 1 )   " "   "*"   " "   " "   " "   " "   " "   " "   " "   " "   " "   " * "
## 3   ( 1 )   " "   "*"   " "   " "   " "   " "   " "   " "   " "   " "   " "   " * "
## 4   ( 1 )   " "   "*"   " "   " "   " "   " "   " "   " "   " "   " "   " "   " * "
## 5   ( 1 )   "*"   "*"   " "   " "   " "   " "   " "   " "   " "   " "   " "   " * "
## 6   ( 1 )   "*"   "*"   " "   " "   " "   " "   "*"   " "   " "   " "   " "   " * "

```

194 CAPÍTULO 6. MÉTODOS DE SELECCIÓN DE VARIABLES Y REGULARIZACIÓN

```

## 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 ...

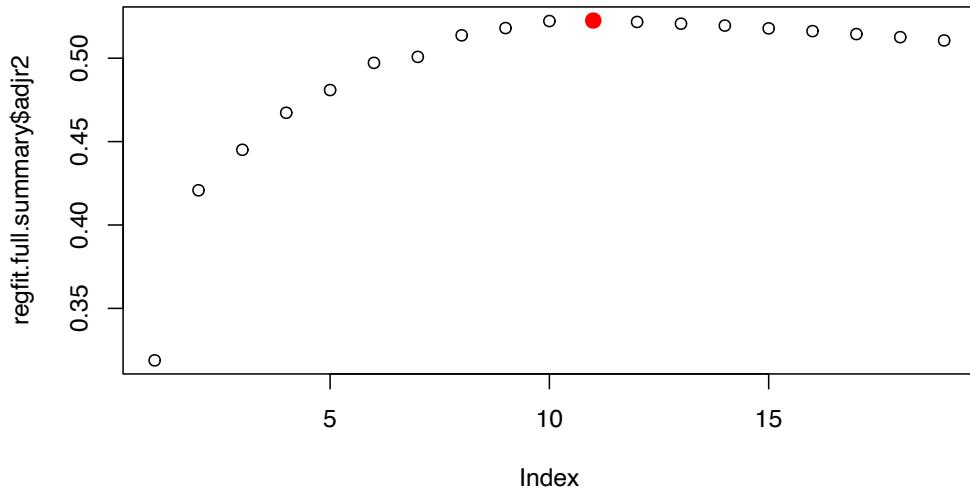
```

```
## ..- 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] 263 110082648 160538 18519052 5581 ...
## ... $ 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] 0.00000000811 0.00001714412 0.0000007908 0.00000708294
## ... $ 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
```

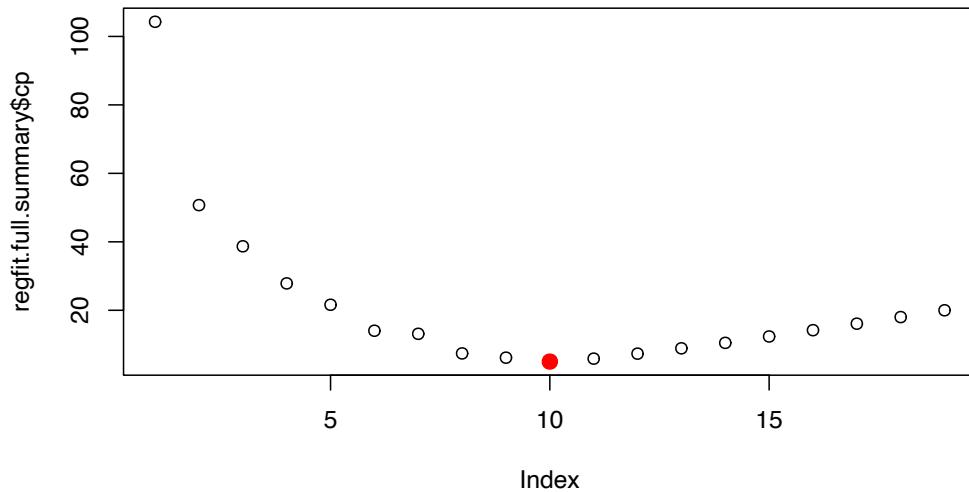
196 CAPÍTULO 6. MÉTODOS DE SELECCIÓN DE VARIABLES Y REGULARIZACIÓN

```
##   ..$ intercept: logi TRUE
##   ..$ lindep    : logi [1:20] 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$adjr2)
plot(regfit.full.summary$adjr2)
points(idx, regfit.full.summary$adjr2[idx], col = "red",
       cex = 2, pch = 20)
```

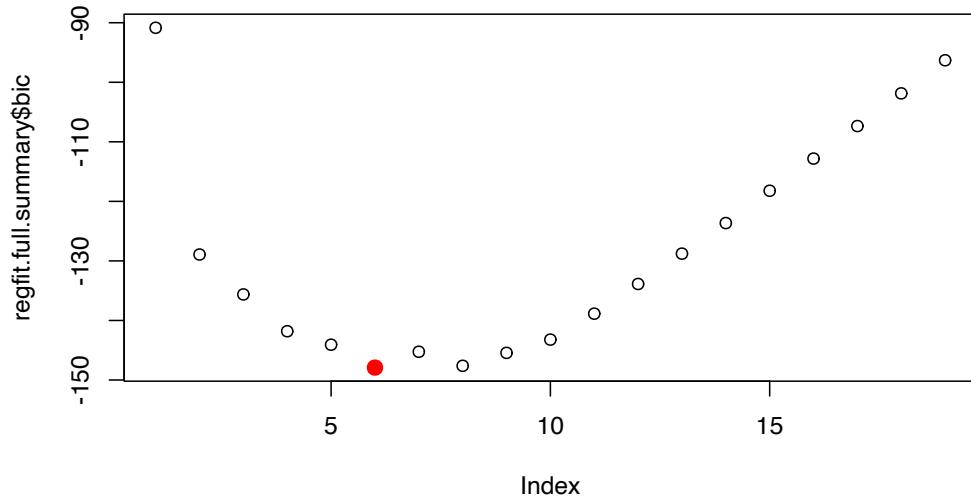


```
idx <- which.min(regfit.full.summary$cp)
plot(regfit.full.summary$cp)
points(idx, regfit.full.summary$cp[idx], col = "red",
       cex = 2, pch = 20)
```

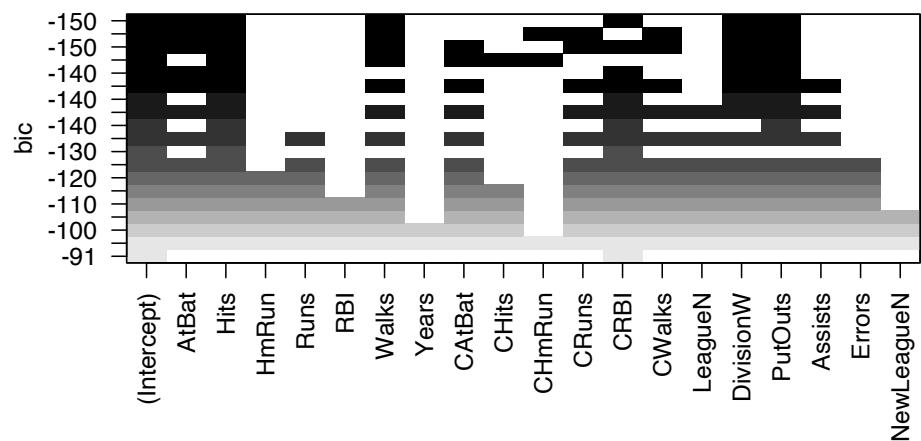


```
idx <- which.min(regfit.full.summary$bic)
plot(regfit.full.summary$bic)
points(idx, regfit.full.summary$bic[idx], col = "red",
       cex = 2, pch = 20)
```

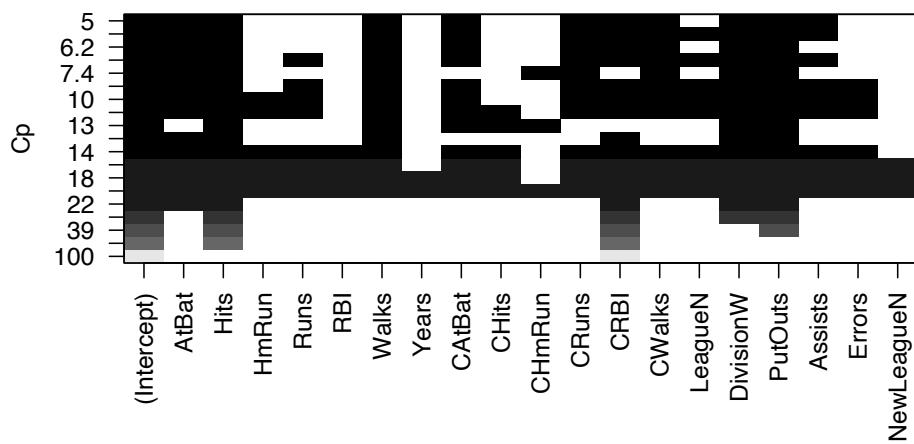
198 CAPÍTULO 6. MÉTODOS DE SELECCIÓN DE VARIABLES Y REGULARIZACIÓN



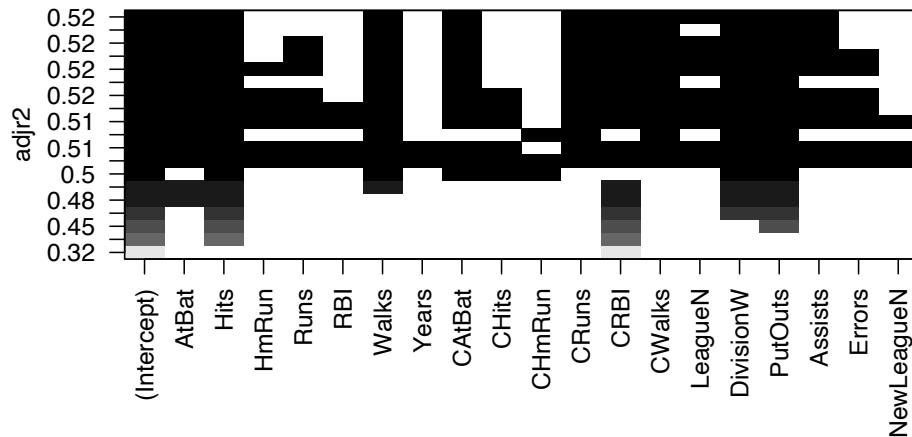
```
plot(regfit.full, scale = "bic")
```



```
plot(regfit.full, scale = "Cp")
```



```
plot(regfit.full, scale = "adjr2")
```



```
coef(regfit.full, 10)
```

## (Intercept)	AtBat	Hits	Walks	CAtBat	CRuns	CWalks	LeagueN	DivisionW	PutOuts	Assists	Errors	NewLeagueN	CRu
## 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									

6.4.2.2. Regresión forward y backward

La función `regsubsets` tiene un parámetro `method`. Usen los valores `forward` y `backward` y comparén 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 = "
## 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 )   "*"  "*"   "*"  "*"   " "   "*"  " "   "*"  " "   "*"  " "   " "   "*"

```

202CAPÍTULO 6. MÉTODOS DE SELECCIÓN DE VARIABLES Y REGULARIZACIÓN

```

## 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 = "
## 19 Variables  (and intercept)
##                 Forced in Forced out
## AtBat           FALSE      FALSE
## Hits            FALSE      FALSE
## HmRun           FALSE      FALSE
## Runs            FALSE      FALSE
## RBI             FALSE      FALSE

```

```

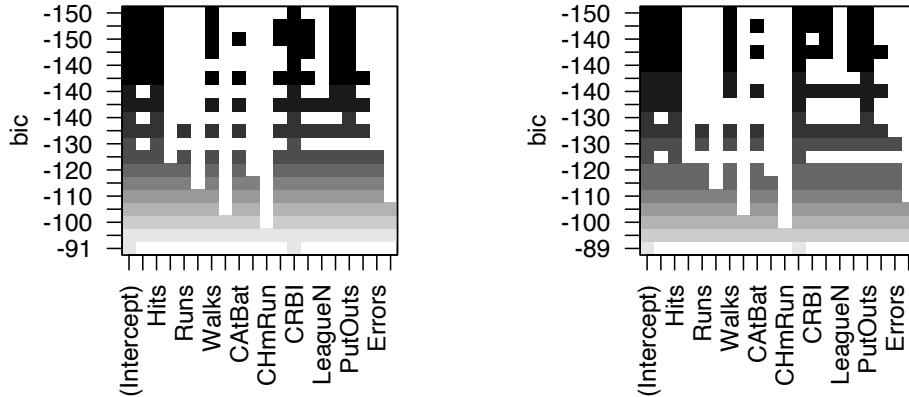
## Walks      FALSE   FALSE
## Years      FALSE   FALSE
## CAtBat    FALSE   FALSE
## CHits      FALSE   FALSE
## CHmRun    FALSE   FALSE
## CRuns      FALSE   FALSE
## CRBI       FALSE   FALSE
## CWalks     FALSE   FALSE
## LeagueN    FALSE   FALSE
## DivisionW  FALSE   FALSE
## PutOuts    FALSE   FALSE
## 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 )   " "   " "   " "   " "   " "   " "   " "   " "

```

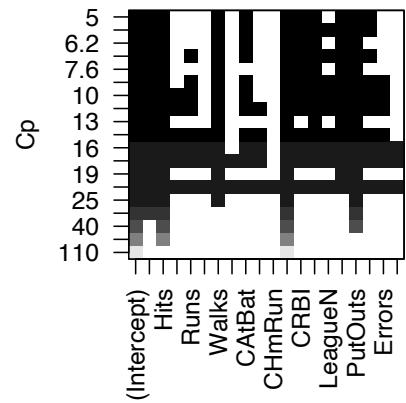
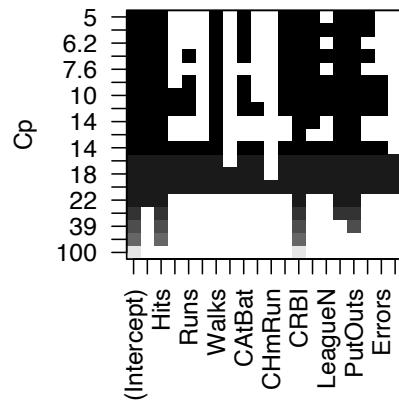
204 CAPÍTULO 6. MÉTODOS DE SELECCIÓN DE VARIABLES Y REGULARIZACIÓN

```
## 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 )  "*"   " "   " "   "*"   " "   "*"   " "   "*"

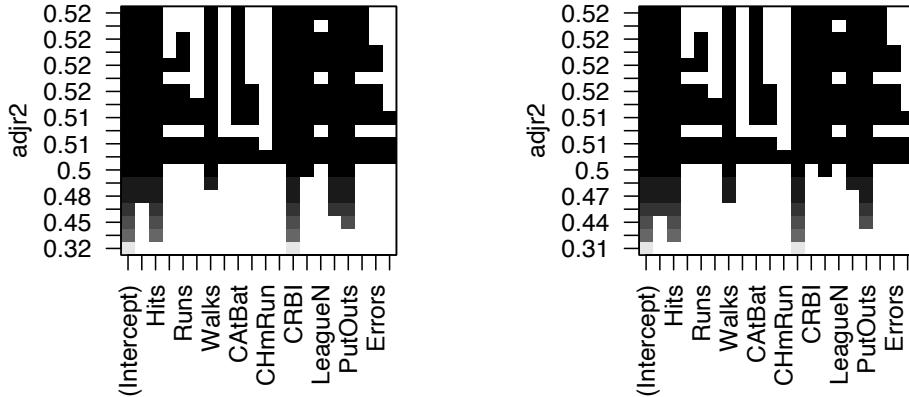
par(mfrow = c(1, 2))
plot(regfit.fwd, scale = "bic")
plot(regfit.bwd, scale = "bic")
```



```
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")
```



6.4.2.3. Regresión Ridge

```
mm <- model.matrix(Salary ~ ., Hitters)[, -1]
x <- mm[, -1]
y <- mm[, 1]
```

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

$$\hat{\beta} = \underset{\beta}{\operatorname{argmin}} \left\{ RSS + \lambda \|\beta\|_2^2 \right\}.$$

Si no se incluye el paramétrico `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

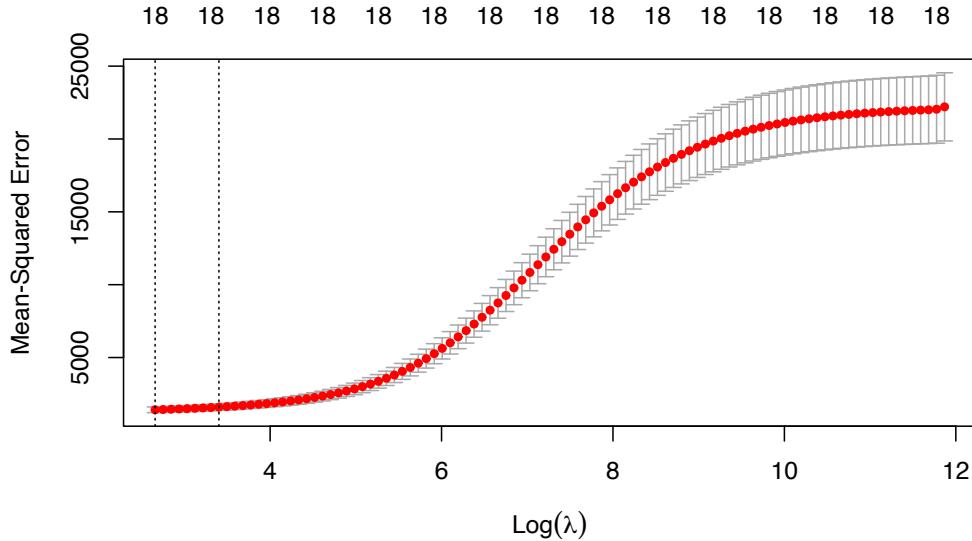
```
set.seed(1)
train <- sample(1:nrow(x), nrow(x)/2)
test <- -train
y.test <- y[test]
ridge.mod <- glmnet(x[train, ], y[train], alpha = 0,
                      lambda = grid)
ridge.pred <- predict(ridge.mod, s = 4, newx = x[test,
])
# MSE
mean((ridge.pred - y.test)^2)

## [1] 1518.103
```

¿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)	Hits	HmRun	Runs	RBI
##	386.3561073385	0.0383991148	0.1116115168	0.0630835744	0.0546958579
##	Walks	Years	CAtBat	CHits	CHmRun
##	0.0509612133	-0.0005083625	0.0001512925	0.0005845613	0.0042705739
##	CRuns	CRBI	CWalks	LeagueN	DivisionW
##	0.0012039190	0.0011381860	0.0007896665	-0.5373933469	-0.1929369010
##	PutOuts	Assists	Errors	NewLeagueN	
##	0.0019622481	0.0042908304	0.0891376637	-0.3078975567	

210 CAPÍTULO 6. MÉTODOS DE SELECCIÓN DE VARIABLES Y REGULARIZACIÓN

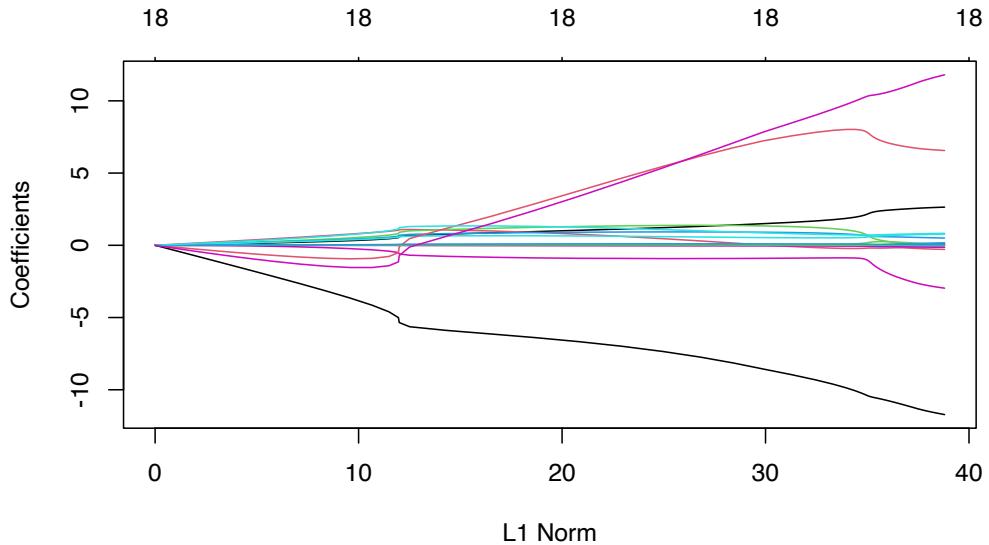
```
sqrt(sum(coef(ridge.mod)[-1, 50]^2))
```

```
## [1] 0.6725377  
ridge.mod$lambda[60]
```

```
## [1] 705.4802  
coef(ridge.mod)[, 60]
```

	(Intercept)	Hits	HmRun	Runs	RBI
##	253.3802418595	0.3853491695	0.8799062459	0.6027727547	0.4892399967
##	Walks	Years	CAtBat	CHits	CHmRun
##	0.4379321710	-0.3131147009	0.0008266820	0.0034422773	0.0159381614
##	CRuns	CRBI	CWalks	LeagueN	DivisionW
##	0.0067934652	0.0052739646	-0.0001061786	-4.1471399456	-0.9052165274
##	PutOuts	Assists	Errors	NewLeagueN	
##	0.0165938946	0.0457914679	0.8721362717	-1.5376591907	
sqrt(sum(coef(ridge.mod)[-1, 60]^2))					

```
## [1] 4.791781  
plot(ridge.mod)
```



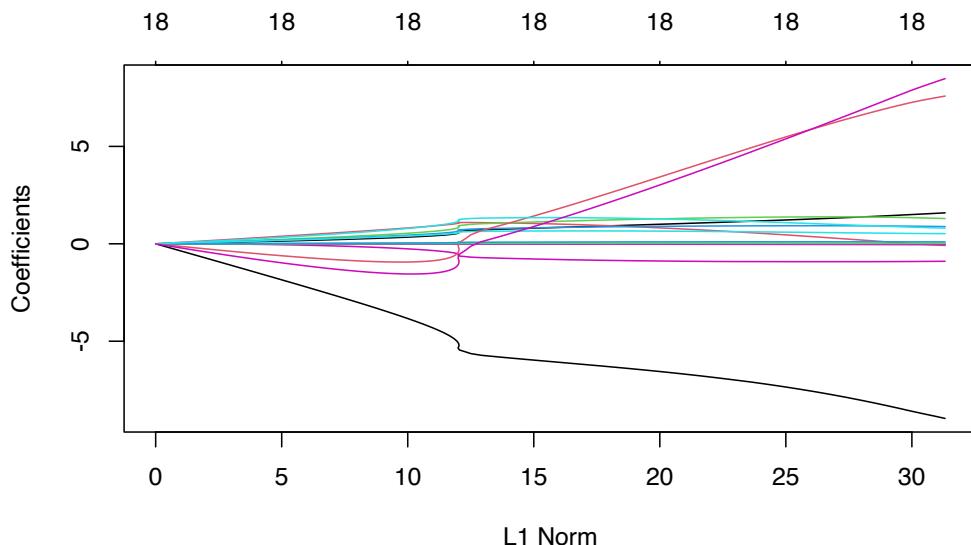
```
ridge.mod <- glmnet(x, y, alpha = 0)
```

```
ridge.mod$lambda
```

## [1]	141729.38335	129138.53556	117666.22398	107213.08094	97688.56632
## [6]	89010.18333	81102.76397	73897.81795	67332.93947	61351.26671
## [11]	55900.98926	50934.89945	46409.98336	42287.04835	38530.38352
## [16]	35107.45045	31988.60131	29146.82213	26557.49879	24198.20379
## [21]	22048.50206	20089.77390	18305.05376	16678.88323	15197.17722
## [26]	13847.10188	12616.96351	11496.10725	10474.82476	9544.27020
## [31]	8696.38354	7923.82080	7219.89040	6578.49523	5994.07985
## [36]	5461.58231	4976.39039	4534.30159	4131.48673	3764.45684
## [41]	3430.03287	3125.31820	2847.67354	2594.69408	2364.18861
## [46]	2154.16061	1962.79091	1788.42198	1629.54350	1484.77935
## [51]	1352.87564	1232.68990	1123.18113	1023.40081	932.48470
## [56]	849.64533	774.16518	705.39048	642.72553	585.62757
## [61]	533.60203	486.19830	443.00578	403.65037	367.79118
## [66]	335.11763	305.34670	278.22054	253.50419	230.98358
## [71]	210.46364	191.76663	174.73061	159.20802	145.06442
## [76]	132.17730	120.43503	109.73591	99.98728	91.10468
## [81]	83.01119	75.63671	68.91735	62.79492	57.21640

212 CAPÍTULO 6. MÉTODOS DE SELECCIÓN DE VARIABLES Y REGULARIZACIÓN

```
## [86] 52.13345 47.50206 43.28211 39.43704 35.93356
## [91] 32.74133 29.83268 27.18242 24.76761 22.56733
## [96] 20.56251 18.73579 17.07135 15.55478 14.17294
plot(ridge.mod)
```



```
predict(ridge.mod, s = 50, type = "coefficients")
```

```
## 19 x 1 sparse Matrix of class "dgCMatrix"
##                               s1
## (Intercept) 94.5913466484
## Hits         1.1742395636
## HmRun        0.5429246143
## Runs         1.3628990314
## RBI          0.9229879932
## Walks        0.6114551657
## Years        -0.9130962202
## CAtBat       0.0022266078
## CHits        0.0071572845
## CHmRun       -0.0190970399
```

```
## CRuns      0.0101467149
## CRBI       0.0006507061
## CWalks     -0.0293738133
## LeagueN    -7.1549920715
## DivisionW   5.0662991581
## PutOuts     0.0207351205
## Assists     0.1035681348
## Errors      1.1324147256
## NewLeagueN  4.8784587235

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] 1518.103

ridge.pred <- predict(ridge.mod, s = 10000000000, newx = x[test,
], exact = TRUE)
mean((ridge.pred - y.test)^2)

## [1] 21582.12

ridge.pred <- predict(ridge.mod, s = 0, newx = x[test,
], exact = FALSE)
mean((ridge.pred - y.test)^2)

## [1] 1806.907

lm(y ~ x, subset = train)

##
## Call:
## lm(formula = y ~ x, subset = train)
```

```

##  

## Coefficients:  

## (Intercept)      xHits      xHmRun      xRuns      xRBI      xWalk  

## 55.76031        2.98008     -0.30105    -0.44890    0.29740   0.7832  

## xYears          xCAtBat     xCHits      xCHmRun    xCRuns      xCRB  

## -2.98949        0.11920     -0.46305    -0.04969    0.07318   0.1305  

## xCWalks         xLeagueN    xDivisionW  xPutOuts    xAssists   xError  

## -0.13227       -9.74183     1.03070     0.00804    0.04660   0.6501  

## xNewLeagueN  

##      5.53145  

predict(ridge.mod, s = 0, type = "coefficients", exact = FALSE)  

## 19 x 1 sparse Matrix of class "dgCMatrix"  

##                               s1  

## (Intercept) 56.400164521  

## Hits         2.935963537  

## HmRun        -0.411469189  

## Runs         -0.412389832  

## RBI          0.355794523  

## Walks        0.770020095  

## Years        -2.741424760  

## CAtBat       0.109568413  

## CHits        -0.417184445  

## CHmRun       0.025750294  

## CRuns        0.059026090  

## CRBI         0.100039043  

## CWalks       -0.128465225  

## LeagueN      -8.986012754  

## DivisionW    1.414164932  

## PutOuts      0.008851545  

## Assists      0.050791034  

## Errors       0.647146833  

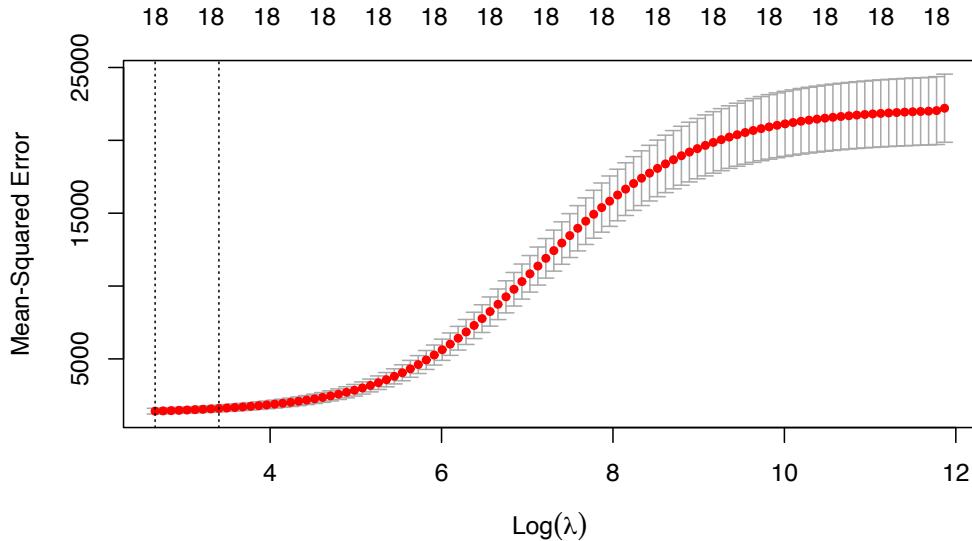
## NewLeagueN   4.525279141  

set.seed(1)  

cv.out <- cv.glmnet(x[train], y[train], alpha = 0)  

plot(cv.out)

```



```

bestlam <- cv.out$lambda.min
bestlam

## [1] 14.27118
log(bestlam)

## [1] 2.658242
ridge.pred <- predict(ridge.mod, s = bestlam, newx = x[test,
  ])
mean((ridge.pred - y.test)^2)

## [1] 1669.854
out <- glmnet(x, y, alpha = 0)
predict(out, type = "coefficients", s = bestlam, exact = FALSE)

## 19 x 1 sparse Matrix of class "dgCMatrix"
##           s1
## (Intercept) 69.824252126

```

```

## Hits          1.586649928
## HmRun        -0.083817620
## Runs         1.298201495
## RBI          0.885217905
## Walks        0.521415084
## Years        -0.897115251
## CAtBat       0.004160358
## CHits        0.005192629
## CHmRun       0.022521114
## CRuns        0.009343228
## CRBI         -0.008077328
## CWalks       -0.045652211
## LeagueN      -8.954990325
## DivisionW    7.576655558
## PutOuts       0.016433884
## Assists       0.098673953
## Errors        0.806331327
## NewLeagueN   8.465527964

```

6.4.3. Regresión Lasso

Ejecute los procedimientos anteriores con `glmnet` pero modifique el parámetro `alpha = 0`. Compare los resultados.

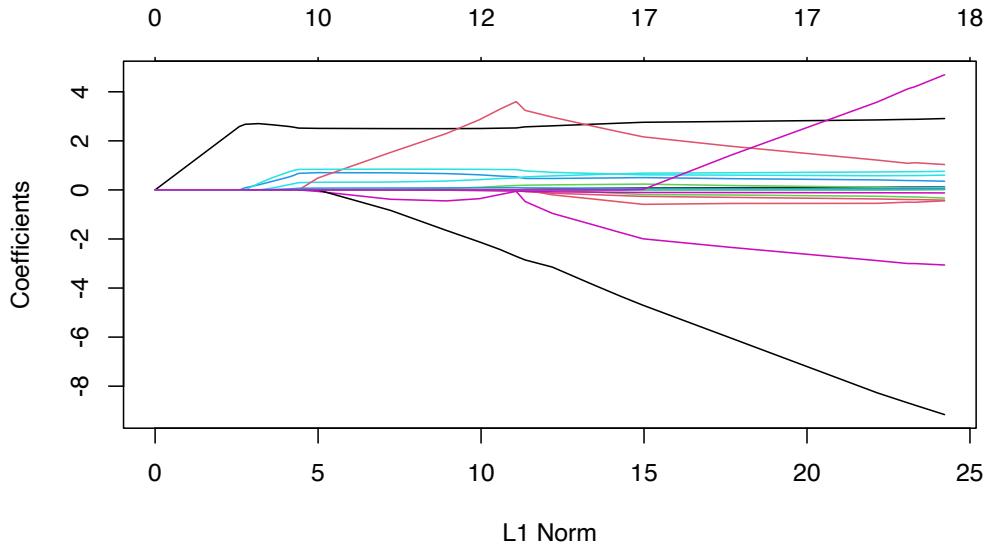
En este caso, se están encontrando los valores de β tal que

$$\hat{\beta} = \underset{\beta}{\operatorname{argmin}} \left\{ RSS + \lambda \|\beta\|_1^2 \right\}.$$

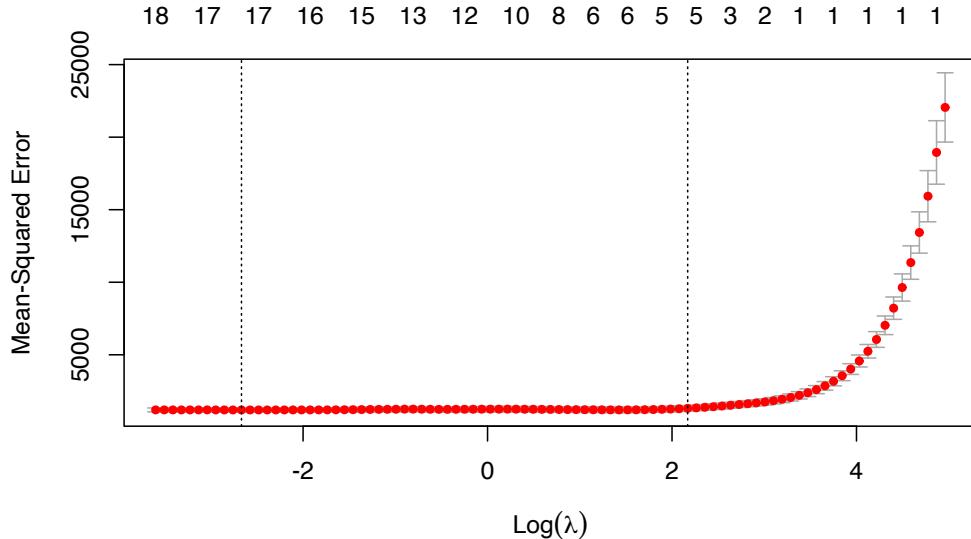
```

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] 0.06939507
log(bestlam)

## [1] -2.667939
lasso.pred <- predict(lasso.mod, s = bestlam, newx = x[test,
  ])
mean((lasso.pred - y.test)^2)

## [1] 1698.516
out <- glmnet(x, y, alpha = 1, lambda = grid)
lasso.coef <- predict(out, type = "coefficients", s = bestlam)
lasso.coef

## 19 x 1 sparse Matrix of class "dgCMatrix"
##           s1
## (Intercept) 49.43034834
## Hits         2.58756283

```

```
## HmRun          .
## Runs           0.01352442
## RBI            0.50950339
## Walks          0.73008990
## Years          -2.72097480
## CAtBat         0.06956598
## CHits          -0.22930533
## CHmRun         0.10795338
## CRuns          0.11833697
## CRBI           -0.05929447
## CWalks         -0.11751099
## LeagueN        -10.52682373
## DivisionW      6.41207311
## PutOuts         0.01253616
## Assists         0.04384029
## Errors          0.79511992
## NewLeagueN     10.34406169

lasso.coef[lasso.coef != 0]

## [1] 49.43034834  2.58756283  0.01352442  0.50950339  0.73008990
## [6] -2.72097480  0.06956598 -0.22930533  0.10795338  0.11833697
## [11] -0.05929447 -0.11751099 -10.52682373  6.41207311  0.01253616
## [16]  0.04384029  0.79511992  10.34406169
```

6.5. Ejercicios

- Del libro (James y col. 2013)
 - Capítulo 5: 2, 5, 8.
 - Capítulo 6: 5, 6, 7, 8, 10.

Capítulo 7

Otros Clasificadores

7.1. Clasificador Bayesiano

Bajo el modelo de aprendizaje estadístico, suponga que se quiere estimar f usando el conjunto de entrenamiento $\{(x_1, y_1), \dots, (x_n, y_n)\}$ donde y_1, \dots, y_n es categórica. Para evaluar la precisión del clasificador \hat{f} podemos usar la tasa de error:

$$\frac{1}{n} \sum_{i=1}^n I(y_i \neq \hat{y}_i)$$

donde \hat{y}_i es el nivel predecido de la variable categórica para el individuo i -ésimo. La tasa de error mide la proporción de observaciones mal clasificadas por \hat{f} dentro del conjunto de entrenamiento. El mismo concepto se puede aplicar en el conjunto de prueba, es decir si Z_0 es el conjunto de índices de datos de prueba con tamaño m :

$$\frac{1}{m} \sum_{i \in Z_0} I(y_i \neq \hat{y}_i) \tag{7.1}$$

Decimos que un clasificador es bueno cuando el error de prueba en (7.1) es el más pequeño.

Es posible demostrar que el error de prueba se minimiza cuando \hat{f} asigna a cada observación el nivel con la probabilidad más alta dados los predictores, es decir se asigna la clase j a la observación x_0 en donde

$$P(Y = j|X = x_0)$$

es máximo. A este clasificador se le llama *clasificador bayesiano*. En el caso en que el número de niveles o categorías de la variable dependiente es 2 ($j = 1, 2$), entonces se selecciona el nivel j -ésimo si:

$$P(Y = j|X = x_0) > 0,5$$

Al conjunto $\{x_0 : P(Y = j|X = x_0) = 0,5\}$ se le llama frontera de decisión de Bayes.

La *tasa bayesiana de error* del clasificador para un conjunto de datos fijos es: $1 - \max_j P(Y = j|X = x_0)$. En general la tasa de error bayesiana sería:

$$1 - E \left[\max_j P(Y = j|X) \right]$$

Para el caso de clasificación la tasa de error bayesiana es equivalente al error irreducible.

Inconveniente: en datos reales no conocemos $P(Y = j|X = x_0)$. Se puede aproximar estas probabilidades usando lo que se conoce como *Bayes ingenuo* (*Naïve Bayes*).

- Paso 1: Calcule la probabilidad previa para las etiquetas de clase dadas
- Paso 2: Encuentre la probabilidad de verosimilitud con cada atributo para cada clase
- Paso 3: Ponga estos valores en la fórmula de Bayes y calcule la probabilidad posterior.
- Paso 4: Revise qué clase tiene una probabilidad más alta, dado que la entrada pertenece a la clase de probabilidad más alta.

7.2. Método de k vecinos más cercanos (KNN)

Este método aproxima la probabilidad condicional del clasificador bayesiano. Dado un entero K y una observación de prueba x_0 , el clasificador hace lo

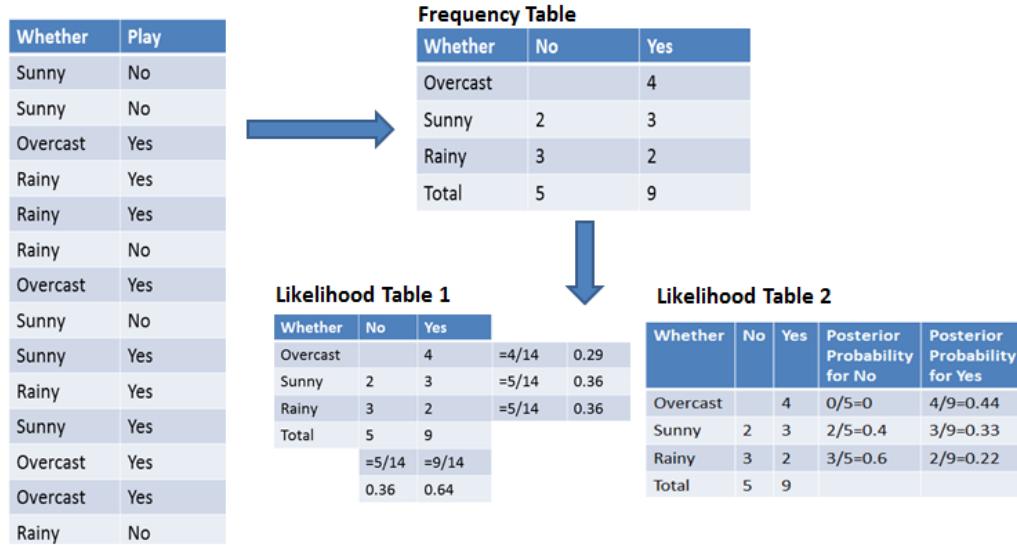


Figura 7.1: Bayes ingenuo (tomado de <https://www.datacamp.com/tutorial/naive-bayes-scikit-learn>)

siguiente 1. Encuentra los primeros K vecinos más cercanos (usando alguna distancia) de observaciones que son más cercanas a x_0 : \mathcal{N}_0 2. Luego calcula

$$P(Y = j|X = x_0) = \frac{1}{K} \sum_{i \in \mathcal{N}_0} I(y_i = j)$$

- Finalmente, siguiendo la regla de bayes se selecciona la categoría con probabilidad condicional máxima.

7.3. Análisis Discriminante

Recuerden que en el caso del modelo logístico, se tiene que:

$$P(Y = 1|X = x) = \frac{e^{\beta_0 + \beta_1 X_1 + \dots + \beta_p X_p}}{1 + e^{\beta_0 + \beta_1 X_1 + \dots + \beta_p X_p}}$$

donde X_1, \dots, X_p son los predictores. Para el modelo logístico tenemos los siguientes inconvenientes:

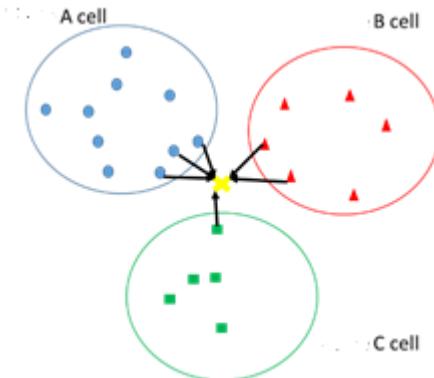


Figura 7.2: Clasificación según K vecinos más cercanos (Wikimedia Commons)

- Cuando las clases están muy separadas, los parámetros del modelo logístico tienden a ser muy inestables.
- Cuando la distribución de los predictores es aproximadamente normal en cada una de las clases, entonces el modelo discriminante lineal es más estable que el logístico.
- El modelo logístico aplica solamente en el caso de 2 clases.

Suponga que se quiere clasificar una observación en $K \geq 2$ clases. Sea π_k la probabilidad previa de que la observación provenga de la clase k -ésima. Sea

$$f_k(x) = P(X = x|Y = k)$$

por el teorema de Bayes:

$$P_k(x) = P(Y = k|X = x) = \frac{\pi_k f_k(x)}{\sum_{l=1}^K \pi_l f_l(x)}$$

Estimación de los componentes:

- π_k : proporción de observaciones en el conjunto de entrenamiento que pertenecen a la clase k -ésima.
- $f_k(x)$: supuesto paramétrico que define el tipo de análisis discriminante.

7.3.1. Análisis discriminante lineal

7.3.1.1. Caso p=1

Asuma que

$$f_k(x) = \frac{1}{\sqrt{2\pi}\sigma_k} \exp\left(-\frac{1}{2\sigma_k^2}(x - \mu_k)^2\right)$$

donde μ_k y σ_k son la media y desviación estándar para cada clase en la variable dependiente. Asumiendo que $\sigma^2 = \sigma_1^2 = \dots = \sigma_K^2$ se puede comprobar que el clasificador bayesiano asigna la clase k si

$$\delta_k(x) = x \frac{\mu_k}{\sigma^2} - \frac{\mu_k^2}{2\sigma^2} + \log(\pi_k) \quad (7.2)$$

es el máximo entre los valores correspondientes a cada clase. A esta función se le llama *función discriminante*.

El método de análisis discriminante lineal (LDA) asume que:

$$\begin{aligned} \hat{\mu}_k &= \frac{1}{n_k} \sum_{i:y_i=k} x_i \\ \hat{\sigma}^2 &= \frac{1}{n-K} \sum_{k=1}^K \sum_{i:y_i=k} (x_i - \hat{\mu}_k)^2 \\ \hat{\pi}_k &= \frac{n_k}{n} \end{aligned}$$

como estimadores plug-in en (7.2).

7.3.1.2. Caso p>1

Generalizando la sección anterior, podemos asumir que $X = (X_1, \dots, X_p)$ proviene de una distribución Gaussiana multivariada. Es decir, asuma que las observaciones en la clase k tienen distribución $N(\mu_k, \Sigma)$ donde μ_k : vector de medias para la clase k y Σ es la matriz de varianza-covarianza para todas las K clases.

La función discriminante en este caso sería:

$$\delta_k(x) = x^T \Sigma^{-1} \mu_k - \frac{1}{2} \mu_k^T \Sigma^{-1} \mu_k + \log \pi_k$$

El método LDA sustituye los parámetros en la fórmula anterior con estimadores empíricos, tal y como se hizo para $p = 1$. La escogencia de la clase estimada sigue el mismo criterio.

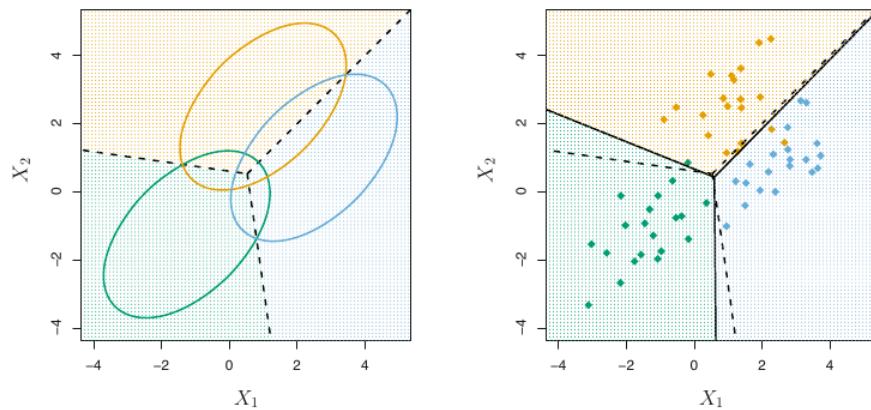


FIGURE 4.6. An example with three classes. The observations from each class are drawn from a multivariate Gaussian distribution with $p = 2$, with a class-specific mean vector and a common covariance matrix. Left: Ellipses that contain 95 % of the probability for each of the three classes are shown. The dashed lines are the Bayes decision boundaries. Right: 20 observations were generated from each class, and the corresponding LDA decision boundaries are indicated using solid black lines. The Bayes decision boundaries are once again shown as dashed lines.

Figura 7.3: Simulación de Análisis Discriminante Lineal (James y col. 2013)

7.3.2. Análisis discriminante cuadrático

Bajo los supuestos del LDA, asuma que Σ_k es la matriz de covarianza para la clase k . En este caso las funciones discriminantes tendrían la forma:

$$\delta_k(x) = -\frac{1}{2}(x - \mu_k)^T \Sigma_k^{-1} (x - \mu_k) - \frac{1}{2} \log |\Sigma_k| + \log \pi_k$$

Al uso de las funciones anteriores como herramientas de clasificación se le llama Análisis Discriminante Cuadrático (QDA).

Relación LDA vs QDA:

- LDA es menos flexible que QDA, por la diferencia en el número de parámetros. Por lo tanto LDA tiene menos varianza que QDA.

- Si el supuesto de varianzas constantes en LDA no es adecuado, entonces el sesgo es alto.
- QDA es más adecuado que LDA cuando el número de observaciones es relativamente alto, debido a que el supuesto de varianzas constantes es más difícil de alcanzar.

Comparación de métodos:

- LDA y regresión logística producen fronteras de decisión lineales.
- LDA asume más sobre el comportamiento de los datos, con respecto a la regresión logística.
- KNN es no paramétrico, por lo tanto produce fronteras de decisión más flexibles que LDA o QDA. El grado de suavidad del clasificador (en términos de sus fronteras) depende del parámetro K .
- QDA ofrece fronteras de decisión más flexibles que LDA o logística.
- KNN no tiene la misma capacidad de interpretabilidad que la regresión logística.
- Como KNN depende de la distancia entre observaciones, entonces la escala de las covariables importa.

7.4. Laboratorio

Datos sociodemográficos y de productos de aseguramiento de 5822 clientes. La variable dependiente es si cada cliente adquirió un seguro de remolques (<https://liacs.leidenuniv.nl/~puttenpwhvander/library/cc2000/data.html>).

```
library(ISLR)
data(Caravan)
dim(Caravan)

## [1] 5822    86
head(Caravan)

##   MOSTYPE MAANTHUI  MGEMOMV MGEMLEEF MOSHOOFD MGODRK MGODPR MGODOV MGODGE MRELGE
## 1      33        1       3       2       8       0       5       1       3       7
## 2      37        1       2       2       8       1       4       1       4       6
## 3      37        1       2       2       8       0       4       2       4       3
## 4       9        1       3       3       3       2       3       2       4       5
## 5     40        1       4       2      10       1       4       1       4       7
```

## 6	23	1	2	1	5	0	5	0	5
## MRELSA	MRELOV	MFALLEEN	MFGEKIND	MFWEKIND	MOPLHOOG	MOPLMIDD	MOPLLAAG	MBERH	
## 1	0	2	1	2	6	1	2	7	
## 2	2	2	0	4	5	0	5	4	
## 3	2	4	4	4	2	0	5	4	
## 4	2	2	2	3	4	3	4	2	
## 5	1	2	2	4	4	5	4	0	
## 6	6	3	3	5	2	0	5	4	
## MBERZELF	MBERBOER	MBERMIDD	MBERARBG	MBERARBO	MSKA	MSKB1	MSKB2	MSKC	MSKD
## 1	0	1	2	5	2	1	1	2	1
## 2	0	0	5	0	4	0	2	3	5
## 3	0	0	7	0	2	0	5	0	4
## 4	0	0	3	1	2	3	2	1	4
## 5	5	4	0	0	0	9	0	0	0
## 6	0	0	4	2	2	2	2	4	2
## MHHUUR	MHKOOP	MAUT1	MAUT2	MAUTO	MZFONDS	MZPART	MINKM30	MINK3045	MINK4575
## 1	1	8	8	0	1	8	1	0	4
## 2	2	7	7	1	2	6	3	2	0
## 3	7	2	7	0	2	9	0	4	5
## 4	5	4	9	0	0	7	2	1	5
## 5	4	5	6	2	1	5	4	0	9
## 6	9	0	5	3	3	9	0	5	2
## MINK7512	MINK123M	MINKGEM	MKOOPKLA	PWAPART	PWABEDR	PWALAND	PPERSAUT	PBESA	
## 1	0	0	4	3	0	0	0	6	
## 2	2	0	5	4	2	0	0	0	
## 3	0	0	3	4	2	0	0	6	
## 4	0	0	4	4	0	0	0	6	
## 5	0	0	6	3	0	0	0	0	
## 6	0	0	3	3	0	0	0	6	
## PMOTSCO	PVRAAUT	PAANHANG	PTRACTOR	PWERKT	PBROM	PLEVEN	PPERSONG	PGEZONG	
## 1	0	0	0	0	0	0	0	0	
## 2	0	0	0	0	0	0	0	0	
## 3	0	0	0	0	0	0	0	0	
## 4	0	0	0	0	0	0	0	0	
## 5	0	0	0	0	0	0	0	0	
## 6	0	0	0	0	0	0	0	0	
## PWAOREG	PBRAND	PZEILPL	PPLEZIER	PFIETS	PINBOED	PBYSTAND	AWAPART	AWABEDR	
## 1	0	5	0	0	0	0	0	0	

```

## 2      0      2      0      0      0      0      0      2      0
## 3      0      2      0      0      0      0      0      1      0
## 4      0      2      0      0      0      0      0      0      0
## 5      0      6      0      0      0      0      0      0      0
## 6      0      0      0      0      0      0      0      0      0
##    AWALAND APERSAUT ABESAUT AMOTSCO AVRAAUT AAANHANG ATRACTOR AWERKT ABROM
## 1      0      1      0      0      0      0      0      0      0
## 2      0      0      0      0      0      0      0      0      0
## 3      0      1      0      0      0      0      0      0      0
## 4      0      1      0      0      0      0      0      0      0
## 5      0      0      0      0      0      0      0      0      0
## 6      0      1      0      0      0      0      0      0      0
##    ALEVEN APERSONG AGEZONG AWAOREG ABRAND AZEILPL APLEZIER AFIETS AINBOED
## 1      0      0      0      0      1      0      0      0      0
## 2      0      0      0      0      1      0      0      0      0
## 3      0      0      0      0      1      0      0      0      0
## 4      0      0      0      0      1      0      0      0      0
## 5      0      0      0      0      1      0      0      0      0
## 6      0      0      0      0      0      0      0      0      0
##    ABYSTAND Purchase
## 1      0      No
## 2      0      No
## 3      0      No
## 4      0      No
## 5      0      No
## 6      0      No

```

Vamos a usar las herramientas en el paquete *tidymodels* para efectuar una comparación entre los métodos de clasificación que hemos visto en clase. El objetivo es clasificar a los clientes entre compradores/no compradores del seguro (variable dependiente: Purchase, covariables: el resto)

```

library(tidymodels)
library(tidyverse)

```

El primer paso es construir una separación de conjunto de entrenamiento y de conjunto de prueba:

```
set.seed(1234)
Caravan.split <- initial_split(Caravan, prop = 0.8,
                                strata = Purchase)
Caravan.training <- Caravan.split %>%
  training()
Caravan.testing <- Caravan.split %>%
  testing()
```

Como vamos a usar el método KNN, lo conveniente es estandarizar todas las covariables:

```
Caravan.recipe <- recipe(Purchase ~ ., data = Caravan.training) %>%
  step_normalize(all_predictors(), -all_outcomes())
```

y aplicamos la receta sobre el conjunto de prueba para verificar que la receta funciona bien:

```
Caravan.recipe %>%
  prep() %>%
  bake(new_data = Caravan.testing)
```

```
## # A tibble: 1,165 x 86
##   MOSTYPE MAANTHUI MGEMOMV MGEMLEEF MOSHOOFD MGODRK MGODPR MGODOV MGODGE
##   <dbl>     <dbl>    <dbl>    <dbl>    <dbl>    <dbl>    <dbl>    <dbl>
## 1 -0.101    -0.275   -0.856   -2.45    -0.274   -0.695   0.225  -1.05   1.08
## 2  0.680    -0.275   -0.856    0.0258   0.779   -0.695   1.39   -1.05  -0.792
## 3 -1.19     -0.275    0.401    0.0258  -0.977   0.288  -0.936   0.917  0.457
## 4  0.680     2.27    0.401    0.0258   0.779   -0.695  -0.356   0.917  -0.168
## 5 -0.881     2.27    1.66    0.0258  -0.977   -0.695  -0.356   0.917  0.457
## 6  1.23      -0.275    0.401    0.0258   1.48   -0.695  -0.936  -1.05   1.71
## 7  0.680     -0.275    1.66    0.0258   0.779   0.288  -0.356  -0.0652  1.08
## 8 -1.35      -0.275    0.401   -1.21    -1.33   -0.695   1.39   0.917  -2.04
## 9 -1.66      -0.275    0.401    0.0258  -1.68   1.27    1.39   -1.05  -2.04
## 10 1.07      -0.275    0.401    0.0258   1.13   -0.695   0.225  -0.0652 -0.168
## # ... with 1,155 more rows, and 77 more variables: MRELGE <dbl>, MRELSA <dbl>,
## # MRELOV <dbl>, MFALLEEN <dbl>, MFGEKIND <dbl>, MFWEKIND <dbl>,
## # MOPLHOOG <dbl>, MOPLMIDD <dbl>, MOPLLAAG <dbl>, MBERHOOG <dbl>,
## # MBERZELF <dbl>, MBERBOER <dbl>, MBERMIDD <dbl>, MBERARBG <dbl>,
## # MBERARBO <dbl>, MSKA <dbl>, MSKB1 <dbl>, MSKB2 <dbl>, MSKC <dbl>,
```

```
## #  MSKD <dbl>, MHHUUR <dbl>, MHKOOP <dbl>, MAUT1 <dbl>, MAUT2 <dbl>,
## #  MAUTO <dbl>, MZFONDS <dbl>, MZPART <dbl>, MINKM30 <dbl>, ...
```

7.4.1. Clasificador logístico

Vamos a ajustar un modelo logístico a los datos. Primero especificamos el modelo:

```
modelo_logistico <- logistic_reg() %>%
  set_engine("glm") %>%
  set_mode("classification")
```

y después definimos un objeto tipo *workflow* para unir el tratamiento de datos (recipe) con el modelo:

```
logistico_wf <- workflow() %>%
  add_model(modelo_logistico) %>%
  add_recipe(Caravan.recipe)
```

y ajustamos el modelo:

```
logistico_ajuste <- logistico_wf %>%
  fit(data = Caravan.training)
```

Obtenemos predicciones en el conjunto de prueba:

```
predicciones_probs <- predict(logistico_ajuste, new_data = Caravan.testing,
  type = "prob")
predicciones_categ <- predict(logistico_ajuste, new_data = Caravan.testing)
head(predicciones_probs)
```

```
## # A tibble: 6 x 2
##   .pred_No .pred_Yes
##       <dbl>     <dbl>
## 1     0.981    0.0189
## 2     0.990    0.0100
## 3     0.880    0.120
## 4     0.921    0.0787
## 5     0.967    0.0329
## 6     0.994    0.00628
```

```
head(predicciones_categ)
```

```
## # A tibble: 6 x 1
##   .pred_class
##   <fct>
## 1 No
## 2 No
## 3 No
## 4 No
## 5 No
## 6 No
```

Unimos todos los resultados en un solo arreglo:

```
resultados_logistico <- Caravan.testing %>%
  dplyr::select(Purchase) %>%
  bind_cols(predicciones_categ) %>%
  bind_cols(predicciones_probs)
```

```
head(resultados_logistico)
```

```
## # A tibble: 6 x 4
##   Purchase .pred_class .pred_No .pred_Yes
## 1 No       No        0.9811405 0.01885947
## 2 No       No        0.9899782 0.01002185
## 3 No       No        0.8796827 0.12031730
## 4 No       No        0.9212685 0.07873146
## 5 No       No        0.9670789 0.03292113
## 6 No       No        0.9937207 0.00627930
```

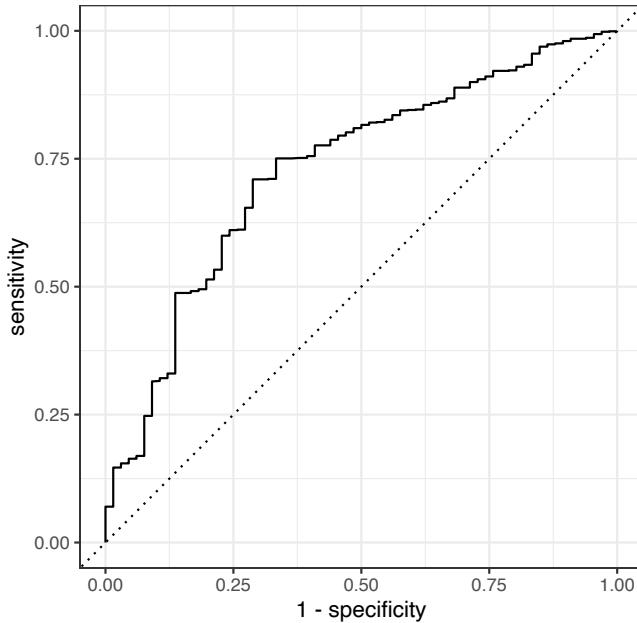
y podemos calcular la matriz de confusión:

```
conf_mat(resultados_logistico, truth = Purchase, estimate = .pred_class)

##          Truth
## Prediction  No  Yes
##           No 1095   64
##           Yes    4    2
```

curva ROC:

```
roc_curve(resultados_logistico, truth = Purchase, estimate = .pred_No) %>%
  autoplot()
```



y finalmente el área bajo la curva ROC:

```
roc_auc(resultados_logistico, truth = Purchase, estimate = .pred_No)
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>        <dbl>
## 1 roc_auc binary     0.728
```

Existe otra alternativa de ajuste con el comando *last_fit* que automatiza el proceso:

```
last_fit_logistica <- logistico_wf %>%
  last_fit(split = Caravan.split)
```

Obtenemos métricas:

```

last_fit_logistica %>%
  collect_metrics()

## # A tibble: 2 x 4
##   .metric  .estimator .estimate .config
##   <chr>    <chr>        <dbl> <chr>
## 1 accuracy binary      0.942 Preprocessor1_Model1
## 2 roc_auc  binary      0.728 Preprocessor1_Model1

```

y predicciones:

```

head(last_fit_logistica %>%
  collect_predictions())

## # A tibble: 6 x 7
##   id          .pred_No .pred_Yes .row .pred_class Purchase .config
##   <chr>       <dbl>     <dbl> <int> <fct>      <fct>   <chr>
## 1 train/test split  0.981    0.0189     6 No        No      Preprocess...
## 2 train/test split  0.990    0.0100     8 No        No      Preprocess...
## 3 train/test split  0.880    0.120      12 No       No      Preprocess...
## 4 train/test split  0.921    0.0787     22 No       No      Preprocess...
## 5 train/test split  0.967    0.0329     25 No       No      Preprocess...
## 6 train/test split  0.994    0.00628    28 No       No      Preprocess...

```

7.4.2. Análisis Discriminante Lineal

Usando el mismo procedimiento de datos anterior, definimos el modelo LDA:

```

library(discrim)
modelo_lda <- discrim_linear() %>%
  set_engine("MASS") %>%
  set_mode("classification")

```

flujo de trabajo:

```

lda_wf <- workflow() %>%
  add_model(modelo_lda) %>%
  add_recipe(Caravan.recipe)

```

ajuste del modelo:

```
last_fit_lda <- lda_wf %>%
  last_fit(split = Caravan.split)
```

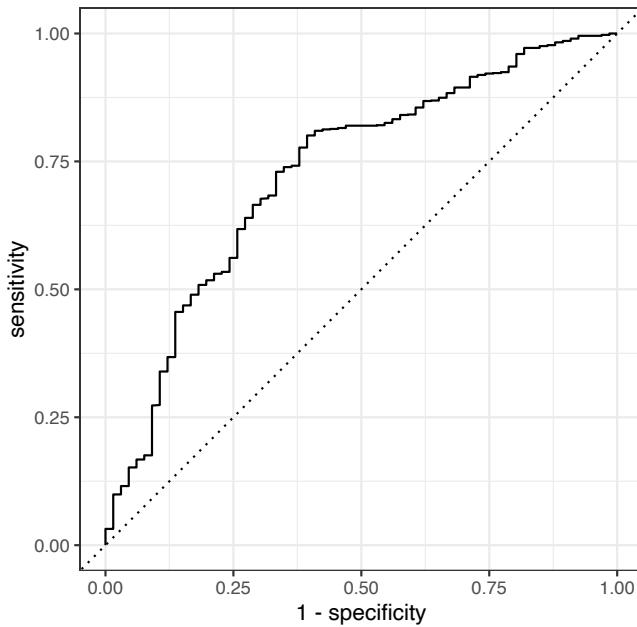
Métricas de LDA:

```
last_fit_lda %>%
  collect_metrics()
```

```
## # A tibble: 2 x 4
##   .metric  .estimator .estimate .config
##   <chr>    <chr>      <dbl> <chr>
## 1 accuracy binary     0.941 Preprocessor1_Model1
## 2 roc_auc  binary     0.728 Preprocessor1_Model1
```

curva ROC:

```
lda_predicciones <- last_fit_lda %>%
  collect_predictions()
lda_predicciones %>%
  roc_curve(truth = Purchase, estimate = .pred_No) %>%
  autoplot()
```



y matriz de confusión:

```
lda_predicciones %>%
  conf_mat(truth = Purchase, estimate = .pred_class)

##           Truth
## Prediction   No   Yes
##           No 1091    61
##           Yes     8     5
```

7.4.3. Análisis Discriminante Cuadrático

En este caso usaremos otro generador (*klaR*).

```
library(klaR)
```

Nota: El argumento *frac_common_cov=1* permite hacer LDA en lugar de QDA.

```
modelo_qda <- discrim_regularized(frac_common_cov = 0) %>%
  set_engine("klaR") %>%
  set_mode("classification")

qda_wf <- workflow() %>%
  add_model(modelo_qda) %>%
  add_recipe(Caravan.recipe)

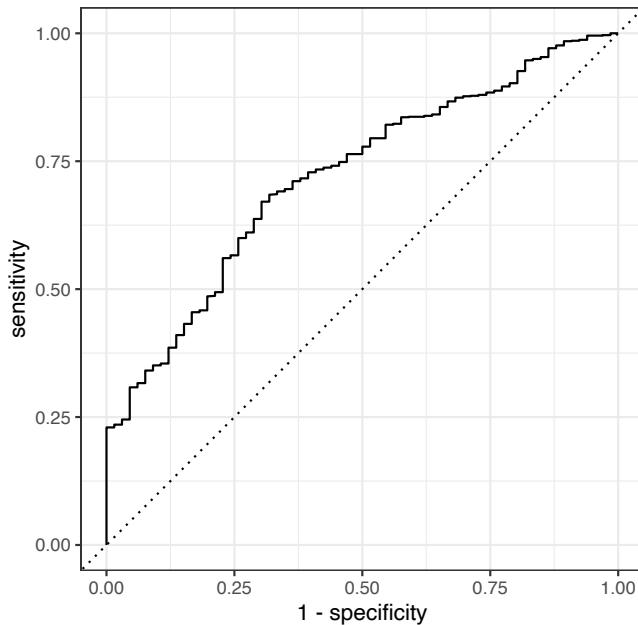
last_fit_qda <- qda_wf %>%
  last_fit(split = Caravan.split)

last_fit_qda %>%
  collect_metrics()

## # A tibble: 2 x 4
##   .metric  .estimator .estimate .config
##   <chr>    <chr>      <dbl> <chr>
## 1 accuracy binary      0.903 Preprocessor1_Model1
## 2 roc_auc  binary      0.720 Preprocessor1_Model1

qda_predicciones <- last_fit_qda %>%
  collect_predictions()
```

```
qda_predicciones %>%
  roc_curve(truth = Purchase, estimate = .pred_No) %>%
  autoplot()
```



```
qda_predicciones %>%
  conf_mat(truth = Purchase, estimate = .pred_class)

##          Truth
## Prediction  No  Yes
##       No 1042   56
##       Yes   57   10
```

7.4.4. K vecinos más cercanos

En el caso del KNN se va a seleccionar el número de vecinos a través de validación cruzada, usando como métrica el AUC.

Este método requiere la siguiente librería para funcionar

```
library(kknn)
```

Primero definimos los conjuntos bajo el k-fold:

```
set.seed(178)

Caravan.folds <- vfold_cv(Caravan.training, v = 5,
    strata = Purchase)
```

y definimos el modelo KNN y el flujo de trabajo:

```
modelo_knn <- nearest_neighbor(neighbors = tune()) %>%
    set_engine("kknn") %>%
    set_mode("classification")

knn_wf <- workflow() %>%
    add_model(modelo_knn) %>%
    add_recipe(Caravan.recipe)
```

Definimos una grilla de posibles valores de # de vecinos que usaremos en el k-fold:

```
k_grid <- tibble(neighbors = c(50, 75, 100, 125, 150,
    175, 200, 225))

set.seed(178)
knn_tuning <- knn_wf %>%
    tune_grid(resamples = Caravan.folds, grid = k_grid)
```

y se selecciona el modelo con el mejor AUC:

```
knn_tuning %>%
    show_best("roc_auc")
```

```
## # A tibble: 5 x 7
##   neighbors .metric .estimator  mean     n std_err .config
##       <dbl> <chr>    <chr>     <dbl> <int>   <dbl> <chr>
## 1       225 roc_auc binary    0.711     5  0.0145 Preprocessor1_Model8
## 2       200 roc_auc binary    0.709     5  0.0151 Preprocessor1_Model7
## 3       175 roc_auc binary    0.707     5  0.0161 Preprocessor1_Model6
```

```
## 4      150 roc_auc binary    0.705      5  0.0177 Preprocessor1_Model5
## 5      125 roc_auc binary    0.701      5  0.0189 Preprocessor1_Model4
```

mejor modelo y actualización del flujo de trabajo:

```
mejor_knn <- knn_tuning %>%
  select_best(metric = "roc_auc")

final_knn_wf <- knn_wf %>%
  finalize_workflow(mejor_knn)
```

Ahora ajustamos el modelo y analizamos su rendimiento con los conjuntos de entrenamiento y prueba iniciales:

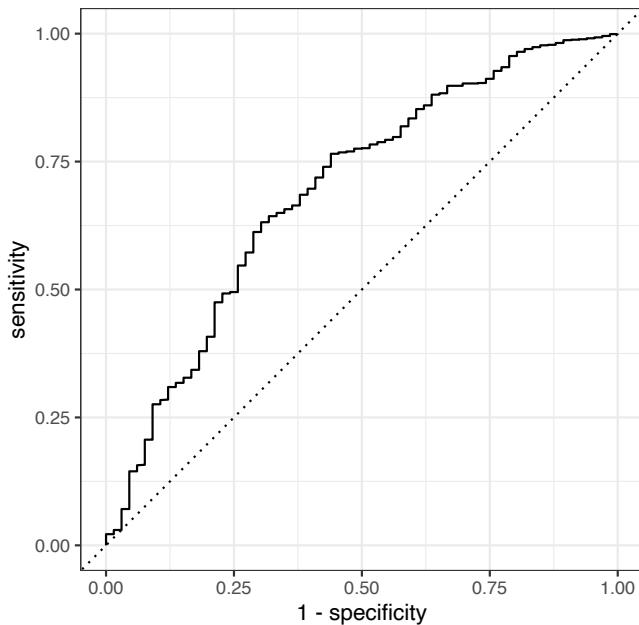
```
last_fit_knn <- final_knn_wf %>%
  last_fit(split = Caravan.split)

last_fit_knn %>%
  collect_metrics()
```

```
## # A tibble: 2 x 4
##   .metric  .estimator .estimate .config
##   <chr>    <chr>        <dbl> <chr>
## 1 accuracy binary     0.943 Preprocessor1_Model1
## 2 roc_auc  binary     0.693 Preprocessor1_Model1

knn_predicciones <- last_fit_knn %>%
  collect_predictions()

knn_predicciones %>%
  roc_curve(truth = Purchase, estimate = .pred_No) %>%
  autoplot()
```



```
knn_predicciones %>%
  conf_mat(truth = Purchase, estimate = .pred_class)
```

```
##           Truth
## Prediction   No   Yes
##           No 1099    66
##           Yes     0    0
```

Máquinas de soporte vectorial

```
modelo_svm <- svm_poly(degree = 1) %>%
  set_mode("classification") %>%
  set_engine("kernlab", scaled = FALSE)

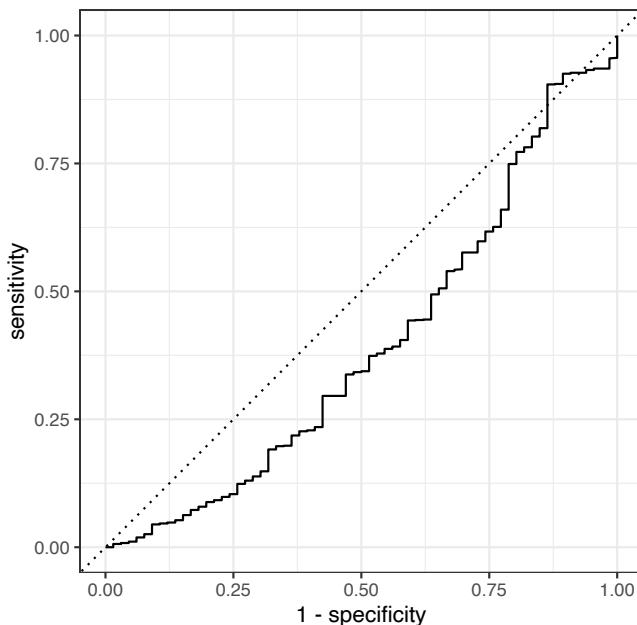
svm_wf <- workflow() %>%
  add_model(modelo_svm) %>%
  add_recipe(Caravan.recipe)

last_fit_svm <- svm_wf %>%
  last_fit(split = Caravan.split)
```

```
last_fit_svm %>%
  collect_metrics()

## # A tibble: 2 x 4
##   .metric  .estimator .estimate .config
##   <chr>    <chr>        <dbl> <chr>
## 1 accuracy binary      0.943 Preprocessor1_Model1
## 2 roc_auc  binary      0.395 Preprocessor1_Model1

svm_prediccciones <- last_fit_svm %>%
  collect_predictions()
svm_prediccciones %>%
  roc_curve(truth = Purchase, estimate = .pred_No) %>%
  autoplot()
```



```
svm_prediccciones %>%
  conf_mat(truth = Purchase, estimate = .pred_class)

##          Truth
## Prediction  No  Yes
```

```
##          No    1099     66
##          Yes      0      0

modelo_svm <- svm_poly(degree = 2) %>%
  set_mode("classification") %>%
  set_engine("kernlab", scaled = FALSE)

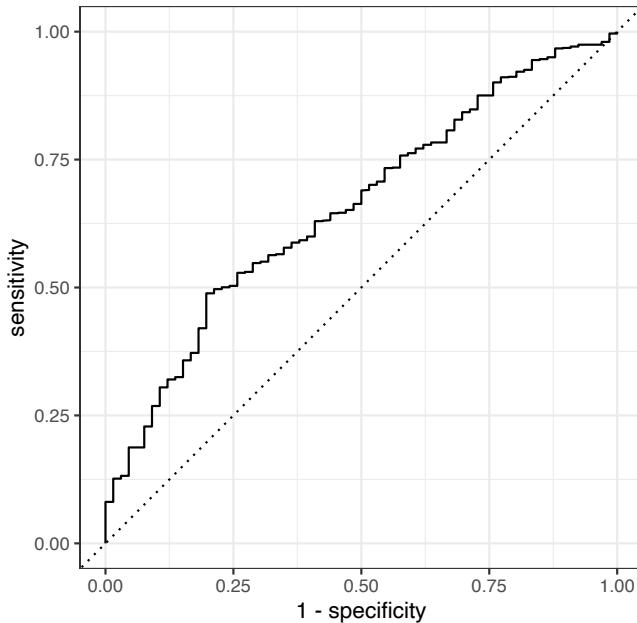
svm_wf <- workflow() %>%
  add_model(modelo_svm) %>%
  add_recipe(Caravan.recipe)

last_fit_svm <- svm_wf %>%
  last_fit(split = Caravan.split)

last_fit_svm %>%
  collect_metrics()
```

```
## # A tibble: 2 x 4
##   .metric  .estimator .estimate .config
##   <chr>    <chr>        <dbl> <chr>
## 1 accuracy binary       0.872 Preprocessor1_Model1
## 2 roc_auc  binary       0.656 Preprocessor1_Model1

svm_predicciones <- last_fit_svm %>%
  collect_predictions()
svm_predicciones %>%
  roc_curve(truth = Purchase, estimate = .pred_No) %>%
  autoplot()
```



```

svm_predicciones %>%
  conf_mat(truth = Purchase, estimate = .pred_class)

##           Truth
## Prediction   No  Yes
##       No 1001   51
##       Yes   98   15

modelo_svm <- svm_poly(degree = 3) %>%
  set_mode("classification") %>%
  set_engine("kernlab", scaled = FALSE)

svm_wf <- workflow() %>%
  add_model(modelo_svm) %>%
  add_recipe(Caravan.recipe)

last_fit_svm <- svm_wf %>%
  last_fit(split = Caravan.split)

last_fit_svm %>%
  collect_metrics()

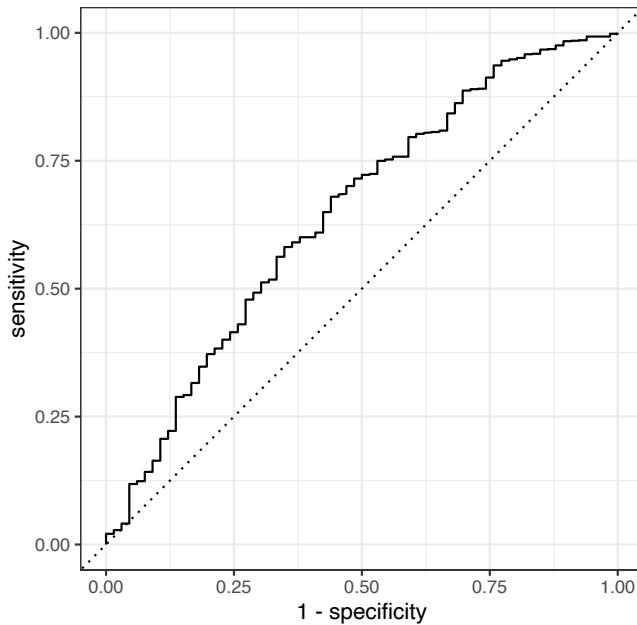
```

```

## # A tibble: 2 x 4
##   .metric  .estimator .estimate .config
##   <chr>    <chr>        <dbl> <chr>
## 1 accuracy binary      0.886 Preprocessor1_Model1
## 2 roc_auc  binary      0.645 Preprocessor1_Model1

svm_predicciones <- last_fit_svm %>%
  collect_predictions()
svm_predicciones %>%
  roc_curve(truth = Purchase, estimate = .pred_No) %>%
  autoplot()

```



```

svm_predicciones %>%
  conf_mat(truth = Purchase, estimate = .pred_class)

##           Truth
## Prediction  No  Yes
##       No 1016   50
##       Yes   83   16

```

```
modelo_svm <- svm_poly(degree = 4) %>%
  set_mode("classification") %>%
  set_engine("kernlab", scaled = FALSE)

svm_wf <- workflow() %>%
  add_model(modelo_svm) %>%
  add_recipe(Caravan.recipe)

last_fit_svm <- svm_wf %>%
  last_fit(split = Caravan.split)

last_fit_svm %>%
  collect_metrics()

## NULL

svm_predicciones <- last_fit_svm %>%
  collect_predictions()
svm_predicciones %>%
  roc_curve(truth = Purchase, estimate = .pred_No) %>%
  autoplot()

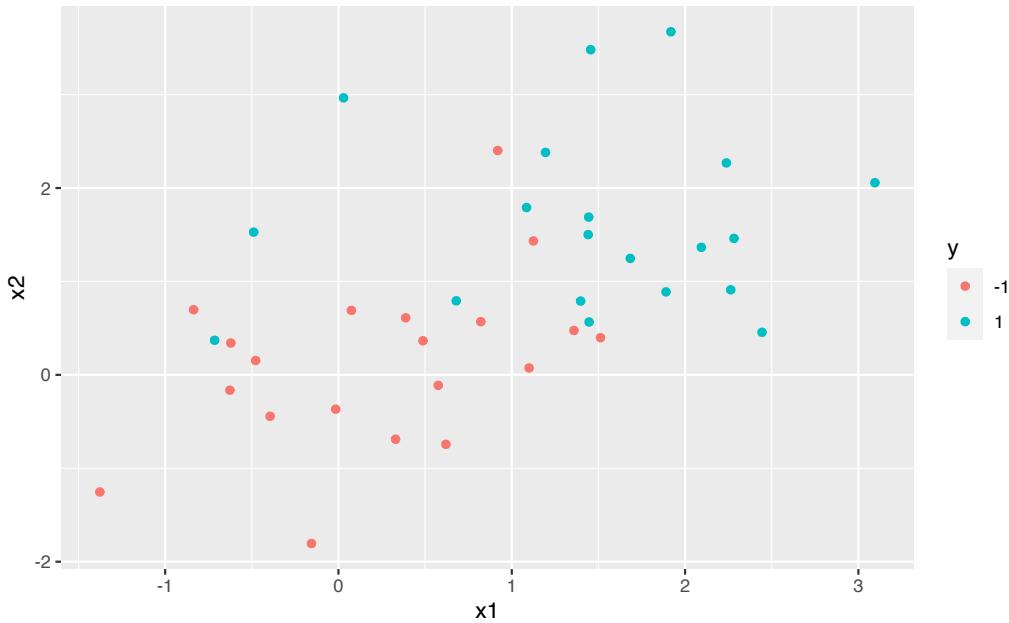
## Error in `chr_as_locations()`:
## ! Can't subset columns that don't exist.
## x Column '.pred_No' doesn't exist.

svm_predicciones %>%
  conf_mat(truth = Purchase, estimate = .pred_class)

## Error:
## ! object 'Purchase' not found

library(kernlab)
set.seed(1)
sim_data <- tibble(x1 = rnorm(40), x2 = rnorm(40),
  y = factor(rep(c(-1, 1), 20))) %>%
  mutate(x1 = ifelse(y == 1, x1 + 1.5, x1), x2 = ifelse(y ==
    1, x2 + 1.5, x2))
```

```
ggplot(sim_data, aes(x1, x2, color = y)) + geom_point()
```



```
svm_linear_spec <- svm_poly(degree = 1) %>%
  set_mode("classification") %>%
  set_engine("kernlab", scaled = FALSE)
```

```
svm_linear_fit <- svm_linear_spec %>%
  set_args(cost = 10) %>%
  fit(y ~ ., data = sim_data)
```

```
svm_linear_fit
```

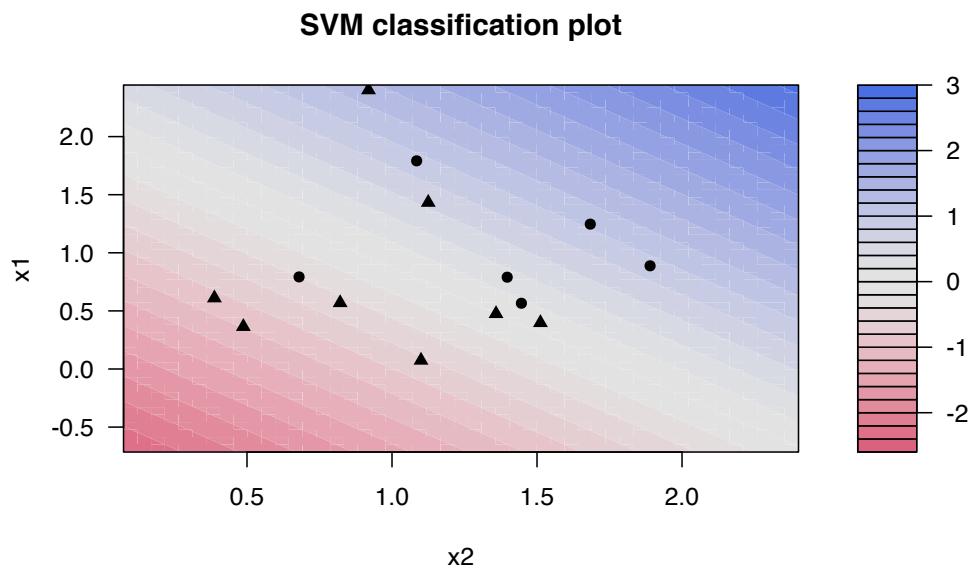
```
## parsnip model object
##
## Support Vector Machine object of class "ksvm"
##
## SV type: C-svc (classification)
## parameter : cost C = 10
##
## Polynomial kernel function.
```

```

## Hyperparameters : degree = 1 scale = 1 offset = 1
##
## Number of Support Vectors : 17
##
## Objective Function Value : -152.0188
## Training error : 0.125
## Probability model included.

library(kernlab)
svm_linear_fit %>%
  extract_fit_engine() %>%
  plot()

```



Si reducimos el costo

```

svm_linear_fit <- svm_linear_spec %>%
  set_args(cost = 0.1) %>%
  fit(y ~ ., data = sim_data)

svm_linear_fit

```

```
## parsnip model object
##
## Support Vector Machine object of class "ksvm"
##
## SV type: C-svc  (classification)
## parameter : cost C = 0.1
##
## Polynomial kernel function.
## Hyperparameters : degree =  1  scale =  1  offset =  1
##
## Number of Support Vectors : 25
##
## Objective Function Value : -2.0376
## Training error : 0.15
## Probability model included.

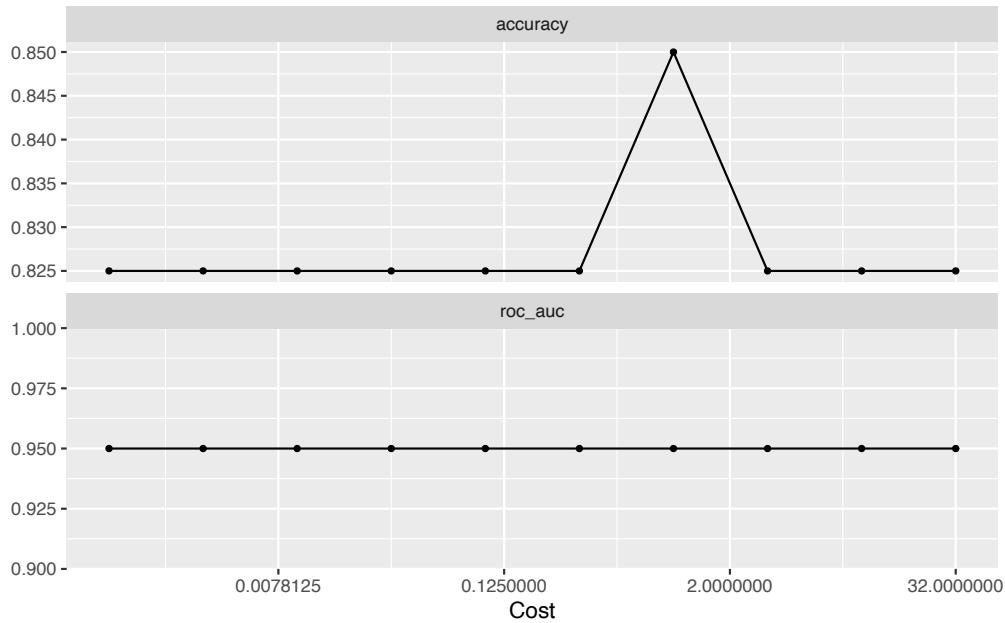
svm_linear_wf <- workflow() %>%
  add_model(svm_linear_spec %>%
    set_args(cost = tune())))
  %>%
  add_formula(y ~ .)

set.seed(1234)
sim_data_fold <- vfold_cv(sim_data, strata = y)

param_grid <- grid_regular(cost(), levels = 10)

tune_res <- tune_grid(svm_linear_wf, resamples = sim_data_fold,
  grid = param_grid)

autoplot(tune_res)
```



```
best_cost <- select_best(tune_res, metric = "accuracy")

svm_linear_final <- finalize_workflow(svm_linear_wf,
    best_cost)

(svm_linear_fit <- svm_linear_final %>%
    fit(sim_data))
```

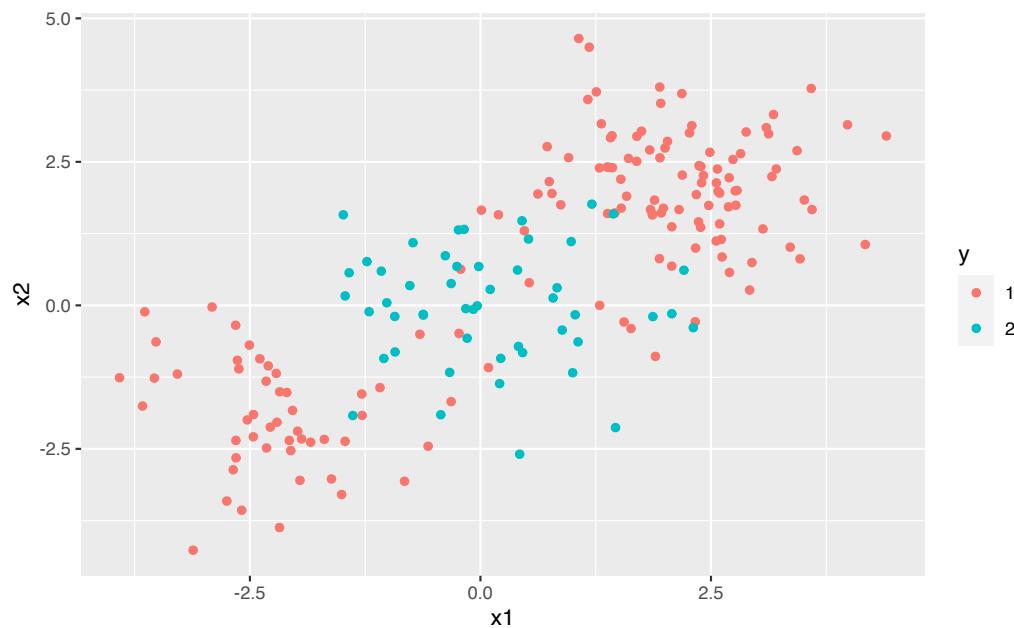
```
## == Workflow [trained] -----
## Preprocessor: Formula
## Model: svm_poly()
##
## -- Preprocessor -----
## y ~ .
##
## -- Model -----
## Support Vector Machine object of class "ksvm"
##
## SV type: C-svc  (classification)
## parameter : cost C = 1
```

```
##
## Polynomial kernel function.
## Hyperparameters : degree = 1 scale = 1 offset = 1
##
## Number of Support Vectors : 18
##
## Objective Function Value : -16.025
## Training error : 0.15
## Probability model included.
```

Kernels en alta dimensión

```
set.seed(1)
sim_data2 <- tibble(x1 = rnorm(200) + rep(c(2, -2,
  0), c(100, 50, 50)), x2 = rnorm(200) + rep(c(2,
  -2, 0), c(100, 50, 50)), y = factor(rep(c(1, 2),
  c(150, 50))))
```

```
sim_data2 %>%
  ggplot(aes(x1, x2, color = y)) + geom_point()
```



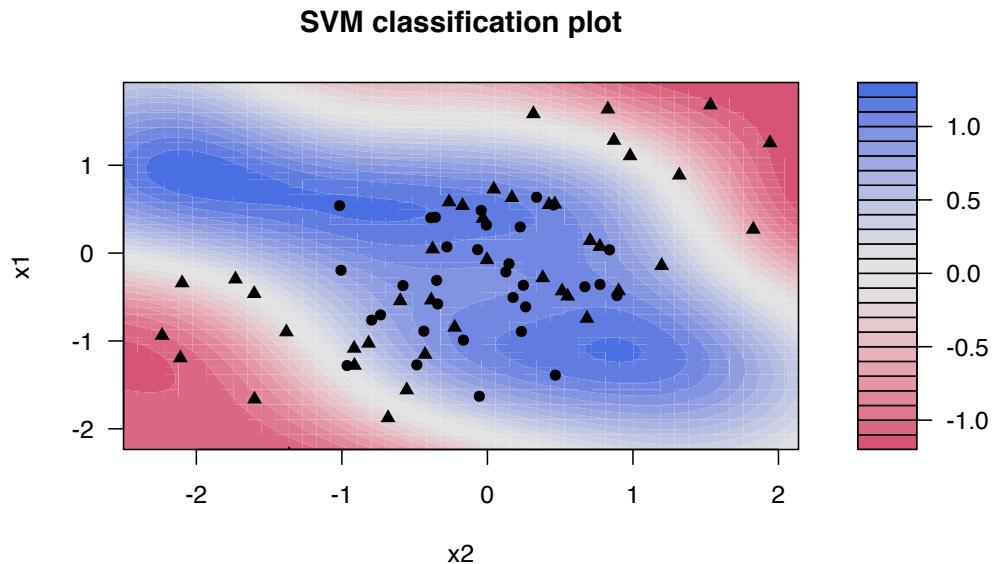
```

svm_rbf_spec <- svm_rbf() %>%
  set_mode("classification") %>%
  set_engine("kernlab")

svm_rbf_fit <- svm_rbf_spec %>%
  fit(y ~ ., data = sim_data2)

svm_rbf_fit %>%
  extract_fit_engine() %>%
  plot()

```



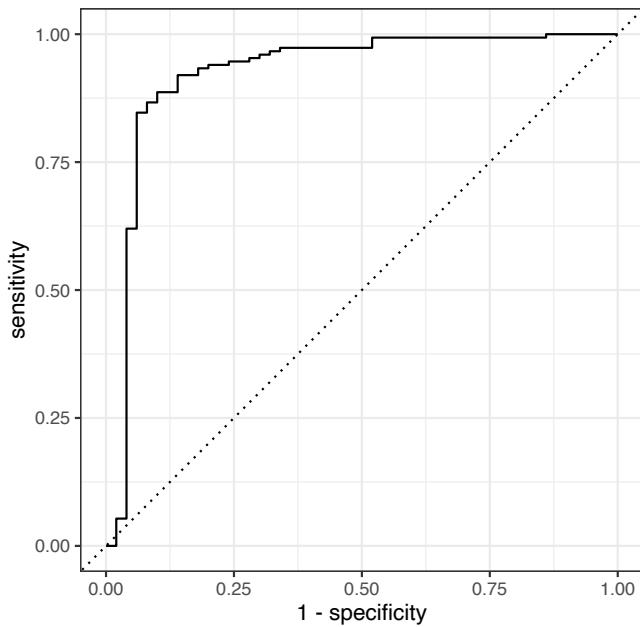
```

set.seed(2)
sim_data2_test <- tibble(x1 = rnorm(200) + rep(c(2,
-2, 0), c(100, 50, 50)), x2 = rnorm(200) + rep(c(2,
-2, 0), c(100, 50, 50)), y = factor(rep(c(1, 2),
c(150, 50))))
```

```

augment(svm_rbf_fit, new_data = sim_data2_test) %>%
  roc_curve(truth = y, estimate = .pred_1) %>%
  autoplot()

```



7.5. Ejercicios

- Del libro (James y col. 2013)
 - Capítulo 4: 10, 11, 13.

Capítulo 8

Análisis en componentes principales

8.1. Representación gráfica

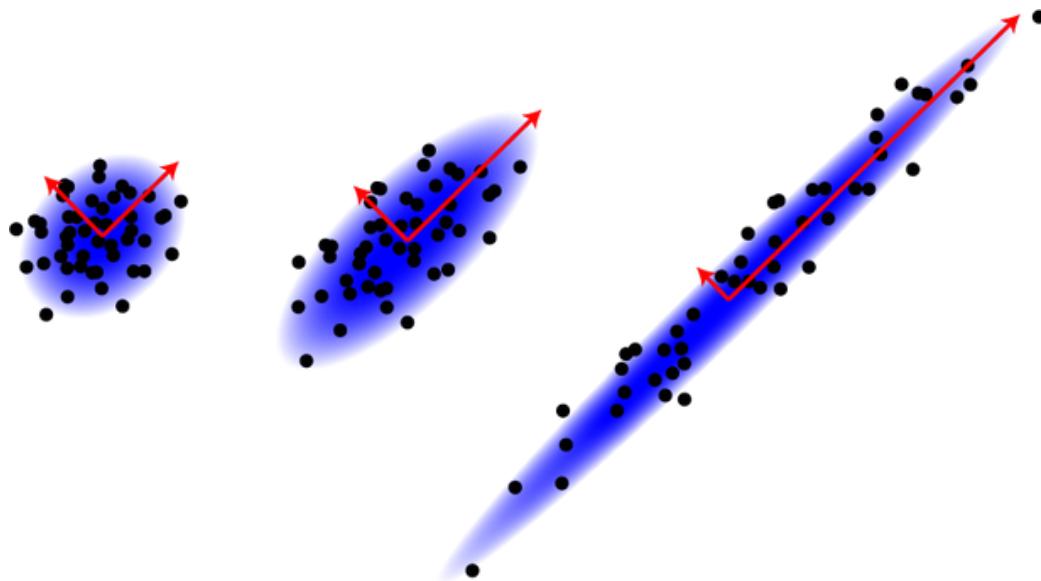


Figura 8.1: Tomado de [The shape of data](#)

8.2. Aprendizaje no-supervisado

Al contrario de los métodos que se han estudiado de regresión y clasificación, en este caso no hay variable dependiente, y el conjunto de datos está compuesto de p variables o características y n observaciones.

El principal objetivo del aprendizaje no-supervisado no es la predicción, sino en el *análisis de datos* por sí mismo, es decir se quiere buscar patrones o relaciones interesantes dentro de la tabla de datos: por ejemplo la visualización de datos o la identificación de subgrupos en los datos. (Análisis Exploratorio de Datos)

En el caso de aprendizaje no-supervisado, no es posible verificar o validar los métodos adoptados.

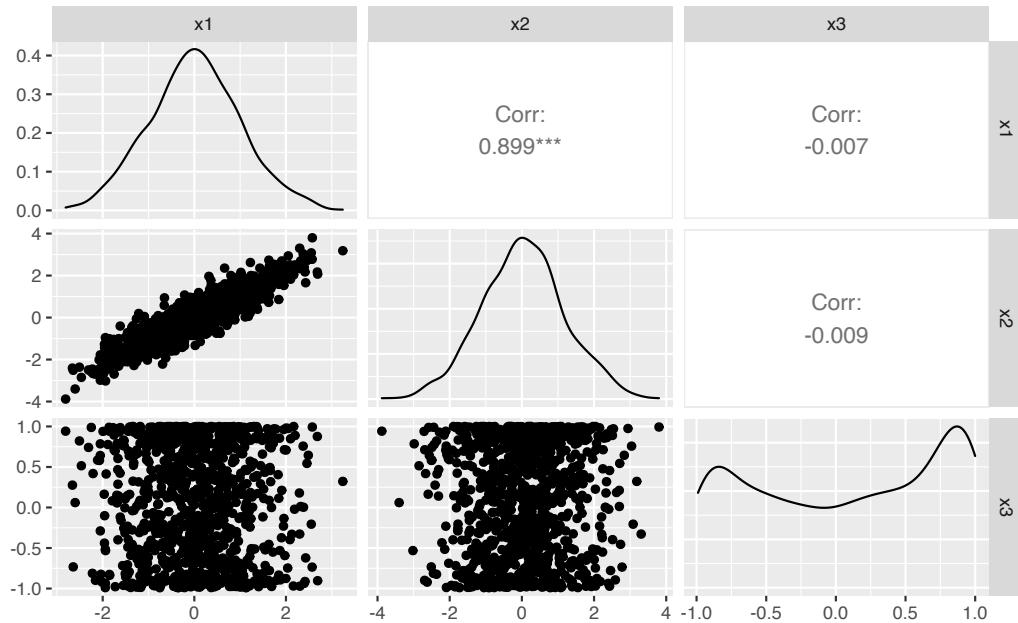
Si se quiere seleccionar la mejor proyección de 2 variables de una nube de puntos X_1, \dots, X_p , se debe hacer $\binom{p}{2}$ gráficos de dispersión. Un criterio de búsqueda es seleccionar la que tenga mayor información, en el sentido de mayor variabilidad.

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, 1)
x2 <- x1 + rnorm(1000, 0, 0.5)
x3 <- cos(runif(1000, -3, 3))

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. El objetivo es crear una serie de variables sintéticas Z_1, \dots, Z_k con $k \leq p$, de modo que posean la misma (o casi) información de las variables originales.

8.3. Primer componente principal

El primer componente se define como el promedio ponderado de las variables originales X_1, \dots, X_p

$$Z_{i1} = \phi_{11}x_{i1} + \phi_{21}x_{i2} + \dots + \phi_{p1}x_{ip}; \quad \text{con } \sum_{j=1}^p \phi_{j1} = 1$$

El objetivo es que el vector Z_1 tenga varianza máxima. Al vector $\phi_1 = (\phi_{11}, \phi_{21}, \dots, \phi_{p1})$ se le llama *pesos o cargas* y funciona para promediar cada uno de las columnas de X . Pensemos que ϕ_1 es la dirección en el espacio característico en \mathbb{R}^p en donde los datos tienen la máxima varianza.

En nuestro caso defina $X = (X_1, \dots, X_p)_{n \times p}$ la *matriz de diseño* donde cada columna tiene media 0. En este caso queremos resolver el problema

$$\hat{\phi}_1 = \operatorname{argmax}_{\|\phi_1\|_2^2=1} \left\{ \frac{1}{n} \sum_{i=1}^n \left(\sum_{j=1}^p \phi_{j1} X_{ij} \right)^2 \right\}$$

Note que la restricción de minimización se puede reescribir como $\|\phi_1\|_2^2 = \sum_{j=1}^p \phi_{j1}^2 = 1$

Esta última expresión se podría reescribir de forma matricial como

$$\begin{aligned}\hat{\phi}_1 &= \operatorname{argmax}_{\|\phi_1\|_2^2=1} \left\{ \frac{1}{n} \phi_1^\top X^\top X \phi_1 \right\} \\ \hat{\phi}_1 &= \operatorname{argmax}_{\|\phi_1\|_2^2=1} \left\{ \phi_1^\top V \phi_1 \right\}\end{aligned}$$

donde $\phi_1 = (\phi_{11}, \phi_{21}, \dots, \phi_{p1})$

Ahora si maximizamos esta expresión usando multiplicadores de lagrange, tenemos que lo siguiente:

$$\begin{aligned}L(\phi_1, \lambda) &= \phi_1^\top V \phi_1 - \lambda(\phi_1^\top \phi_1 - 1) \\ \frac{\partial L}{\partial \lambda} &= \phi_1^\top \phi_1 - 1 \\ \frac{\partial L}{\partial \phi_1} &= 2V\phi_1 - 2\lambda\phi_1\end{aligned}$$

Igualando a 0 las condiciones obtenemos que

$$\begin{aligned}\phi_1^\top \phi_1 &= 1 \\ V\phi_1 &= \lambda V.\end{aligned}$$

Entonces el valor óptimo de ϕ_1 es el componente principal de $\frac{1}{n}X^\top X = \text{Cov}(X)$ cuyo λ sea más grande si las columnas de X son centradas.

8.4. Segunda componente principal

La segunda componente se escribe de la forma

$$Z_2 := \phi_{12}x_1 + \phi_{22}x_2 + \cdots + \phi_{p2}x_p.$$

En este caso debemos resolver el mismo problema de maximización que antes.

$$\underset{\|\phi_2\|_2^2=1}{\operatorname{argmax}} \left\{ \frac{1}{n} \sum_{i=1}^n \left(\sum_{j=1}^p \phi_{j2} X_{ij} \right)^2 \right\}$$

La diferencia es que se quiere que el segundo componente sea ortogonal al primero. Es decir, 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 donde queremos maximizar

$$\begin{aligned} \hat{\phi}_2 &= \underset{\|\phi_2\|_2^2=1}{\operatorname{argmax}} \left\{ \frac{1}{n} \phi_2^\top \tilde{X}_2^\top \tilde{X}_2 \phi_2 \right\} \\ \hat{\phi}_2 &= \underset{\|\phi_2\|_2^2=1}{\operatorname{argmax}} \left\{ \phi_2^\top \tilde{V}_2 \phi_2 \right\} \end{aligned}$$

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.

8.5. Círculo 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 $\text{PC}_{j'}$.

8.6. Volvamos a nuestro ejemplo

```
library("factoextra")
library("FactoMineR")

(PC1 <- prcomp(cbind(x1, x2, x3), center = TRUE))

## Standard deviations (1, ..., p=3):
## [1] 1.4815788 0.6822887 0.3366470
##
## Rotation (n x k) = (3 x 3):
##          PC1        PC2        PC3
## x1  0.646510691 -0.004875673 -0.762889346
## x2  0.762889958 -0.002132344  0.646524837
## x3 -0.004778986 -0.999985840  0.002341018
```

```
str(PC1)
```

```
## List of 5
## $ sdev    : num [1:3] 1.482 0.682 0.337
## $ rotation: num [1:3, 1:3] 0.64651 0.76289 -0.00478 -0.00488 -0.00213 ...
## ..- attr(*, "dimnames")=List of 2
## ... $ : chr [1:3] "x1" "x2" "x3"
## ... $ : chr [1:3] "PC1" "PC2" "PC3"
## $ center  : Named num [1:3] 0.0161 0.0374 0.0814
## ..- attr(*, "names")= chr [1:3] "x1" "x2" "x3"
## $ scale   : logi FALSE
## $ x       : num [1:1000, 1:3] -1.2102 -0.7577 2.1474 0.0117 -0.8303 ...
## ..- attr(*, "dimnames")=List of 2
## ... $ : NULL
## ... $ : chr [1:3] "PC1" "PC2" "PC3"
## - attr(*, "class")= chr "prcomp"
(PC2 <- princomp(cbind(x1, x2, x3)))
```

```
## Call:
## princomp(x = cbind(x1, x2, x3))
##
## Standard deviations:
##      Comp.1     Comp.2     Comp.3
## 1.4808378 0.6819475 0.3364786
##
## 3 variables and 1000 observations.
str(PC2)
```

```
## List of 7
## $ sdev    : Named num [1:3] 1.481 0.682 0.336
## ..- attr(*, "names")= chr [1:3] "Comp.1" "Comp.2" "Comp.3"
## $ loadings: 'loadings' num [1:3, 1:3] 0.64651 0.76289 -0.00478 0.00488 0.00213 ...
## ..- attr(*, "dimnames")=List of 2
## ... $ : chr [1:3] "x1" "x2" "x3"
## ... $ : chr [1:3] "Comp.1" "Comp.2" "Comp.3"
## $ center  : Named num [1:3] 0.0161 0.0374 0.0814
## ..- attr(*, "names")= chr [1:3] "x1" "x2" "x3"
```

```

## $ scale   : Named num [1:3] 1 1 1
## ..- attr(*, "names")= chr [1:3] "x1" "x2" "x3"
## $ n.obs   : int 1000
## $ scores  : num [1:1000, 1:3] -1.2102 -0.7577 2.1474 0.0117 -0.8303 ...
## ..- attr(*, "dimnames")=List of 2
## ...$ : NULL
## ...$ : chr [1:3] "Comp.1" "Comp.2" "Comp.3"
## $ call    : language princomp(x = cbind(x1, x2, x3))
## - attr(*, "class")= chr "princomp"

p <- PCA(scale(cbind(x1, x2, x3)))

p$var$cor

##           Dim.1      Dim.2      Dim.3
## x1  0.97450087 0.009678028 -0.224174906
## x2  0.97451729 0.007743861  0.224178685
## x3 -0.01698022 0.999855732  0.000433625

p$var$cos2

##           Dim.1      Dim.2      Dim.3
## x1  0.949651947 9.366422e-05 5.025439e-02
## x2  0.949683950 5.996738e-05 5.025608e-02
## x3  0.000288328 9.997115e-01 1.880306e-07

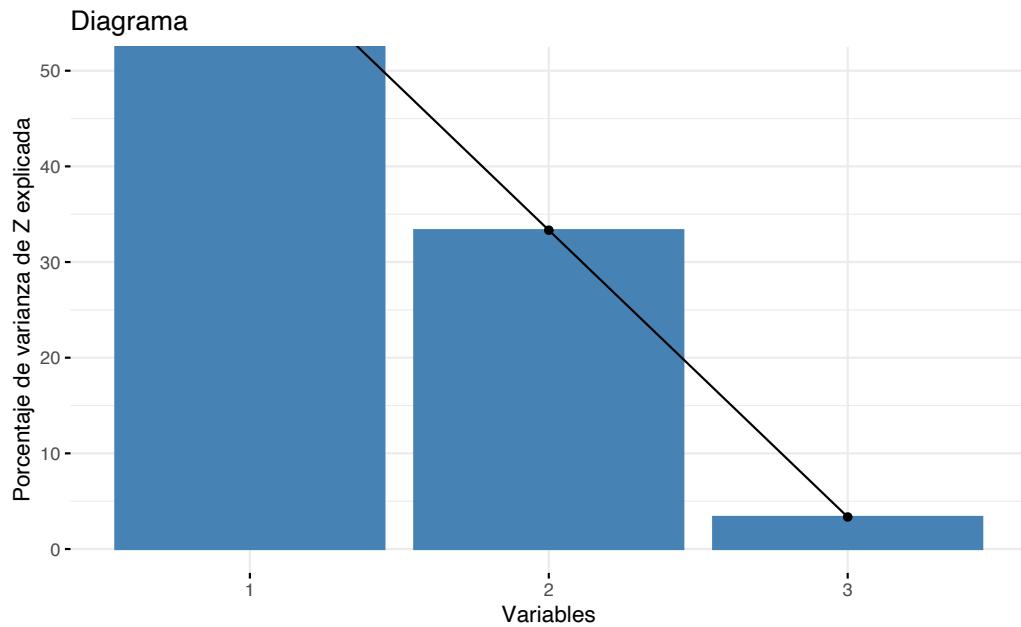
```

8.7. ¿Cuántos componentes usar?

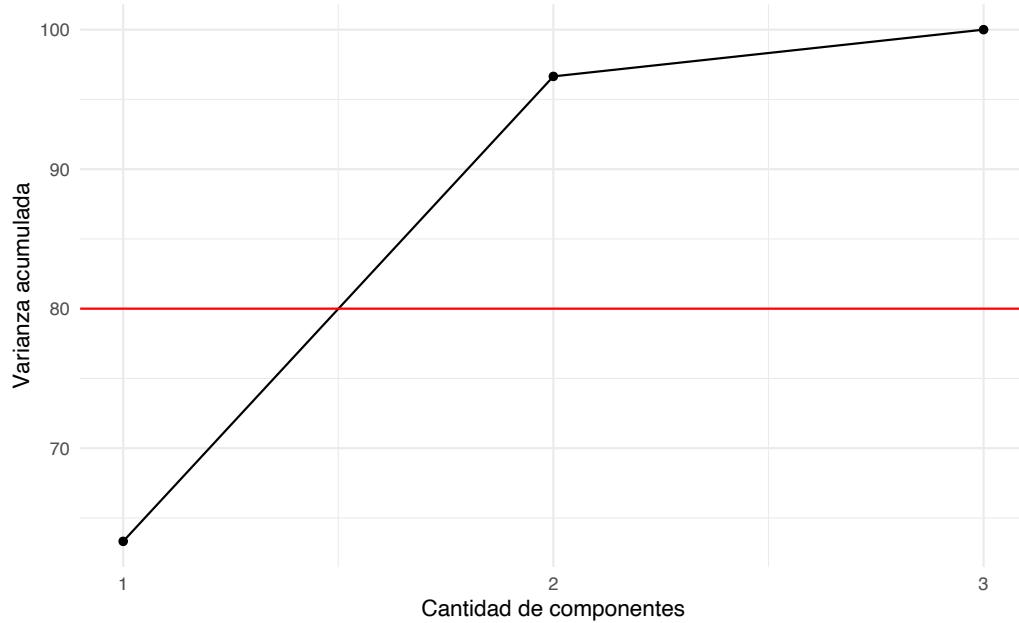
```

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)
```



8.8. Laboratorio

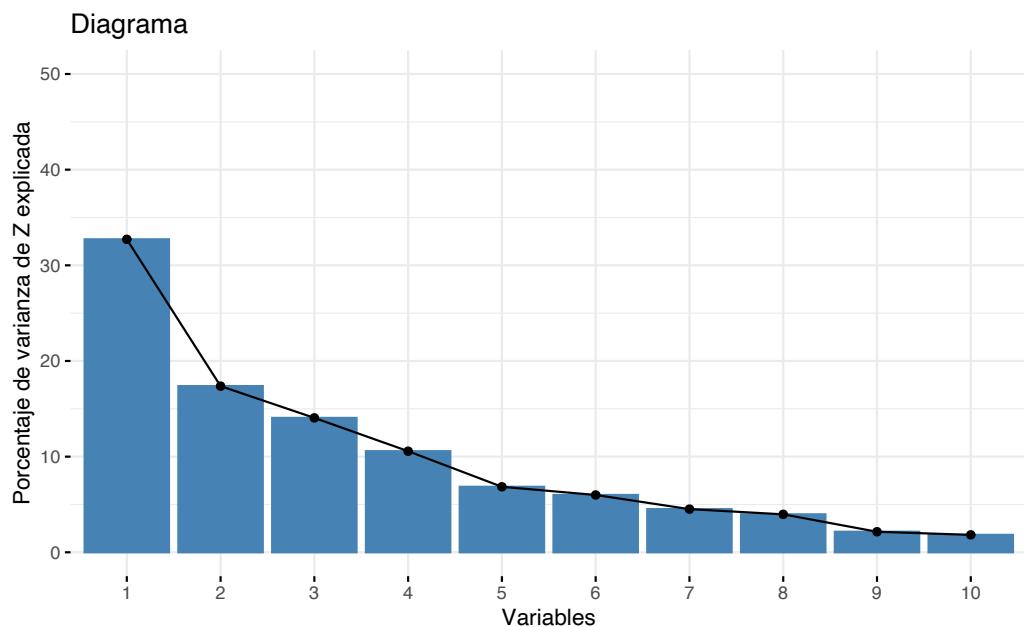
Vamos a usar los datos `decathlon` de FactomineR que representa los resultados de varios atletas en pruebas de decathlon en el 2004.

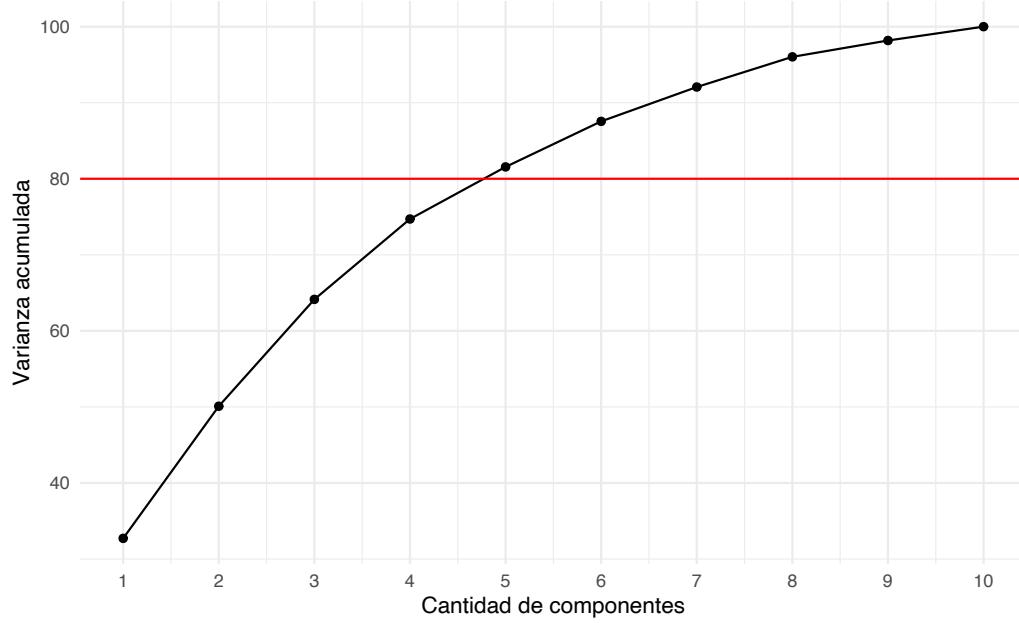
El objetivo es encontrar si hay patrones entre ciudad y tipos de crimen.

Exploración de datos Ejecute una exploración de datos

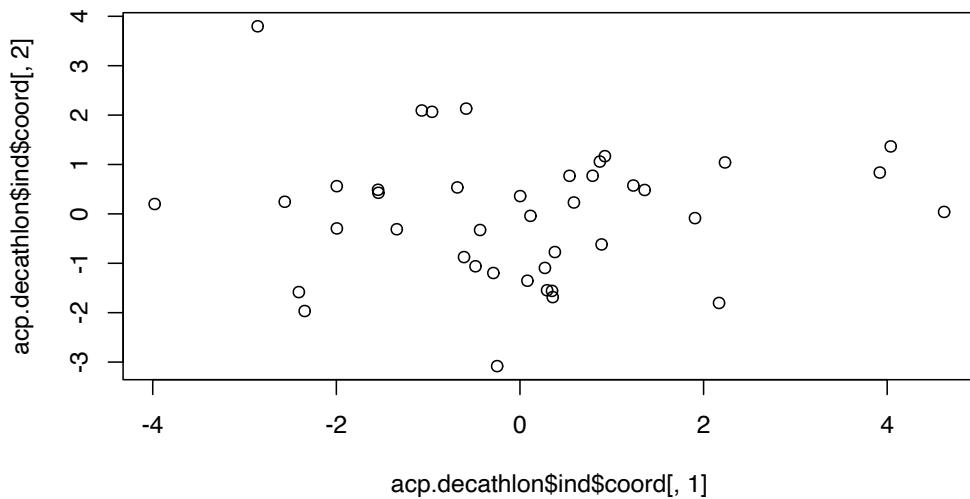
```
##          100m      Long.jump      Shot.put      High.jump      400m
##  Min.   :10.44   Min.   :6.61   Min.   :12.68   Min.   :1.850   Min.   :46.
##  1st Qu.:10.85  1st Qu.:7.03  1st Qu.:13.88  1st Qu.:1.920  1st Qu.:48.
##  Median :10.98  Median :7.30   Median :14.57   Median :1.950   Median :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
```

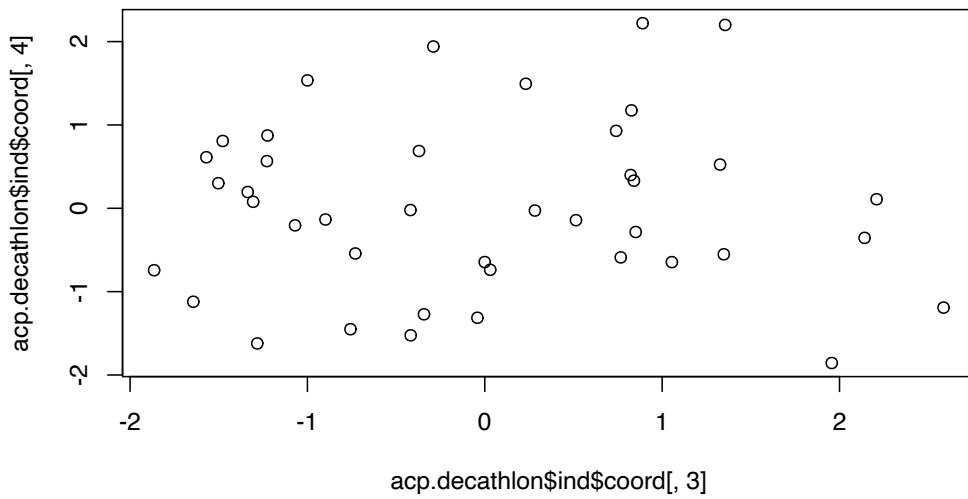




```
plot(acp.decathlon$ind$coord[, 1], acp.decathlon$ind$coord[,  
 2])
```



```
plot(acp.decathlon$ind$coord[, 3], acp.decathlon$ind$coord[, 4])
```



8.9. Ejercicios

- Del libro (James y col. 2013)
- Capítulo 10: 6, 8

Capítulo 9

Cálculo Bayesiano Computacional

9.1. Repaso de Estadística Bayesiana

9.1.1. Modelo de un parámetro

Vamos a considerar el ejemplo en la sección 3.3 del (Albert y col. 2009). En este caso se quiere estimar la tasa de éxito en transplantes de corazón en un hospital de EEUU. Suponga que en ese hospital hay n transplantes e y es el número de muertes en el transcurso de 30 días del transplante. Si se sabe el número esperado de muertes e a través de un modelo auxiliar, entonces un modelo sencillo para y es asumir que:

$$y \sim \text{Poisson}(e\lambda)$$

donde λ es la tasa de mortalidad por unidad de exposición y tiempo.

Solución clásica: Estimar $\hat{\lambda} = y/e$, pero el estimador es malo si hay pocas muertes observadas y .

Solución bayesiana: Considere una previa conjugada (gamma) para λ de la forma,

$$p(\lambda) \propto \lambda^{\alpha-1} \exp(-\beta\lambda).$$

La ventaja de los modelos bayesianos es que se pueden integrar información externa al modelo. Supongamos que se cuenta con información externa de un grupo pequeño de hospitales con condiciones similares a la del hospital de interés, es decir se cuenta con muertes z_j y exposición o_j para diez hospitales ($j = 1, \dots, 10$). Asumamos que cada hospital tiene la distribución de sus muertes de la forma,

$$z_j \sim \text{Poisson}(o_j\lambda).$$

Entonces, asignamos una previa no-informativa a $p(\lambda) \propto \lambda^{-1}$ (cuando $\alpha = 0$ y $\lambda = 0$) y se obtiene un previa actualiza con todos los hospitales para λ de la forma,

$$p(\lambda) \propto \lambda^{\sum_{j=1}^{10} z_j - 1} \exp\left(-\lambda \sum_{j=1}^{10} o_j\right)$$

Suponga que $\alpha := \sum z_j = 16$ y $\beta := \sum o_j = 15174$. Si para el hospital de interés y_{obs} es el número observado de muertes y e es la exposición entonces la distribución posterior de λ es:

$$g(\lambda|y_{obs}) \sim \Gamma(\alpha + y_{obs}, \beta + e)$$

y la densidad predictiva de y es (Ejercicio):

$$f(y) = \frac{f(y|\lambda)p(\lambda)}{g(\lambda|y_{obs})}$$

donde $f(y|\lambda) \sim \text{Poisson}(e\lambda)$ (verosimilitud).

Supongamos que existen dos posibles hospitales:

- Hospital A: Se observa una muerte con 66 personas expuestas. Cálculo de la densidad posterior y densidad predictiva con $\lambda = \alpha/\beta$:

```
alpha <- 16
beta <- 15174
yobs <- 1
ex <- 66
y <- 0:10
```

```

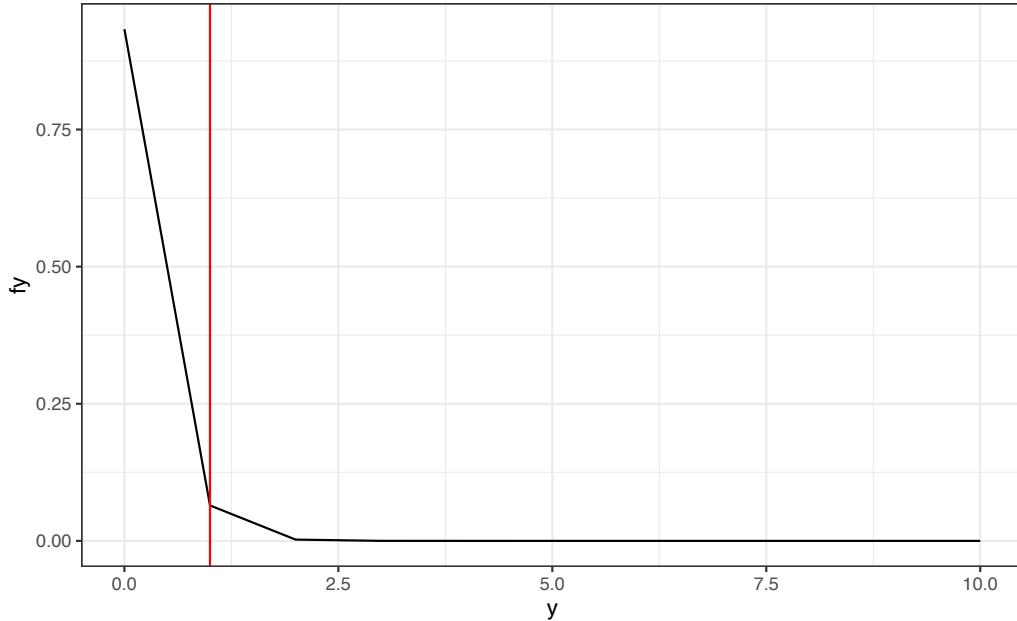
lam <- alpha/beta

## f(y/lambda) p(lambda) / g(lambda/y_obs)
fy <- dpois(y, lam * ex) * dgamma(lam, shape = alpha,
  rate = beta)/dgamma(lam, shape = alpha + y, rate = beta +
  ex)

dpred <- tibble(y, fy)

ggplot(dpred) + geom_line(mapping = aes(x = y, y = fy)) +
  geom_vline(xintercept = yobs, col = "red") + theme_bw()

```



por lo tanto una muerte no es un valor inusual en el comportamiento de muertes bajo transplantes. La comparación de las densidades posterior y previa de lambda:

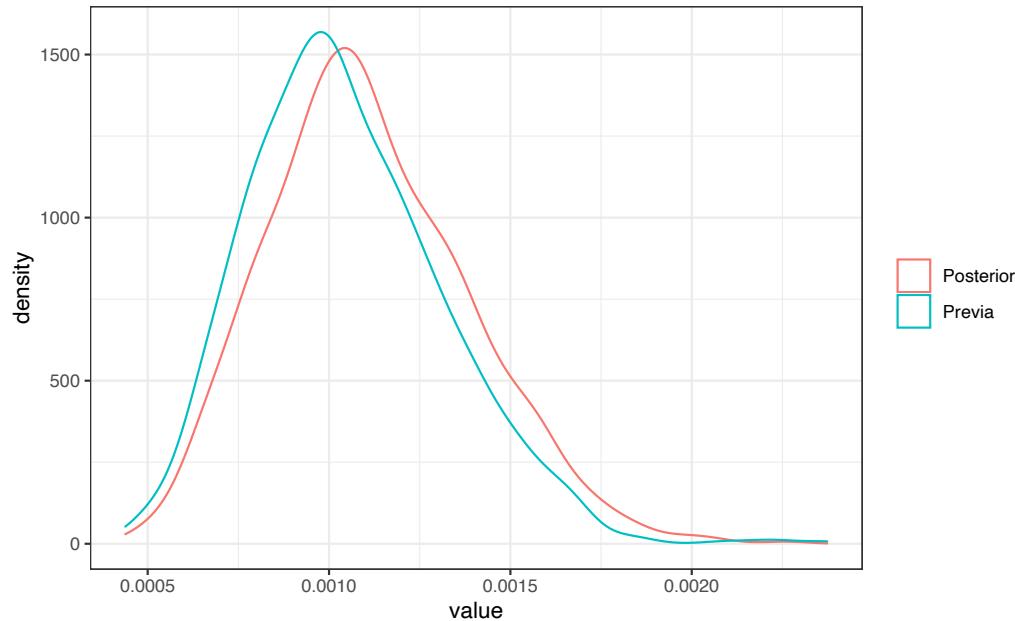
```

set.seed(1)
lambda_prev <- rgamma(1000, shape = alpha, rate = beta)
lambda_post <- rgamma(1000, shape = alpha + yobs, rate = beta +
  ex)

```

```
datoslambda <- tibble(Previa = lambda_prev, Posterior = lambda_post) %>%
  pivot_longer(cols = everything())

ggplot(data = datoslambda) + geom_density(mapping = aes(x = value,
  color = name)) + theme_bw() + theme(legend.title = element_blank())
```



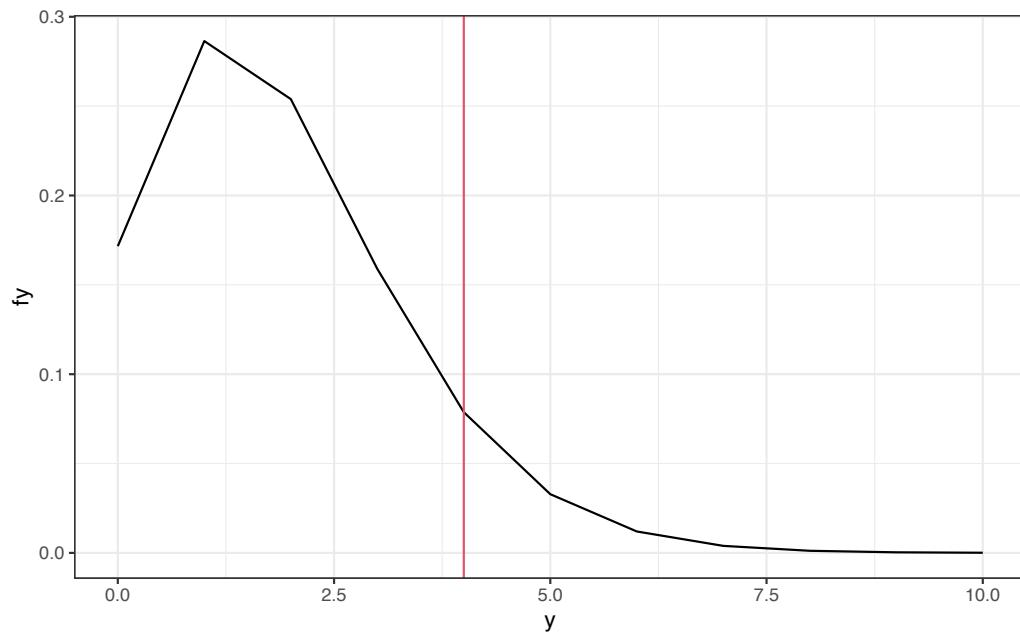
- Hospital B: 4 muertes en 1767 expuestos. Mismos cálculos:

```
alpha <- 16
beta <- 15174
yobs <- 4
ex <- 1767
y <- 0:10
lam <- alpha/beta

fy <- dpois(y, lam * ex) * dgamma(lam, shape = alpha,
  rate = beta)/dgamma(lam, shape = alpha + y, rate = beta +
  ex)

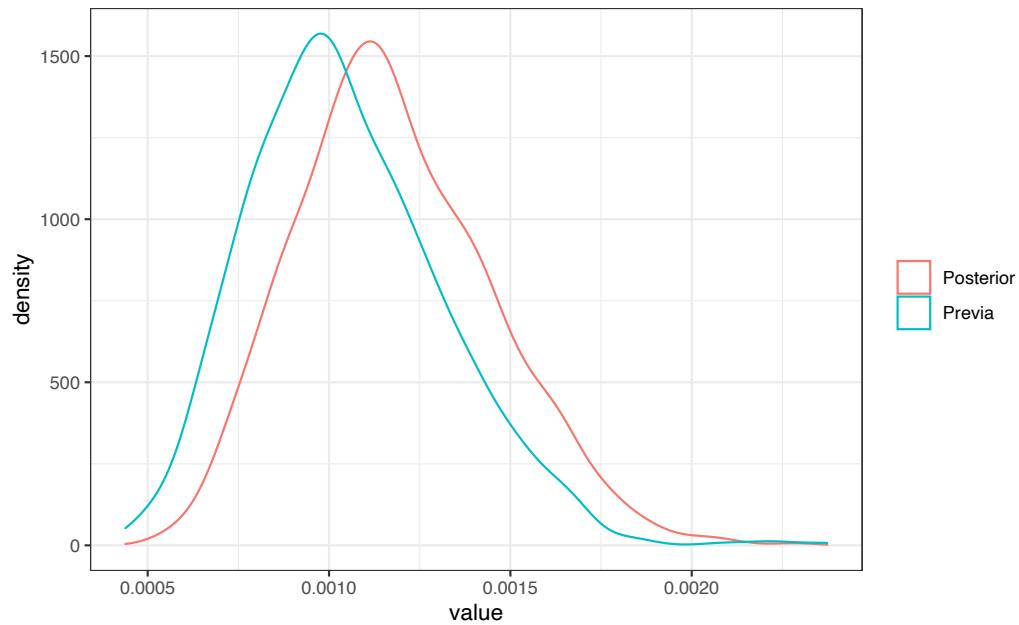
dpred <- tibble(y, fy)
```

```
ggplot(dpred) + geom_line(mapping = aes(x = y, y = fy)) +
  geom_vline(xintercept = yobs, col = 2) + theme_bw()
```



```
set.seed(1)
lambda_prev <- rgamma(1000, shape = alpha, rate = beta)
lambda_post <- rgamma(1000, shape = alpha + yobs, rate = beta +
  ex)
datoslambda <- tibble(Previa = lambda_prev, Posterior = lambda_post) %>%
  pivot_longer(cols = everything())

ggplot(data = datoslambda) + geom_density(mapping = aes(x = value,
  color = name)) + theme_bw() + theme(legend.title = element_blank())
```



9.1.2. Modelo de más de un parámetro

Se usará el ejemplo de la sección 4.2 del (Albert y col. 2009) para ilustrar la inferencia bayesiana conjugada en el caso de más un parámetro. Suponga que se tiene datos del tiempo en completar la maratón de Nueva York para 20 atletas entre 20 y 29 años y asumimos que la muestra proviene de una $N(\mu, \sigma^2)$. Si asumimos la previa no informativa:

$$g(\mu, \sigma^2) \propto 1/\sigma^2$$

entonces la distribución posterior de (μ, σ^2) es:

$$g(\mu, \sigma^2 | y) \propto \frac{1}{(\sigma^2)^{n/2+1}} \exp \left(-\frac{1}{2\sigma^2} (S + n(\mu - \bar{y})^2) \right)$$

donde n es el tamaño de muestra, \bar{y} es la media empírica y $S = \sum_{i=1}^n (y_i - \bar{y})^2$. Recuerden que la distribución posterior conjunta satisface:

- La distribución posterior de μ condicional en σ^2 se distribuye como $N(\bar{y}, \sigma^2/\sqrt{n})$.

- La distribución posterior marginal de σ^2 se distribuye según $S\chi_{n-1}^{-2}$ (S veces una chi-cuadrada inversa con $n - 1$ grados de libertad).

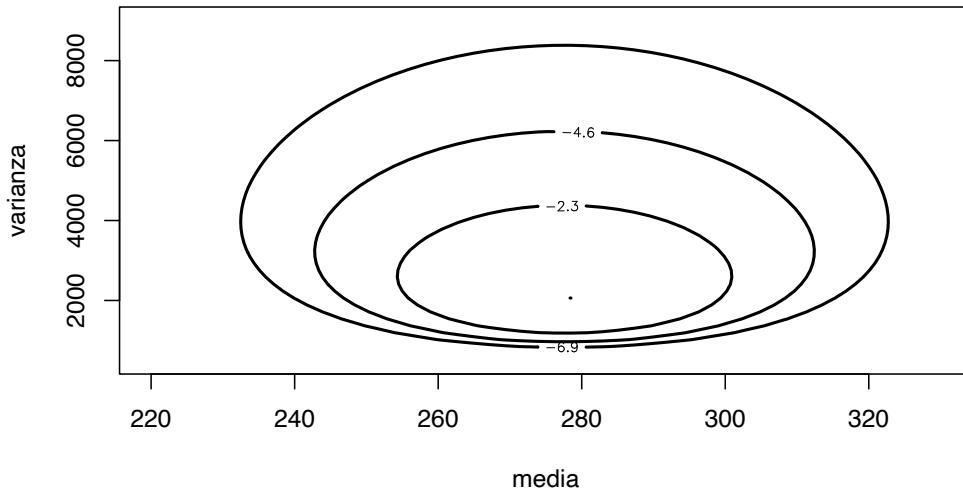
Cargamos los datos de los 20 atletas:

```
library(LearnBayes)
head(marathontimes)
```

```
##   time
## 1 182
## 2 201
## 3 221
## 4 234
## 5 237
## 6 251
```

y graficamos un diagrama de contorno de la distribución posterior de μ, σ^2 :

```
attach(marathontimes)
mycontour(logf = normchi2post, limits = c(220, 330,
500, 9000), data = time, xlab = "media", ylab = "varianza")
```



y les agregamos una muestra aleatoria de tamaño 1000 de la distribución posterior conjunta, generada a través de las distribuciones marginales:

```
S <- sum((time - mean(time))^2)

## Error in time - mean(time): non-numeric argument to binary operator
n <- length(time)
sigma2 <- S/rchisq(1000, n - 1)

## Error in S/rchisq(1000, n - 1): non-numeric argument to binary operator
mu <- rnorm(1000, mean = mean(time), sd = sqrt(sigma2)/sqrt(n))

## Error in rnorm(1000, mean = mean(time), sd = sqrt(sigma2)/sqrt(n)): object
mycontour(normchi2post, c(220, 330, 500, 9000), time,
           xlab = "media", ylab = "varianza")

## Error in y - mu: non-numeric argument to binary operator
points(mu, sigma2)

## Error in points(mu, sigma2): object 'mu' not found
```

Si estamos interesados en hacer inferencia de μ , podemos calcular un intervalo de credibilidad al 95 %, usando la muestra marginal:

```
quantile(mu, c(0.025, 0.975))

##      2.5%    97.5%
## 255.1484 301.2853
```

y también inferencia sobre σ :

```
quantile(sqrt(sigma2), c(0.025, 0.975))

##      2.5%    97.5%
## 37.66930 72.69096
```

o aún sobre otros parámetros, por ejemplo el coeficiente de variación ($CV = \sigma/\mu$):

```
quantile(sqrt(sigma2)/mu, c(0.025, 0.975))
```

```
##      2.5%    97.5%
## 0.1334257 0.2677118
```

9.2. Motivación: Cálculo de Integrales

Recuerden que según el teorema de Bayes, si observamos datos y a partir de una verosimilitud $f(y|\theta)$ y se le asigna al parámetro θ una previa $g(\theta)$, entonces:

$$g(\theta|y) \propto g(\theta)f(y|\theta)$$

Problema: tratar de manejar la distribución posterior de θ desde un punto de vista computacional con el fin de hacer inferencia.

Los procesos de inferencia requieren el cálculo o aproximación de integrales, por ejemplo:

- Valor esperado de una función de θ :

$$E(h(\theta)|y) = \frac{\int h(\theta)g(\theta)f(y|\theta)d\theta}{\int g(\theta)f(y|\theta)d\theta}$$

- Probabilidad posterior de que $h(\theta) \in A$:

$$P(h(\theta) \in A|y) = \frac{\int_{h(\theta) \in A} g(\theta)f(y|\theta)d\theta}{\int g(\theta)f(y|\theta)d\theta}$$

- Densidades marginales. Si $\theta = (\theta_1, \theta_2)$ y se quiere obtener la distribución para θ_1 , entonces se integra con respecto a θ_2 .

$$g(\theta_1|y) \propto \int g(\theta_1, \theta_2|y)d\theta_2$$

9.3. Ejemplo base: modelo beta-binomial.

En este ejemplo se estimará las tasas de muerte por cáncer gástrico en una población de hombres entre 45 y 64 años. Para ello se tiene datos de muertes y_j y exposición n_j para 20 ciudades en Missouri:

```
data("cancermortality")
head(cancermortality)

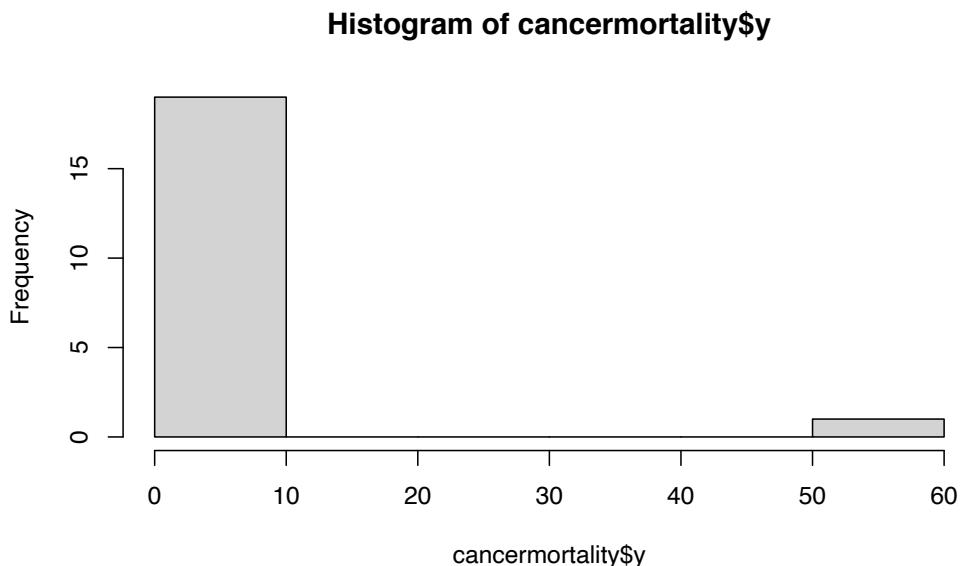
##   y     n
## 1 0 1083
## 2 0  855
## 3 2 3461
## 4 0  657
## 5 1 1208
## 6 1 1025
```

Un primer intento de modelación podría considerar $y_j \sim \text{Binomial}(p, n_j)$. Es decir que existe una probabilidad común p para toda la población.

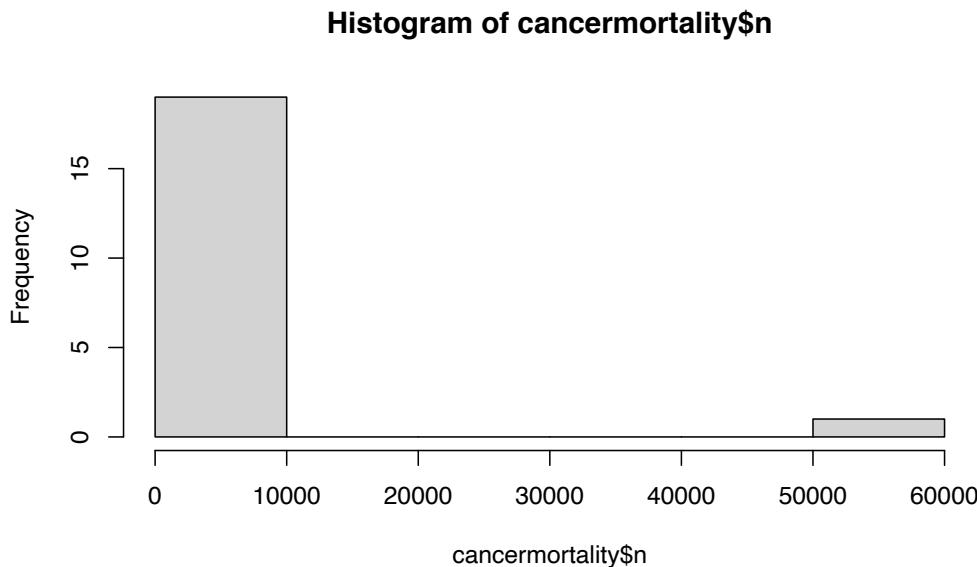
```
(p_emp <- sum(cancermortality$y)/(sum(cancermortality$n)))

## [1] 0.0009933126

hist(cancermortality$y)
```



```
hist(cancermortality$n)
```



En este caso se puede comprobar que el modelo binomial no logra captar la variabilidad de las muertes totalmente.

Otro intento de modelación que no tiene ese problema es un modelo beta-binomial con media η y precisión K :

$$f(y_j|\eta, K) = \binom{n_j}{y_j} \frac{B(K\eta + y_j, K(1 - \eta) + n_j - y_j)}{B(K\eta, K(1 - \eta))}$$

con previa no informativa:

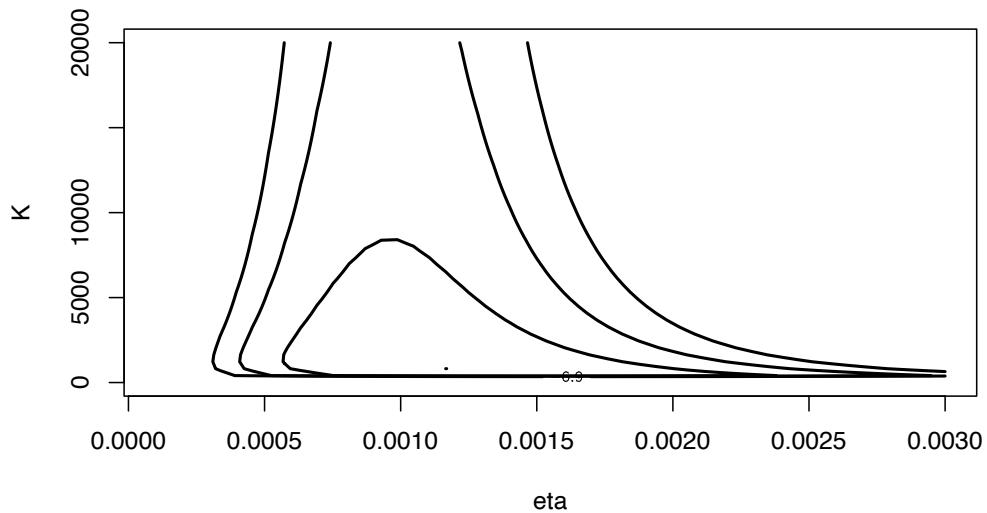
$$g(\eta, K) \propto \frac{1}{\eta(1 - \eta)} \frac{1}{(1 + K)^2}$$

entonces la densidad posterior de los parámetros sería:

$$g(\eta, K | \text{datos}) \propto \frac{1}{\eta(1 - \eta)} \frac{1}{(1 + K)^2} \prod_{j=1}^{20} \frac{B(K\eta + y_j, K(1 - \eta) + n_j - y_j)}{B(K\eta, K(1 - \eta))}$$

donde $0 < \eta < 1$, $K > 0$ y $B(\cdot, \cdot)$ es la función beta. La función *betabinexch0* contiene la implementación de la log-densidad posterior de η, K :

```
mycontour(betabinexch0, c(1e-04, 0.003, 1, 20000),
cancermortality, xlab = "eta", ylab = "K")
```



y note la gran asimetría en el comportamiento de la densidad conjunta, especialmente en la dirección de la variable K . Por el dominio de las variables K y η , entonces se transforman según:

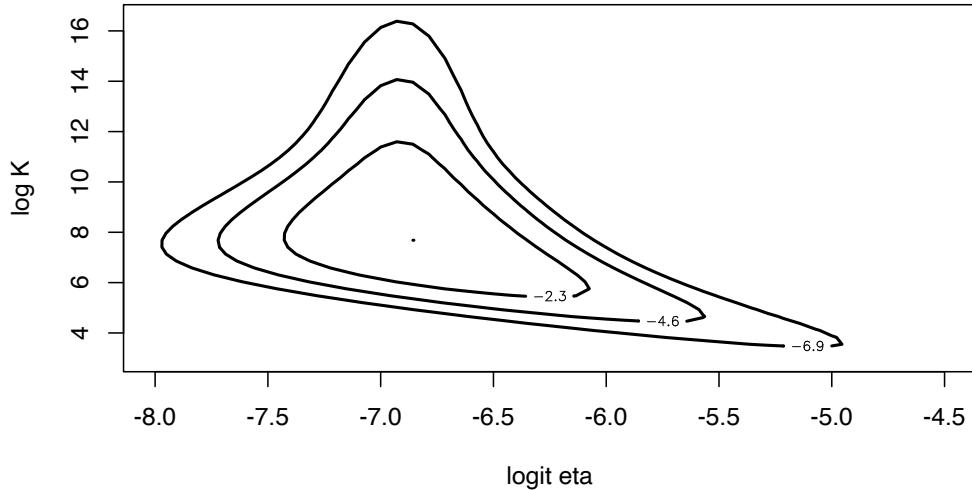
$$\theta_1 = \text{logit}(\eta) = \log\left(\frac{\eta}{1-\eta}\right), \quad \theta_2 = \log(K)$$

y usando el teorema de cambio de variable en densidades:

$$g_1(\theta_1, \theta_2 | \text{datos}) = g\left(\frac{e^{\theta_1}}{1+e^{\theta_1}}, e^{\theta_2}\right) \frac{e^{\theta_1+\theta_2}}{(1+e^{\theta_1})^2}$$

g_1 está implementada en la función *betabinexch* y el gráfico de contorno es más manejable ahora desde el punto de vista computacional:

```
mycontour(betabinexch, c(-8, -4.5, 3, 16.5), cancermortality,
xlab = "logit eta", ylab = "log K")
```



Definitivamente esta es una distribución posterior a la que no se le puede aplicar las técnicas usuales para hacer inferencia (caso no conjugado). Se va a considerar dos formas de realizar inferencia:

- Aproximación de Laplace.
- Simulación Monte Carlo.

9.4. Aproximación de Laplace

Una forma de resumir el comportamiento de la posterior, es a través del comportamiento de la moda de la densidad.

Considere el logaritmo de la densidad posterior conjunta de θ e y :

$$h(\theta, y) = \log(g(\theta)f(y|\theta))$$

Suponga que $\hat{\theta}$ es la moda de θ . Un desarrollo de Taylor alrededor de $\hat{\theta}$ para $h(\theta)$ da la siguiente aproximación:

$$h(\theta) \approx h(\hat{\theta}) + (\theta - \hat{\theta})^\top h'(\theta) + \frac{1}{2}(\theta - \hat{\theta})^\top h''(\hat{\theta})(\theta - \hat{\theta})$$

Ahora dado que $h'(\hat{\theta}) = 0$ dado que es la moda, entonces la expresión se simplifica a:

$$h(\theta) \approx h(\hat{\theta}) + \frac{1}{2}(\theta - \hat{\theta})^\top h''(\hat{\theta})(\theta - \hat{\theta})$$

Para estimar el comportamiento de θ note que

$$\begin{aligned} g(\theta)f(y \mid \theta) &= \exp(h(\theta)) \\ &= \exp\left(h(\hat{\theta}) + \frac{1}{2}(\theta - \hat{\theta})^\top h''(\hat{\theta})(\theta - \hat{\theta})\right) \\ &= \exp(h(\hat{\theta})) \exp\left(\frac{1}{2}(\theta - \hat{\theta})^\top h''(\hat{\theta})(\theta - \hat{\theta})\right) \\ &= \exp(h(\hat{\theta})) \exp\left(-\frac{1}{2}(\theta - \hat{\theta})^\top (-h''(\hat{\theta}))^{-1}(\theta - \hat{\theta})\right) \end{aligned}$$

Por lo tanto podemos aproximar el comportamiento en distribución de θ como:

$$\theta \sim N(\hat{\theta}, V)$$

donde $V = (-h''(\hat{\theta}))^{-1}$.

Se podría encontrar una solución analítica del problema integrando $\exp(h(\theta, y))$ con respecto a θ y obteniendo la posterior predictiva de la siguiente forma,

$$f(y) \approx (2\pi)^{d/2} g(\hat{\theta}) f(y \mid \hat{\theta}) \left| -h''(\hat{\theta}) \right|^{1/2}.$$

Con el fin de encontrar la moda $\hat{\theta}$ se puede usar algún algoritmo para encontrar máximos en funciones de varias variables, por ejemplo el método de Newton

o el de Nelder-Mead (default en *optim*). El caso de método de Newton, este usa información de las derivadas para encontrar las direcciones de más bajo decrecimiento.

$$\theta^t = \theta^{t-1} - [h''(\theta^{t-1})]^{-1} h'(\theta^{t-1})$$

El método de Nelder-Mead usa el método simplex, junto con reflecciones, simetrías, contracciones, expansiones para encontrar el punto mínimo en una superficie. En el caso no convexo, podría caer en mínimos locales.

La función *laplace* tiene el método de optimización implementado tomando como argumentos la log-densidad posterior, un valor inicial de los parámetros y el conjunto de datos.

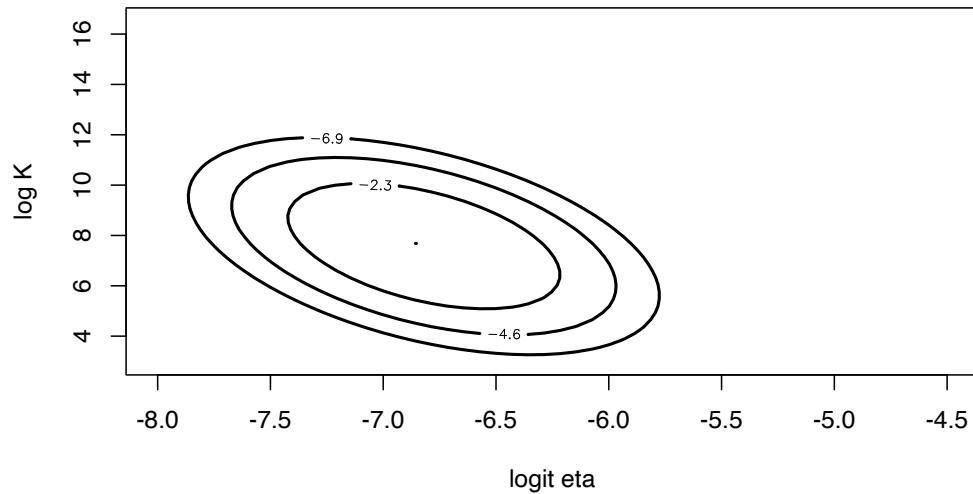
Por ejemplo, en el caso anterior podemos tomar $\text{logit}(\eta)$, $l \log K = (-7, 6)$ como puntos iniciales para el algoritmo de Nelder-Mead. Estos se pueden inferir a través del gráfico de un gráfico de contorno. Por lo tanto podemos aproximar la densidad posterior conjunta de $(\text{logit}(\eta), \log K)$ se puede aproximar como una normal multivariada con media *fit\$mode* y varianza *fit.var*.

```
fit <- laplace(betabinexch, c(-7, 6), cancermortality)
fit
```

```
## $mode
## [1] -6.819793  7.576111
##
## $var
##           [,1]      [,2]
## [1,]  0.07896568 -0.1485087
## [2,] -0.14850874  1.3483208
##
## $int
## [1] -570.7743
##
## $converge
## [1] TRUE
```

Un gráfico de contorno de la aproximación es:

```
npar <- list(m = fit$mode, v = fit$var)
mycontour(lbinorm, c(-8, -4.5, 3, 16.5), npar, xlab = "logit eta",
           ylab = "log K")
```



También podemos hacer inferencia de los parámetros:

```
se <- sqrt(diag(fit$var))
lb <- fit$mode - qnorm(0.975) * se
ub <- fit$mode + qnorm(0.975) * se

etainv <- c(lb[1], ub[1])
Kinv <- c(lb[2], ub[2])

exp(etainv)/(1 + exp(etainv))

## [1] 0.0006291199 0.0018904899
exp(Kinv)

## [1] 200.3879 18995.6680
```

son intervalos de predicción al 95 % para η y K respectivamente.

9.5. Simulación

9.5.1. Simulación Monte Carlo

Suponga que $g(\theta|y)$ es la densidad posterior de θ y queremos estimar una característica de θ , a través de la función $h(\theta)$. La media posterior de $h(\theta)$ es:

$$E(h(\theta)|y) = \int h(\theta)g(\theta|y)d\theta$$

y suponga que podemos simular una muestra independiente $\theta^1, \dots, \theta^m$ de $g(\theta|y)$. El estimador Monte Carlo del valor esperado es:

$$\bar{h} = \frac{1}{m} \sum_{j=1}^m h(\theta^j)$$

con su error estándar:

$$se_{\bar{h}} = \sqrt{\frac{1}{m(m-1)} \sum_{j=1}^m (h(\theta^j) - \bar{h})^2}$$

En el caso en que no es posible obtener muestras de la densidad posterior, entonces se pueden definir algoritmos que aproximan la generación de muestras.

Por ejemplo, en el caso de los hospitales, supongamos que queremos saber que pasaría si la tasa de mortalidad se vuelve 2 veces más rápida. Entonces caso tenemos que

```
head(lambda_post)

## [1] 0.0014598268 0.0011440068 0.0012703128 0.0012821699 0.0009823593
## [6] 0.0009834807

est <- mean(2 * lambda_post)
se <- sd(2 * lambda_post)/sqrt(1000)
c(est, se)

## [1] 2.361136e-03 1.715134e-05
```

9.5.2. Muestreo por rechazo

Suponga que queremos obtener una muestra de $g(\theta|y)$ donde la constante de normalización no es conocida. Suponga que conocemos una densidad $p(\theta)$ que satisface:

- Fácil de obtener muestras.
- p aproxima g en términos de localización y escala.
- Para todo θ : $g(\theta|y) \leq cp(\theta)$, para una constante c .

Algoritmo:

1. Simule una realización independiente de $\theta \sim p$ y $U \sim Unif(0, 1)$.
2. Si $U \leq g(\theta|y)/(cp(\theta))$ entonces acepte θ , caso contrario rechace la muestra propuesta.
3. Continue 1 y 2 hasta que se haya generado un número deseado de muestras.

Para este algoritmo se necesita definir y estimar: - La distribución propuesta p . - La constante c .

Note que en el paso 2, la probabilidad de aceptar candidatos es dado por $g\frac{\theta|y}{cp(\theta)}$. Se puede revisar este valor y ver la efectividad de la distribución propuesta. Si es bien escogida, estos valores deberían ser altos.

Apliquemos este método en el ejemplo de la mortalidad por cancer de los adultos en Missouri.

Primero, debemos encontrar una distribución que al multiplicarla por c , cubra efectivamente toda la distribución $g(\theta | y)$. Una opción sería la Gaussiana bivariada. El problema es que las colas de la normal son ligeras, por lo que $g\frac{\theta|y}{cp(\theta)}$ podría ser no acotado.

Una opción mejor para $p(\theta)$ es una distribución t multivariada de modo que la media y escala sean compatibles con las posteriores.

Suponga que usamos el parámetro de locación igual a la media aproximada del método de Laplace, matriz de escala igual a 2 veces la matriz de varianza aproximada según Laplace y 4 grados de libertad. De esta forma nos aseguramos que $g(\theta|y)/p(\theta)$ está acotado superiormente.

Con el fin de encontrar c , dado que estamos en la log escala debemos encontrar una constante

$$\log g(\theta | y) - \log p(\theta) \leq d$$

Esto se logra maximizando $\log g(\theta | y) - \log p(\theta)$.

Podemos usar la función `laplace` para lograr esto.

```
betabinT <- function(theta, datapar) {
  data <- datapar$data
  tpar <- datapar$par

  d <- betabinexch(theta, data) - dmt(theta, mean = c(tpar$m),
    S = tpar$var, df = tpar$df, log = TRUE)
  return(d)
}
```

definimos parámetros:

```
tpar <- list(m = fit$mode, var = 2 * fit$var, df = 4)
datapar <- list(data = cancermortality, par = tpar)
```

y resolvemos:

```
start <- c(-6.9, 12.4)
fit1 <- laplace(betabinT, start, datapar)
fit1$mode
```

```
## [1] -6.888963 12.421993
```

y el valor máximo de las diferencias de logaritmos es:

```
dmax <- betabinT(fit1$mode, datapar)
dmax
```

```
## [1] -569.2829
```

el algoritmo sería:

```
set.seed(1)
n <- 10000
theta <- rmt(n, mean = c(tpar$m), S = tpar$var, df = tpar$df)
lf <- map_dbl(1:10000, ~betabinexch(theta[., ], cancermortality))
lg <- dmt(theta, mean = c(tpar$m), S = tpar$var, df = tpar$df,
```

```

log = TRUE)
prob <- exp(lf - lg - dmax)
thetaRS <- theta[runif(n) < prob, ]

```

la tasa de aceptación es:

```
dim(thetaRS)[1]/10000
```

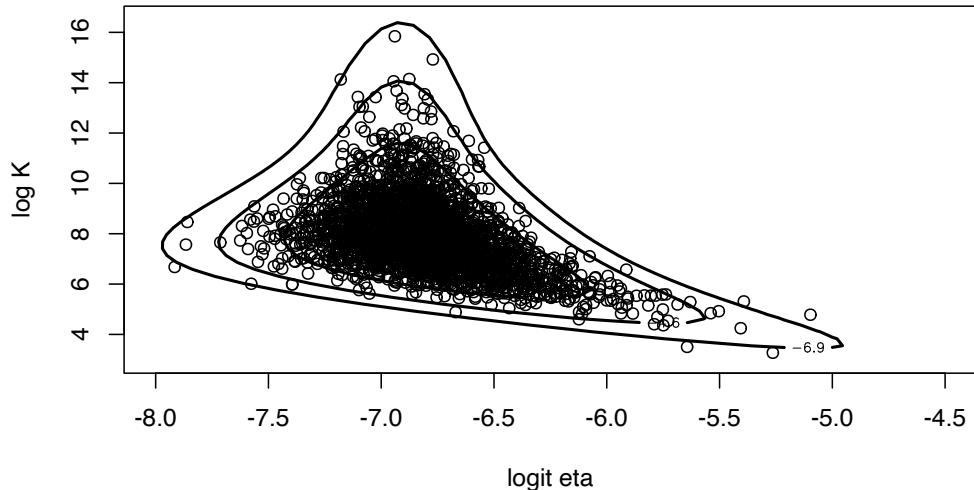
```
## [1] 0.2447
```

y dibujamos el gráfico de contorno con la muestra obtenida:

```

mycontour(betabinexch, c(-8, -4.5, 3, 16.5), cancermortality,
           xlab = "logit eta", ylab = "log K")
points(thetaRS[, 1], thetaRS[, 2])

```



El mismo proceso se podría hacer con el siguiente código

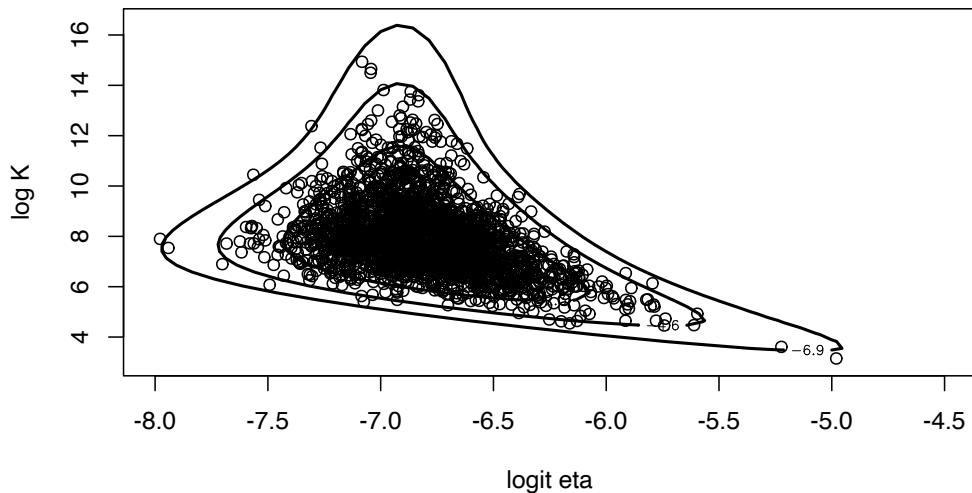
```

thetaRS2 <- rejectsampling(betabinexch, tpar, -569.2813,
                            10000, cancermortality)

dim(thetaRS2)

```

```
## [1] 2408      2
mycontour(betabinexch, c(-8, -4.5, 3, 16.5), cancermortality,
           xlab = "logit eta", ylab = "log K")
points(thetaRS2[, 1], thetaRS2[, 2])
```



9.6. Muestreo por importancia

Suponga que queremos calcular el siguiente valor esperado posterior:

$$E(h(\theta)|y) = \frac{\int h(\theta)g(\theta)f(y|\theta)d\theta}{\int g(\theta)f(y|\theta)d\theta}$$

en el caso en donde no se puede obtener una muestra directa de la distribución posterior y usar Monte Carlo por ejemplo. Usemos la densidad propuesta $p(\theta)$ que aproxima la posterior:

$$\begin{aligned} E(h(\theta)|y) &= \frac{\int h(\theta) \frac{g(\theta)f(y|\theta)}{p(\theta)} p(\theta) d\theta}{\int \frac{g(\theta)f(y|\theta)}{p(\theta)} p(\theta) d\theta} \\ &= \frac{\int h(\theta) w(\theta) p(\theta) d\theta}{\int w(\theta) p(\theta) d\theta} \end{aligned}$$

donde $w(\theta) = \frac{g(\theta)f(y|\theta)}{p(\theta)}$. Si $\theta^1, \dots, \theta^m \sim p(\theta)$ entonces el estimador de muestreo por importancia de la media posterior es:

$$\bar{h}_{IS} = \frac{\sum_{j=1}^m h(\theta^j)w(\theta^j)}{\sum_{j=1}^m w(\theta^j)}$$

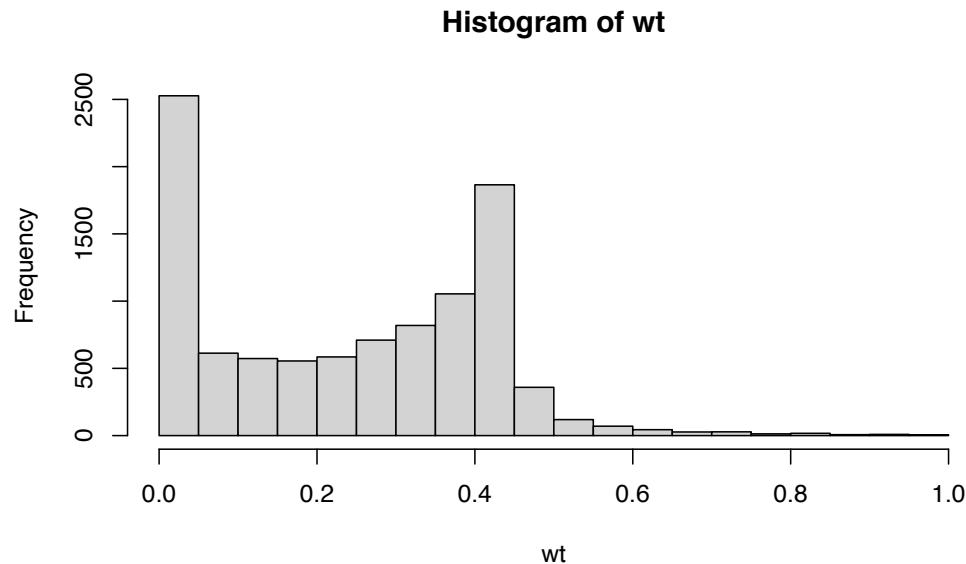
con error estándar:

$$se_{\bar{h}_{IS}} = \frac{\sqrt{\sum_{j=1}^m ((h(\theta^j) - \bar{h}_{IS})w(\theta^j))^2}}{\sum_{j=1}^m w(\theta^j)}$$

Nota: al igual que en el método anterior, la escogencia de $p(\theta)$ se basa en su facilidad de muestreo y en la acotación por arriba de los pesos $w(\theta)$.

Ahora implementamos el algoritmo en el ejemplo (buena parte es una repetición del anterior), usando como propuesta la misma distribución t multivariada. Además graficamos un histograma de los pesos para comprobar que están acotados (en este caso por 1).

```
set.seed(1)
n <- 10000
theta <- rmt(n, mean = c(tpar$m), S = tpar$var, df = tpar$df)
lf <- map_dbl(1:10000, ~betabinexch(theta[., ], cancermortality))
lp <- dmt(theta, mean = c(tpar$m), S = tpar$var, df = tpar$df,
           log = TRUE)
md <- max(lf - lp)
wt <- exp(lf - lp - md)
hist(wt)
```



y calculamos el valor esperado de $\log K$ usando los pesos obtenidos del paso anterior:

```
est <- sum(wt * theta[, 2])/sum(wt)
SE_est <- sqrt(sum((theta[, 2] - est)^2 * wt^2))/sum(wt)
show(c(est, SE_est))
```

```
## [1] 7.92445868 0.01905786
```

La función para LearnBayes sería,

```
set.seed(1)
tpar <- list(m = fit$mode, var = 2 * fit$var, df = 4)
myfunc <- function(theta) {
  return(theta[2])
}
s <- impsampling(betabinexch, tpar, myfunc, 10000,
  cancermortality)
cbind(s$est, s$se)

##          [,1]      [,2]
## [1,] 7.924459 0.01905786
```

9.7. Remuestreo por importancia

Al igual que en el caso anterior simulamos $\theta^1, \dots, \theta^m \sim p(\theta)$ y calculamos los pesos $\{w(\theta^j) = g(\theta^j|y)/p(\theta^j)\}$. Los pesos se convierten a probabilidades según:

$$p^j = \frac{w(\theta^j)}{\sum_{k=1}^m w(\theta^k)}$$

Tomamos una nueva muestra $\theta^{*1}, \dots, \theta^{*m} \sim \{p^k\}$ es decir se obtiene una nueva muestra con reemplazo a partir de la muestra original $\theta^1, \dots, \theta^m$ con pesos $\{p^k\}$ (muestra bootstrap ponderada). A este método se le llama remuestreo por importancia.

Siguiendo con el ejemplo:

```
probs <- wt/sum(wt)
indices <- sample(1:n, size = n, prob = probs, replace = T)
theta_SIR <- theta[indices, ]
```

y los intervalos de predicción al 95 % para $\text{logit}(\eta)$ y $\log K$ son:

```
quantile(theta_SIR[, 1], probs = c(0.025, 0.975))
```

```
##      2.5%    97.5%
## -7.353821 -6.179805
```

```
quantile(theta_SIR[, 2], probs = c(0.025, 0.975))
```

```
##      2.5%    97.5%
##  5.618474 11.198193
```

Hay otra función que hace exactamente este procedimiento,

```
theta_SIR2 = sir(betabinexch, tpar, 10000, cancermortality)
```

```
quantile(theta_SIR2[, 1], probs = c(0.025, 0.975))
```

```
##      2.5%    97.5%
## -7.355285 -6.147236
```

```
quantile(theta_SIR2[, 2], probs = c(0.025, 0.975))
```

```
##      2.5%    97.5%
```

```
## 5.638189 11.165893
```

9.8. Métodos Monte Carlo

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.

9.8.1. Ejemplo del viajero con una moneda

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

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)

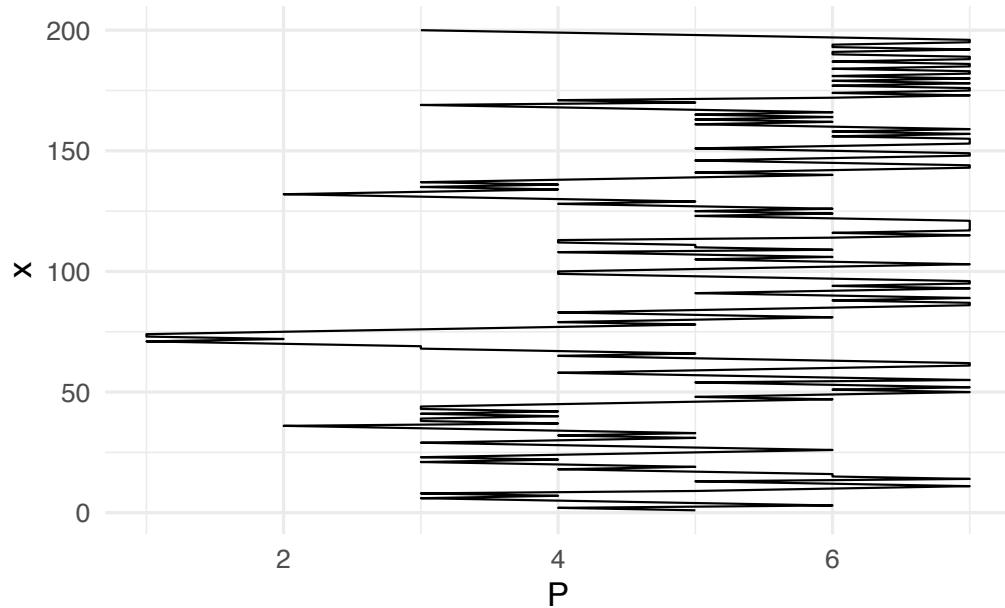
  if (hay_movimiento) {
    pos_actual <- pos_nueva
  }

  trayectoria[k] <- pos_nueva
}

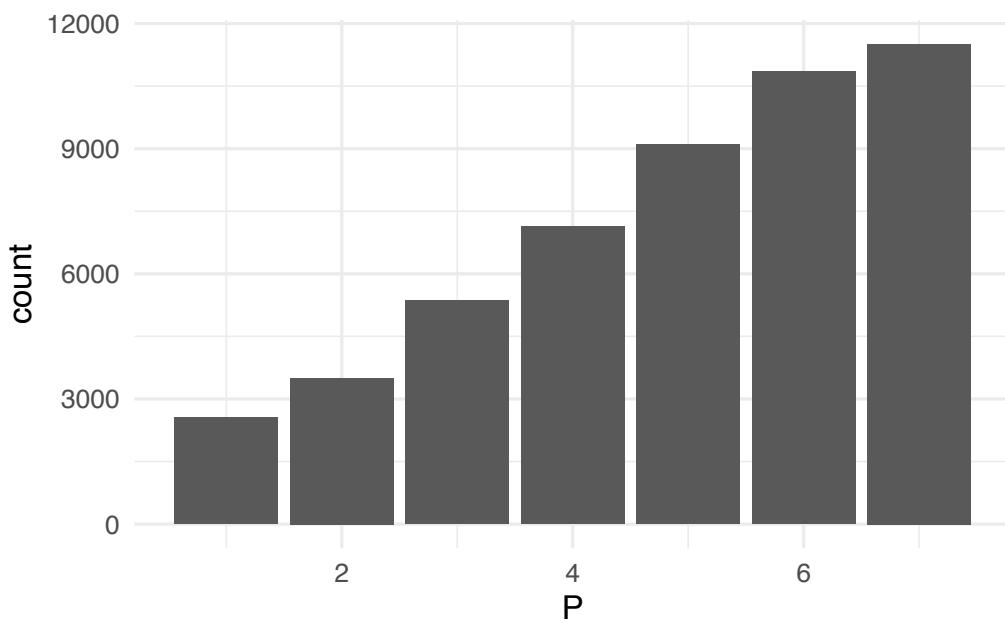
df <- data.frame(x = 1:n_pasos, P = trayectoria)

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] 4.90532
sd(trayectoria)
## [1] 1.774739
```

9.8.2. 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 para alguna potencia de T todos los sus elementos serán positivos estrictos.

9.8.3. 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 Metropolis 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,

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

Otras formas es escoger es usando

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

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 simular solo el *salto* Z

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

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

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

$$\mathbb{P}(\theta^* | \theta^{(t-1)}) = L(\theta^* | \theta^{(t-1)}) \cdot \min \left\{ 1, \frac{f(\theta^*) L(\theta^{(t-1)} | \theta^*)}{f(\theta^{(t-1)}) L(\theta^* | \theta^{(t-1)})} \right\} \quad (9.1)$$

Si se comienza en $f(\theta^{(t-1)})$ entonces

$$= f(\theta^{(t-1)}) \mathbb{P}(\theta^* | \theta^{(t-1)}) \quad (9.2)$$

$$= f(\theta^{(t-1)}) L(\theta^* | \theta^{(t-1)}) \min \left\{ 1, \frac{f(\theta^*) L(\theta^{(t-1)} | \theta^*)}{f(\theta^{(t-1)}) L(\theta^* | \theta^{(t-1)})} \right\} \quad (9.3)$$

$$= \min \left\{ f(\theta^{(t-1)}) L(\theta^* | \theta^{(t-1)}), f(\theta^*) L(\theta^{(t-1)} | \theta^*) \right\} \quad (9.4)$$

$$= f(\theta^*) L(\theta^{(t-1)} | \theta^*) \min \left\{ \frac{f(\theta^{(t-1)}) L(\theta^* | \theta^{(t-1)})}{f(\theta^*) L(\theta^{(t-1)} | \theta^*)}, 1 \right\} \quad (9.5)$$

$$= f(\theta^*) \mathbb{P}(\theta^{(t-1)} | \theta^*) \quad (9.6)$$

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

$$f(\theta_j) = \underbrace{\sum_{i=1}^M f(\theta_i) \mathbb{P}(\theta_j | \theta_i)}_{\text{Probabilidad total}} = \sum_{i=1}^M f(\theta_j) \mathbb{P}(\theta_i | \theta_j)$$

$$f(\boldsymbol{\theta})^\top T = f(\boldsymbol{\theta}) \quad (9.7)$$

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

9.8.5. 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}

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

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 <- theta^z * (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 49.8 y tasa de rechazo del 50.2

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] 0.683422
sd(trayectoria[500:n_pasos])

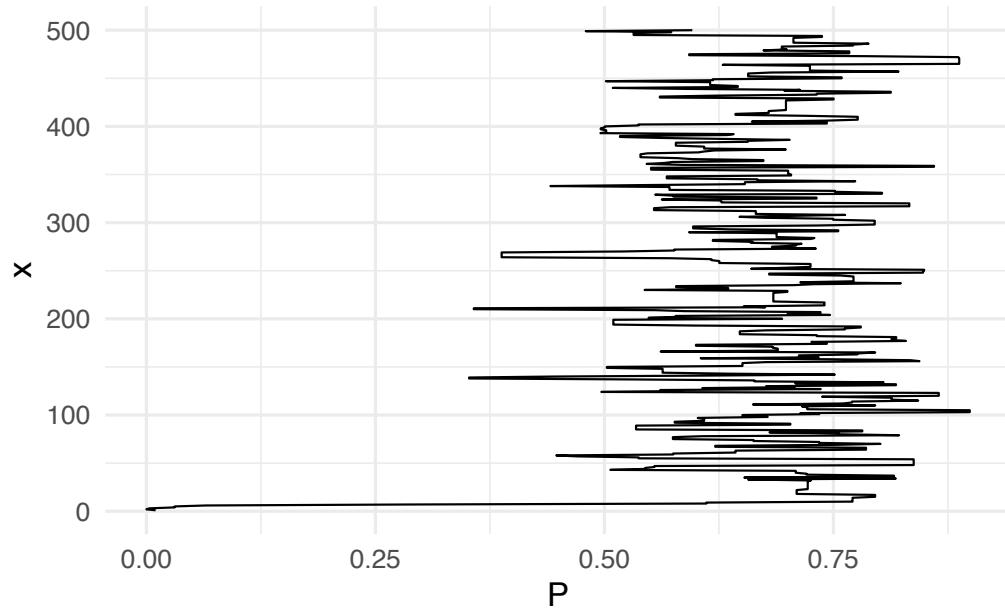
```

```

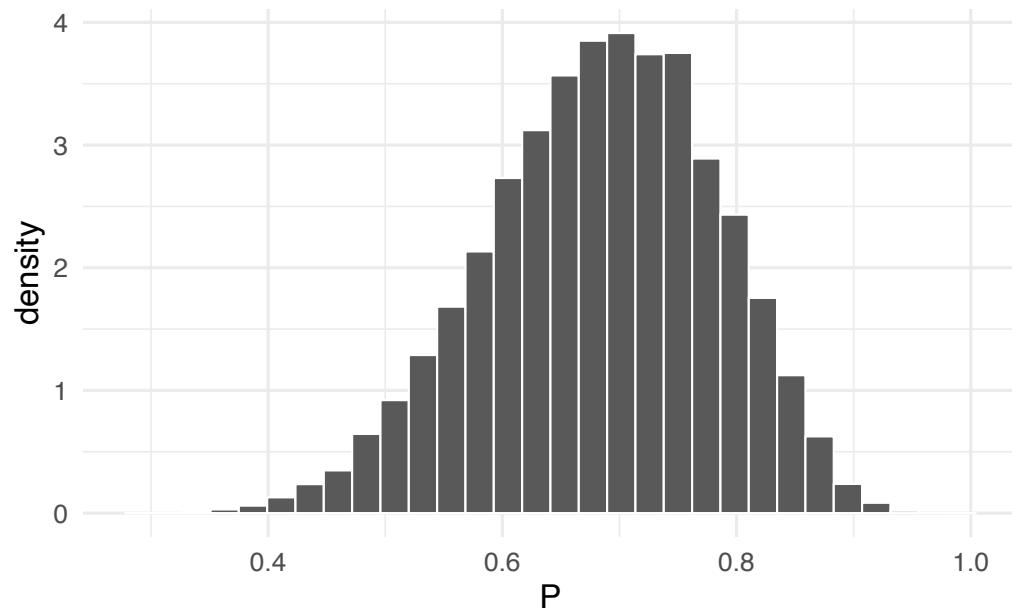
## [1] 0.09785508
df <- data.frame(x = 1:n_pasos, P = trayectoria)

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)
```



9.9. El problema del viajero con 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-Hastings. [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]
}
}

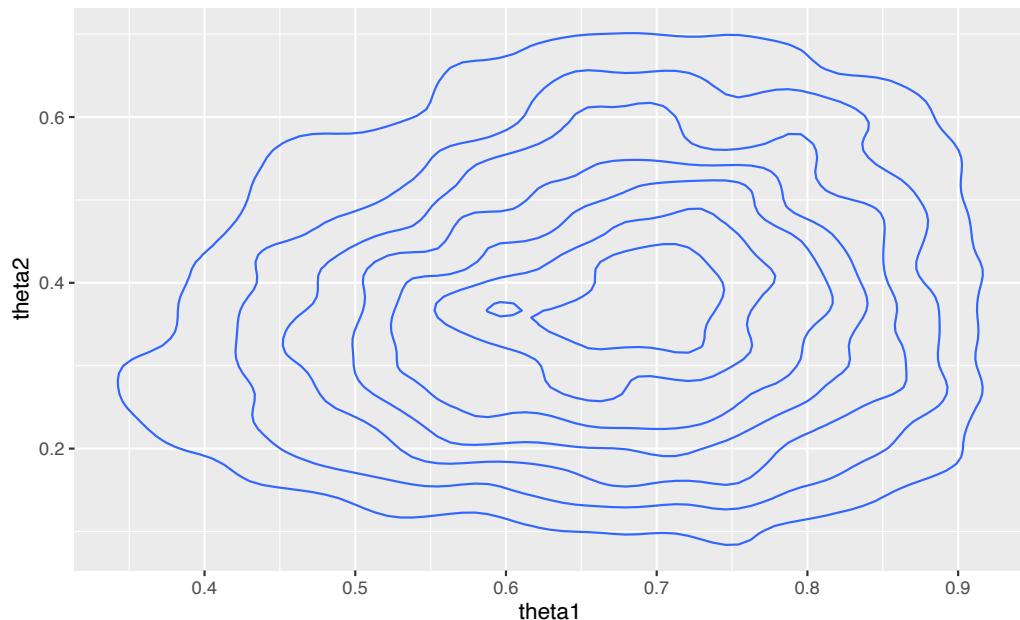
tibble(
  step = 1:num_steps,
  theta1 = theta1,
  theta2 = theta2,
  proposed_theta1 = proposed_theta1,
  proposed_theta2 = proposed_theta2,
  move = move,
  size1 = size1,
  size2 = size2
)

```

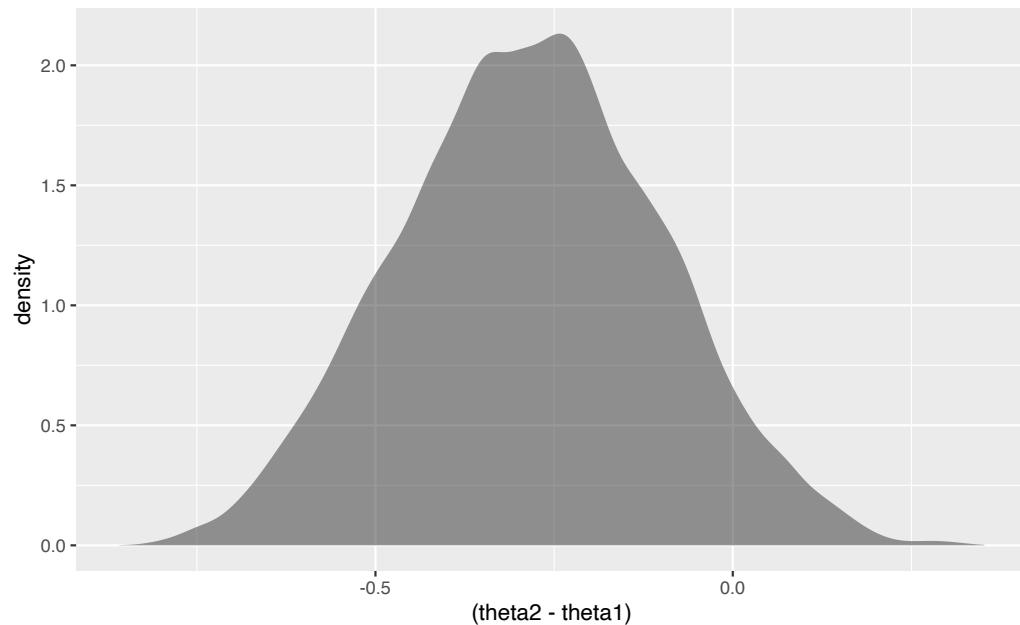
```
}
```

```
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))
```

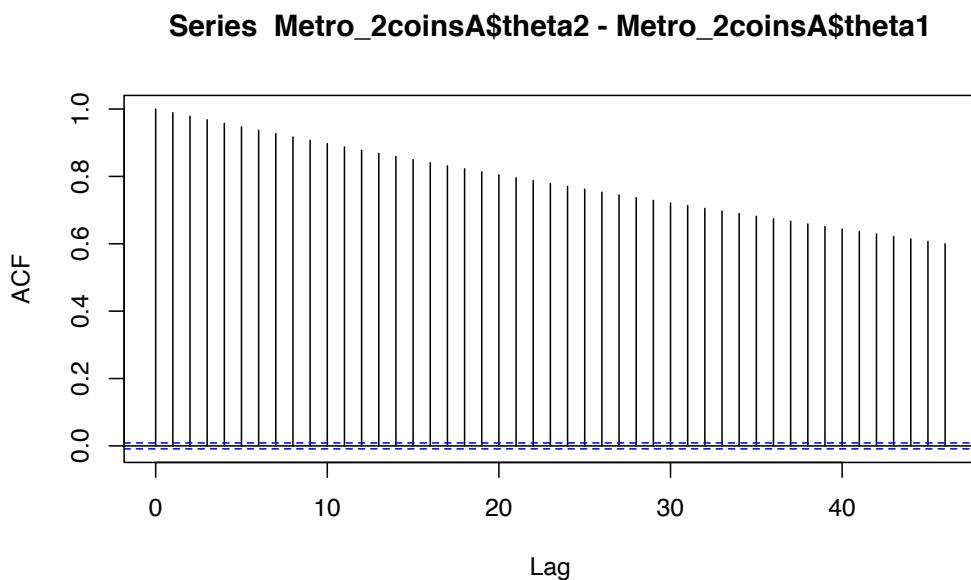
```
Metro_2coinsA %>%
  gf_density2d(theta2 ~ theta1)
```



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



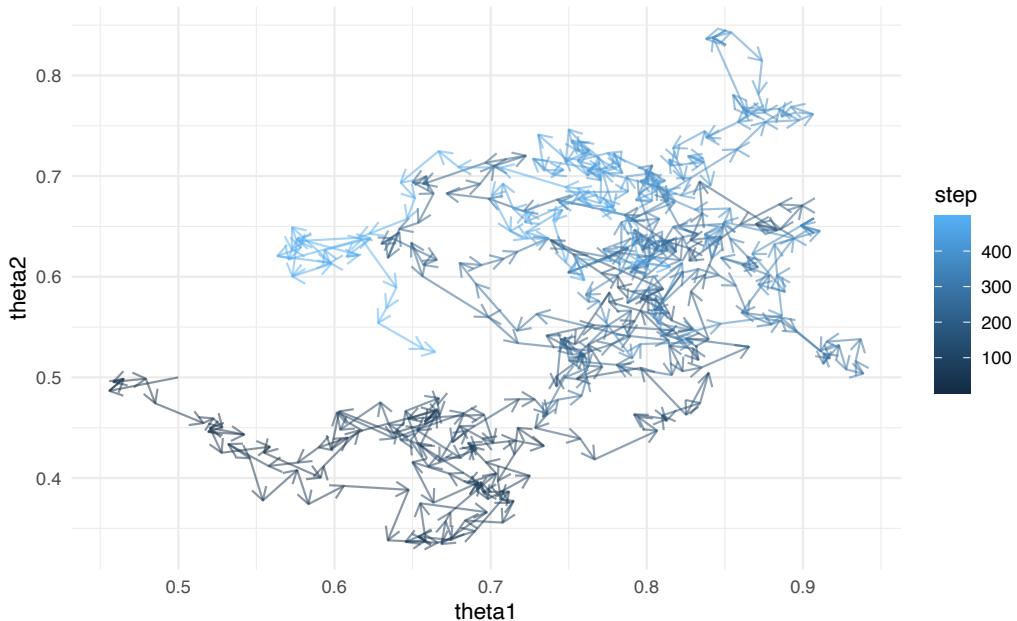
```
acf(Metro_2coinsA$theta2 - Metro_2coinsA$theta1)
```



```

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()

```



```

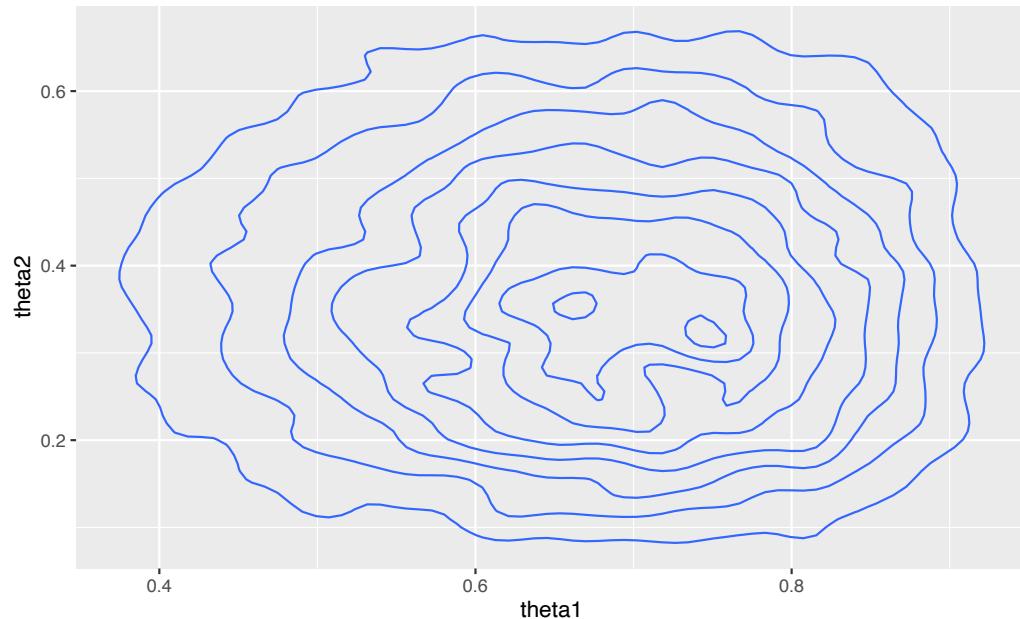
library(gganimate)
Metro_2coinsAplot <- Metro_2coinsA %>%
  filter(step < 500) %>%
  gf_path(theta2 ~ theta1, color = ~step, alpha = 0.5,
    arrow = arrow(type = "open", angle = 30)) +
  theme_minimal() + transition_reveal(step)

animate(Metro_2coinsAplot, fps = 1)

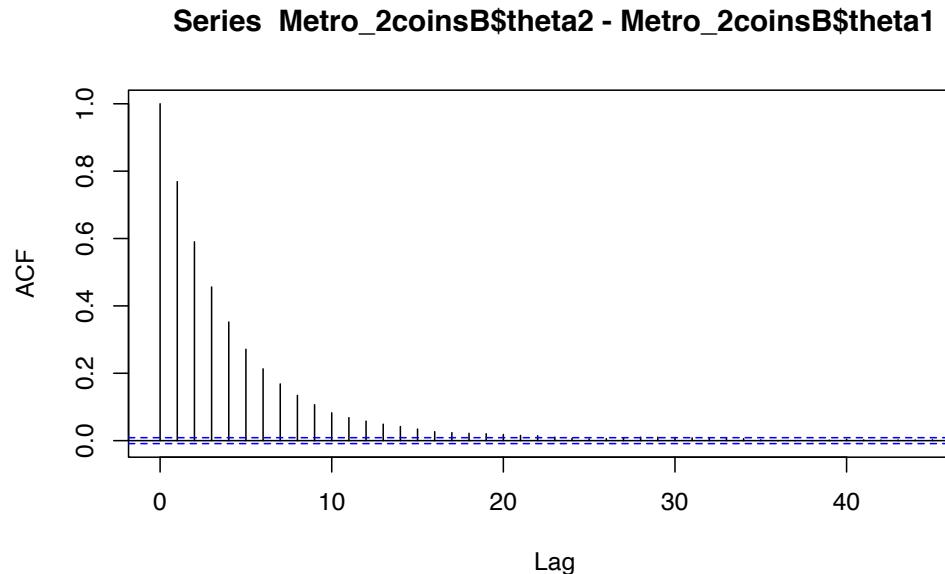
Metro_2coinsB <- metro_2coins(z1 = 6, n1 = 8, z2 = 2,
  n2 = 7, size = c(0.2, 0.2), args1 = list(shape1 = 2,
  shape2 = 2), args2 = list(shape1 = 2, shape2 = 2))

```

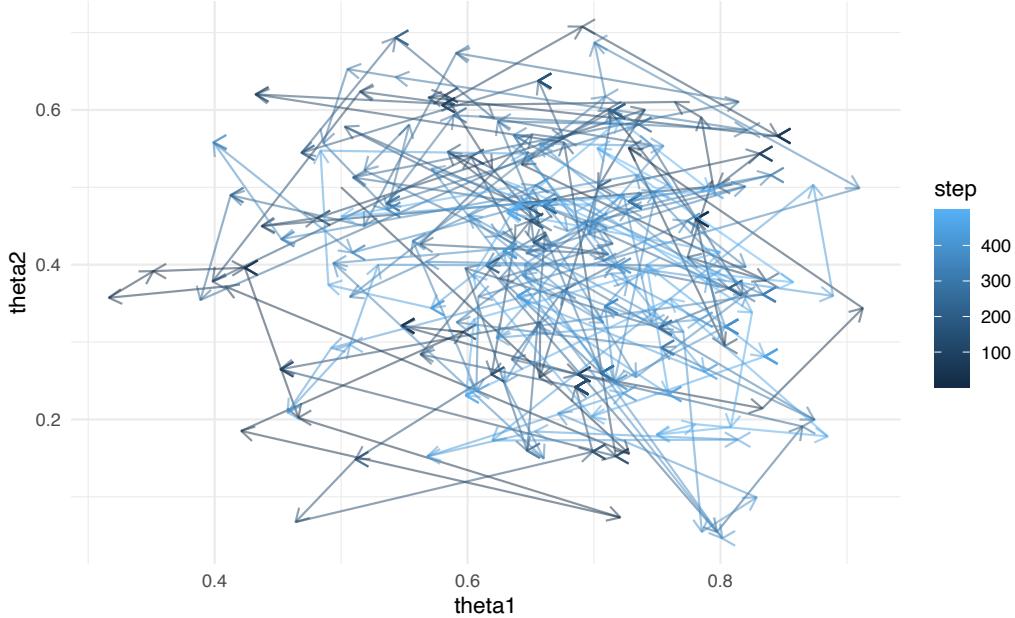
```
Metro_2coinsB %>%
  gf_density2d(theta2 ~ theta1)
```



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



```
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)

```

9.9.1. 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]
}

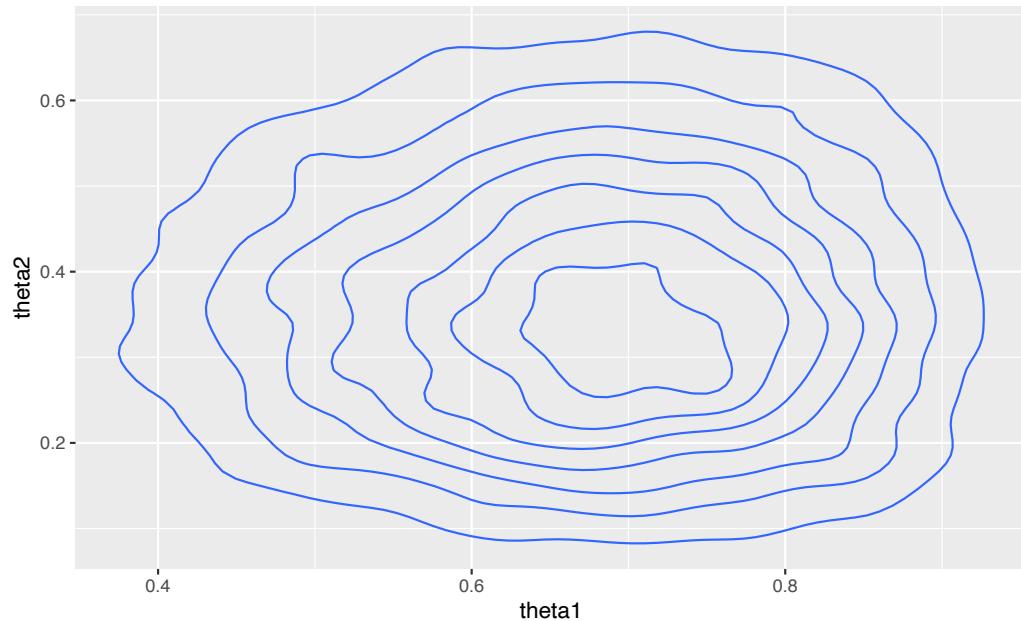
```

```
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)
  }
}

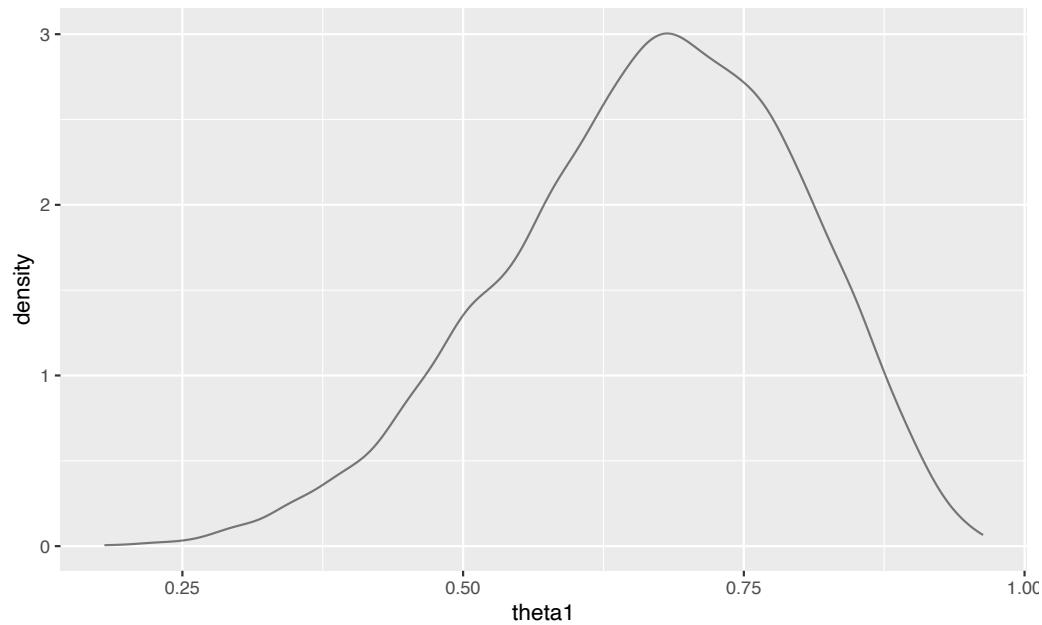
tibble(
  step = 1:num_steps,
  theta1 = theta1,
  theta2 = theta2,
)
}

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

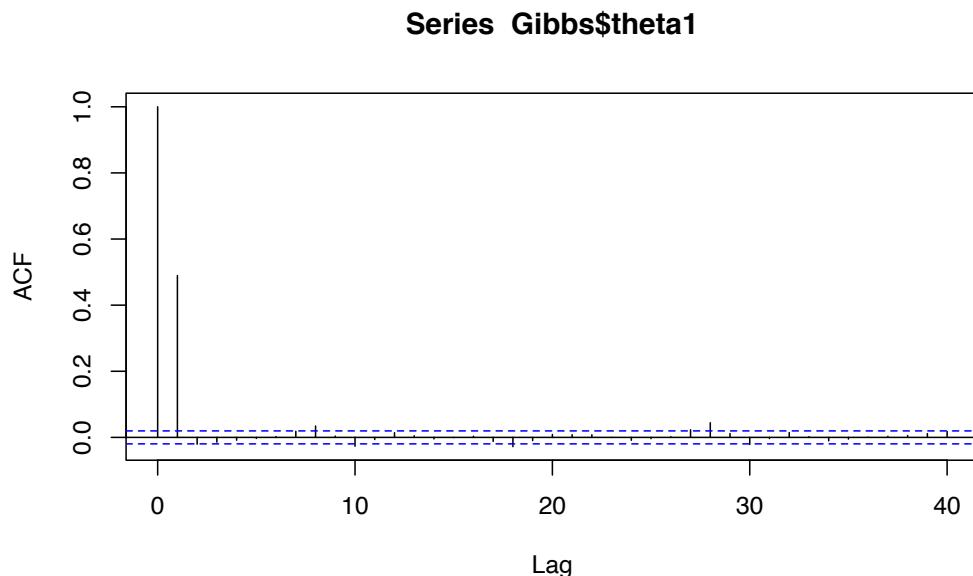
Gibbs %>%
  gf_density2d(theta2 ~ theta1)
```



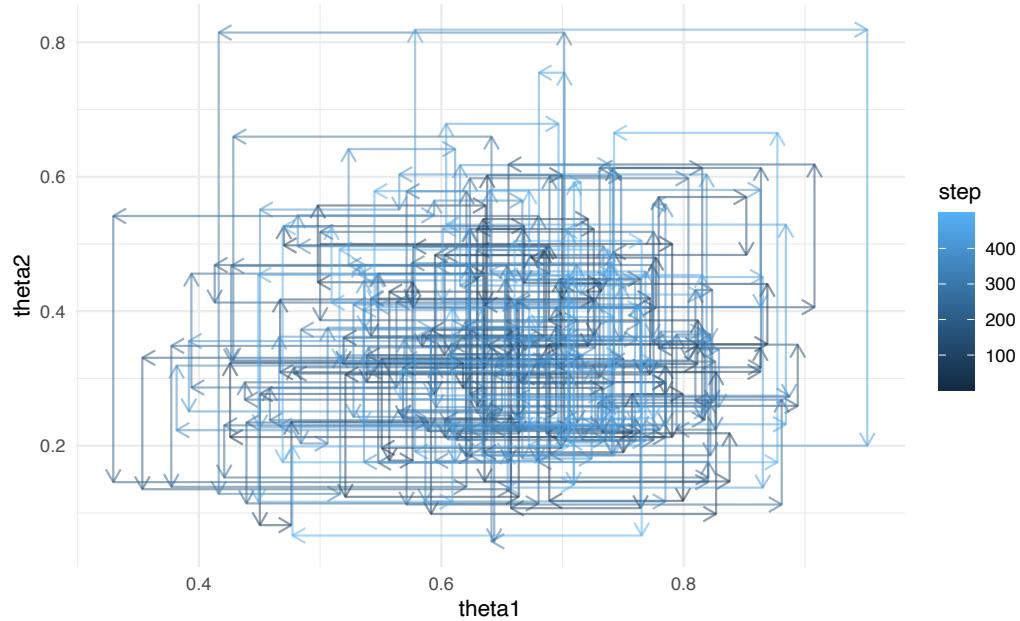
```
Gibbs %>%  
  gf_dens(~theta1)
```



```
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()
```

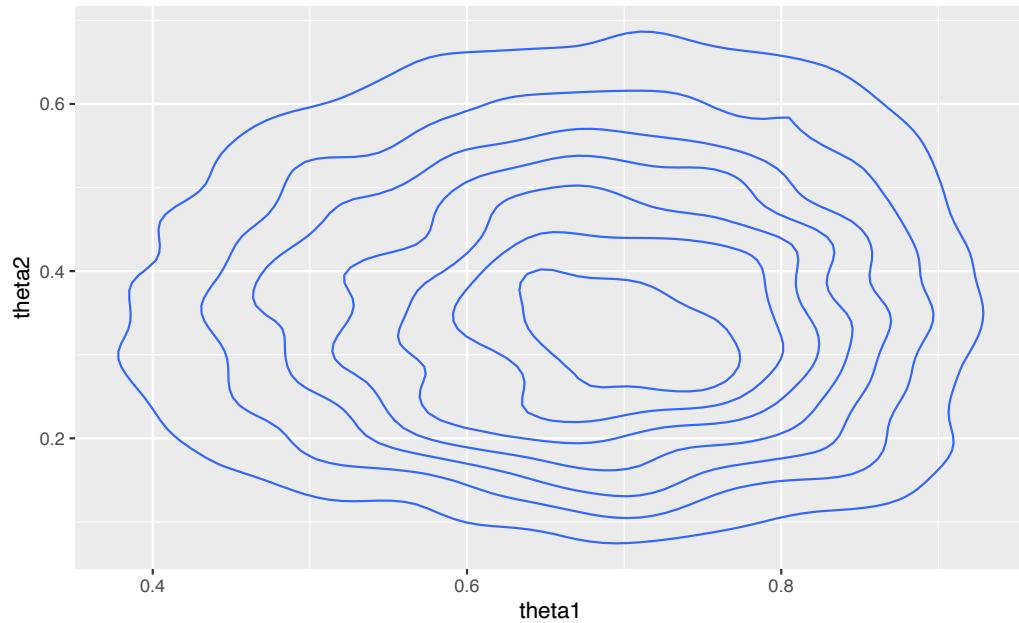


```
Gibbsplot <- Gibbs %>%
  filter(step < 500) %>%
  gf_path(theta2 ~ theta1, color = ~step, alpha = 0.5,
           arrow = arrow(type = "open", angle = 30)) +
  theme_minimal() + transition_reveal(step)

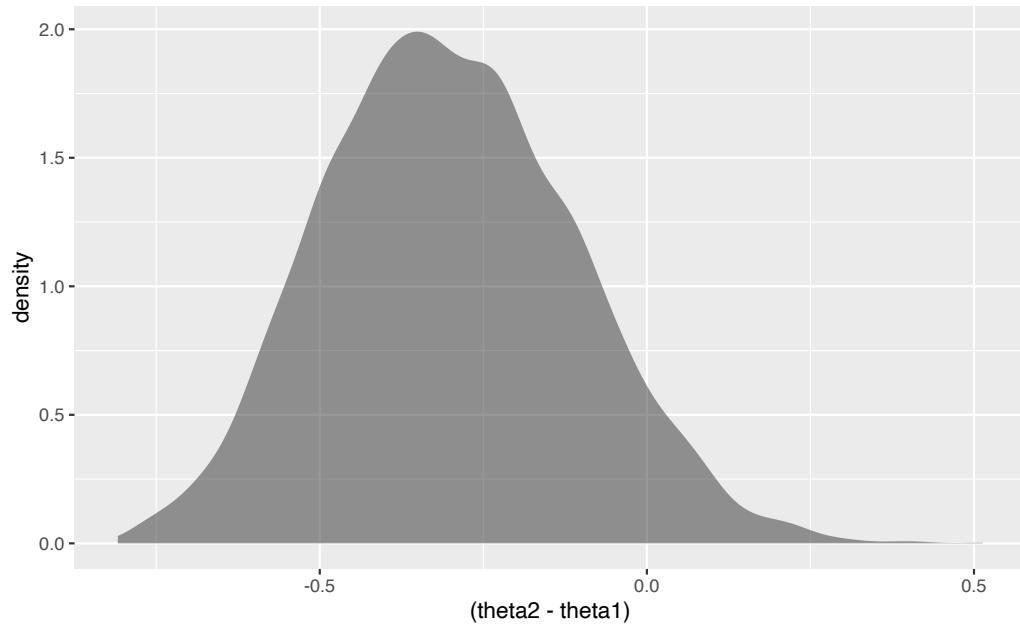
animate(Gibbsplot, fps = 1)
```

Ciclos completos

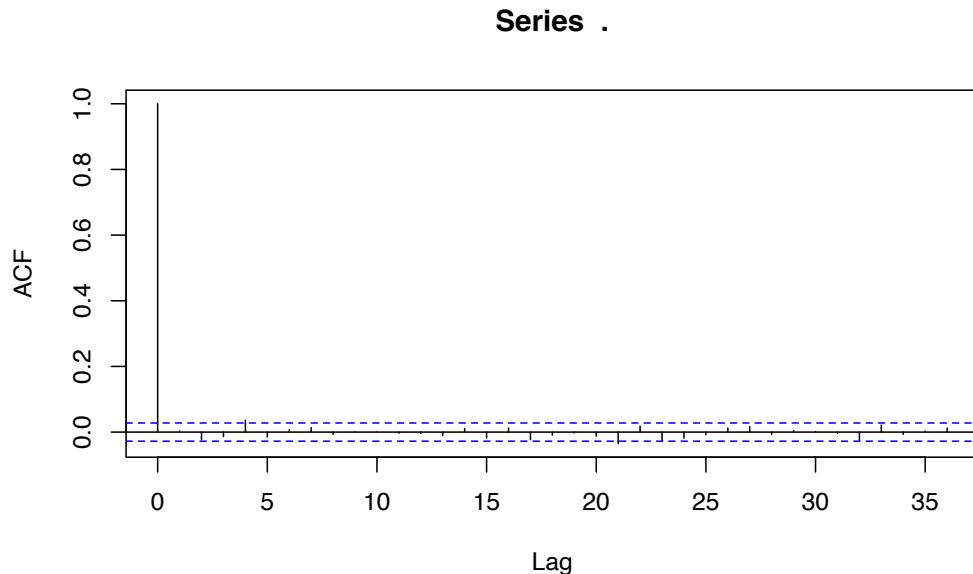
```
Gibbs %>%
  filter(step %% 2 == 0) %>%
  gf_density2d(theta2 ~ theta1)
```



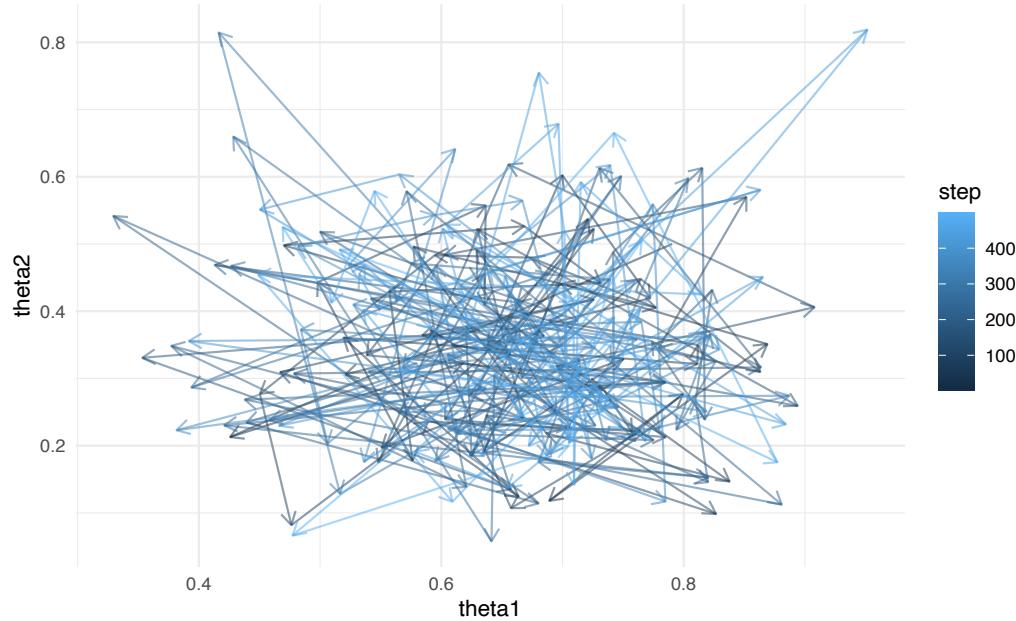
```
Gibbs %>%
  filter(step %>% 0) %>%
  gf_density(~(theta2 - theta1))
```



```
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)
```

9.10. 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)
  }
}

```

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

```

    theta ~ dbeta(1, 1)
}

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 "/var/folders/4d/qj4qr8zx1n36td0hlt0p7x_h0000gn/T//Rtmp...
## 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      0.305  0.063  0.193  0.261  0.303  0.346  0.436 1.001  3000
## deviance   62.031  1.300 61.088 61.186 61.519 62.313 65.727 1.005   920
##
## 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 = 0.8 and DIC = 62.9
## DIC is an estimate of expected predictive error (lower deviance is better).

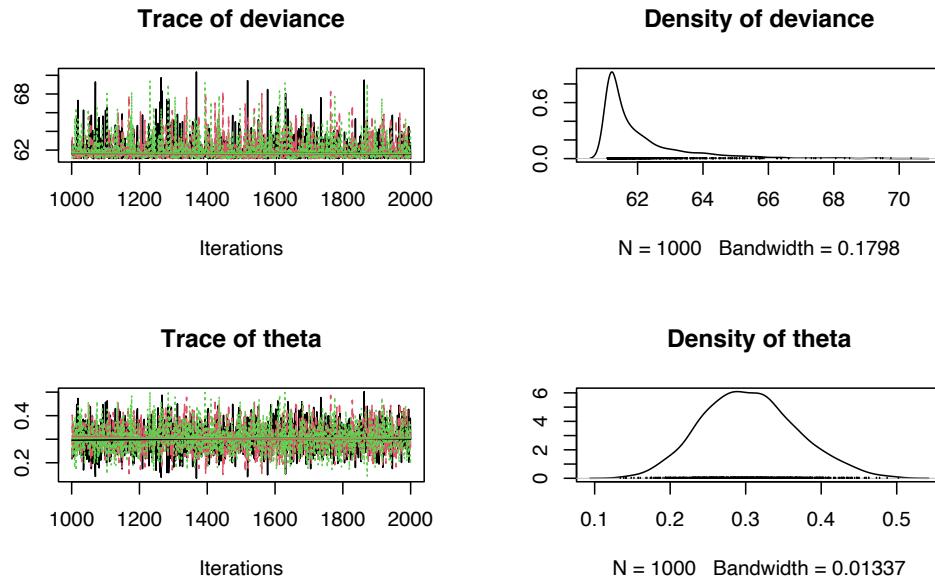
head(posterior(bern_jags))

## Error in h(simpleError(msg, call)): error in evaluating the argument 'x' in select
gf_dhistogram(~theta, data = posterior(bern_jags),
  bins = 50) %>%

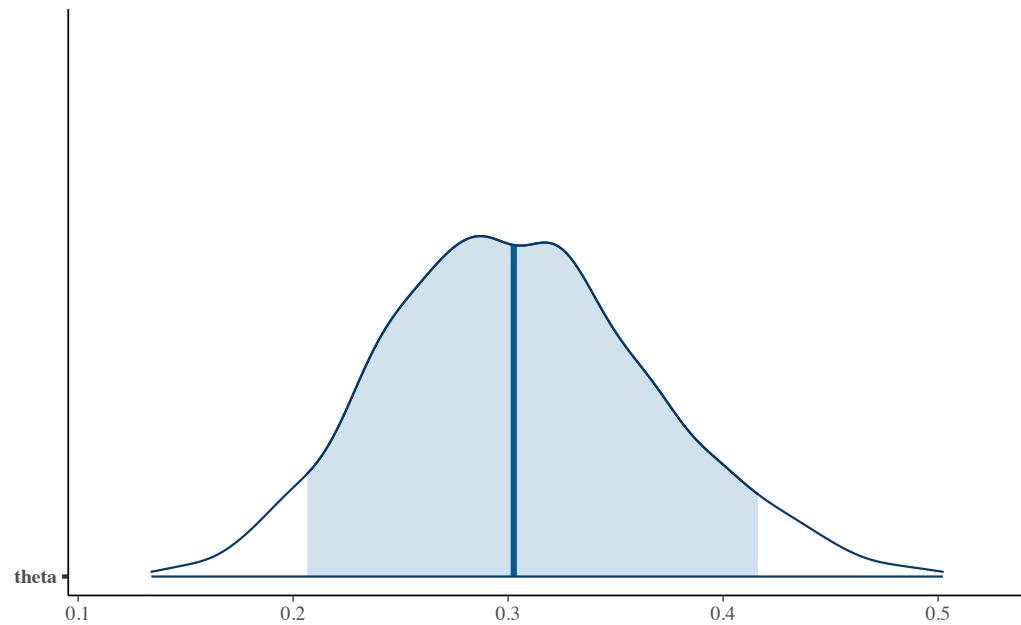
```

```
gf_dens(~theta, size = 1.5, alpha = 0.8) %>%
  gf_dist("beta", shape1 = 16, shape2 = 36, color = "red")
```

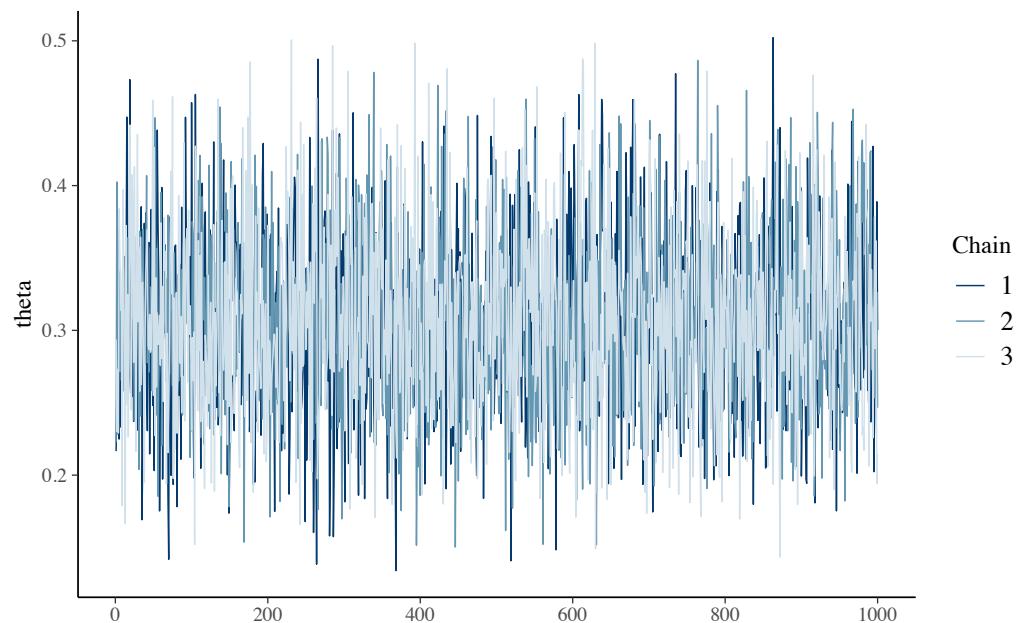
```
## Error in verosimilitud(theta, data): argument "data" is missing, with no de
bern_mcmc <- as.mcmc(bern_jags)
plot(bern_mcmc)
```



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



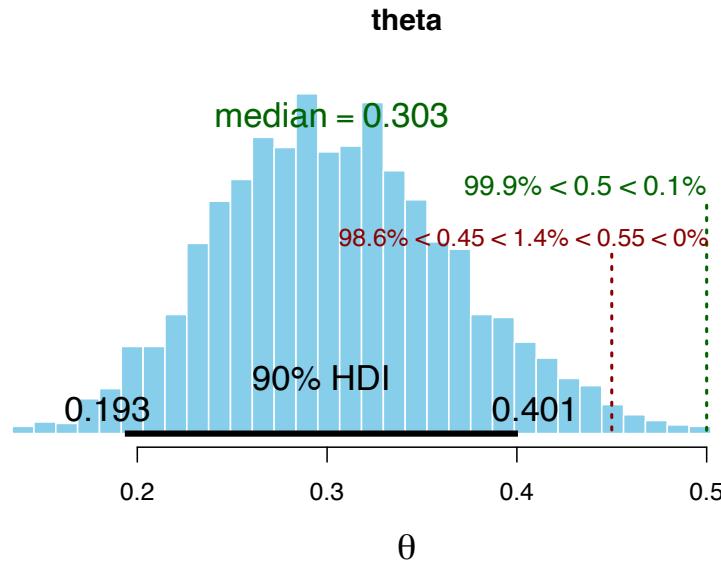
```
mcmc_trace(bern_mcmc, pars = "theta")
```



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

```
## Error in ‘combine_vars()’:
## ! 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(),
            attemps = n())

## # A tibble: 2 x 4
##   subject  prop_0  prop_1 attemps
##   <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
## 
```

```

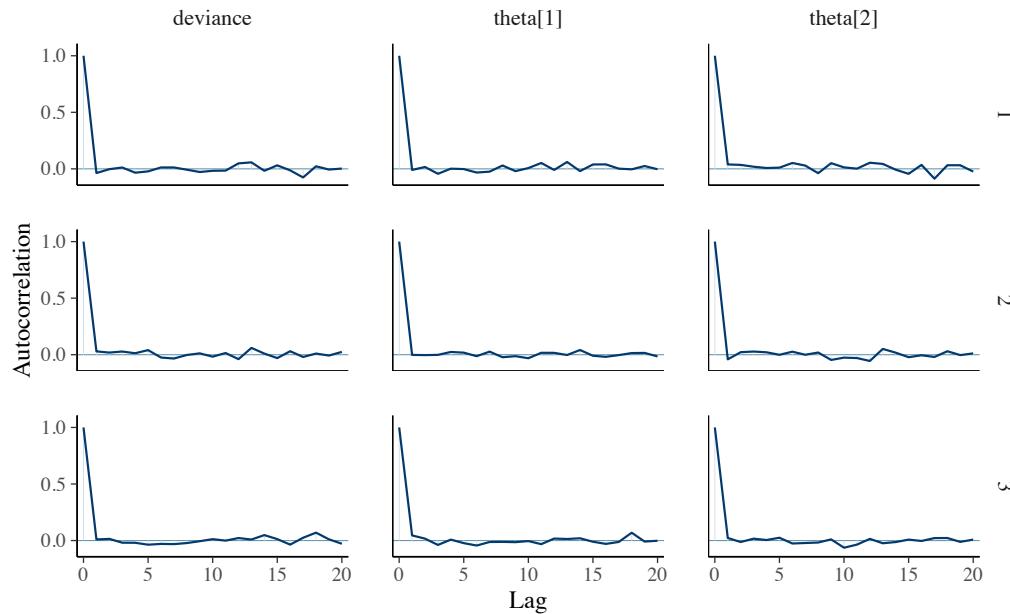
## $subject
## [1] 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2

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

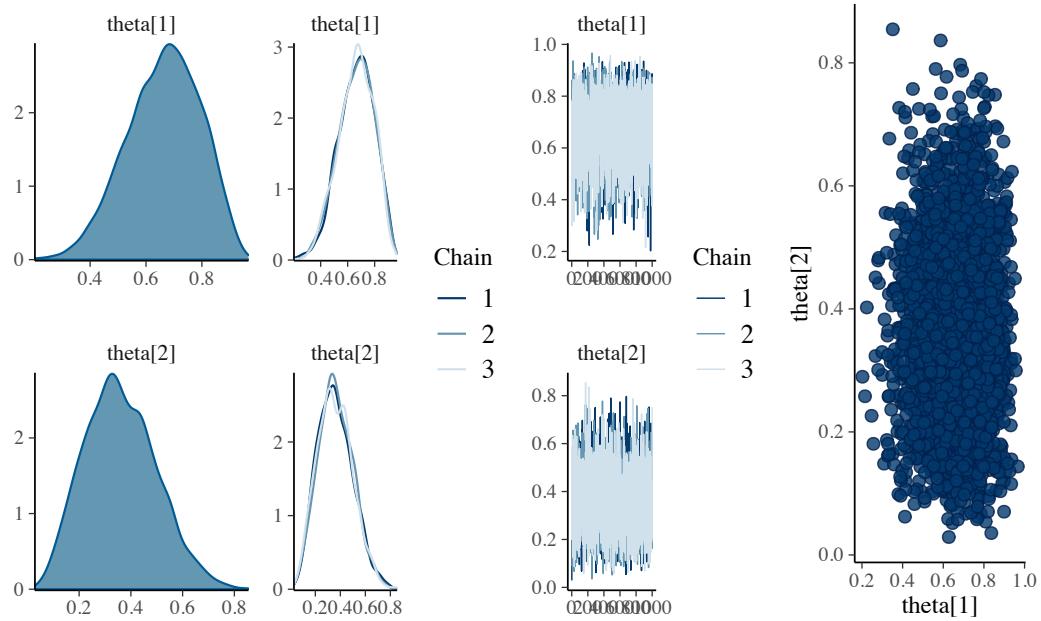
## Compiling model graph
## Resolving undeclared variables
## Allocating nodes
## Graph information:
##   Observed stochastic nodes: 15
##   Unobserved stochastic nodes: 2
##   Total graph size: 35
##
## Initializing model
bern2_mcmc <- as.mcmc(bern2_jags)

mcmc_acf(bern2_mcmc)

```



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



9.11. 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
```

²<https://mc-stan.org/>

```

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

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

model {
  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 2.1e-05 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0.21 s
## 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: 0.024206 seconds (Warm-up)
## Chain 1:           0.026863 seconds (Sampling)
## Chain 1:           0.051069 seconds (Total)
## Chain 1:
## 
## SAMPLING FOR MODEL '4584de91ce47196187979d2da8a67926' NOW (CHAIN 2).
## Chain 2:
## Chain 2: Gradient evaluation took 7e-06 seconds
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would take
## 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: 0.023686 seconds (Warm-up)
## Chain 2:           0.023843 seconds (Sampling)
## Chain 2:           0.047529 seconds (Total)
## Chain 2:
## 
## SAMPLING FOR MODEL '4584de91ce47196187979d2da8a67926' NOW (CHAIN 3).
## Chain 3:
## Chain 3: Gradient evaluation took 5e-06 seconds
## Chain 3: 1000 transitions using 10 leapfrog steps per transition would take
## 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: 0.023822 seconds (Warm-up)
## Chain 3: 0.022415 seconds (Sampling)
## Chain 3: 0.046237 seconds (Total)
## Chain 3:
## 
## SAMPLING FOR MODEL '4584de91ce47196187979d2da8a67926' NOW (CHAIN 4).
## Chain 4:
## Chain 4: Gradient evaluation took 6e-06 seconds
## Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 0.06 s
## 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: 0.024683 seconds (Warm-up)
```

```

## Chain 4:          0.02536 seconds (Sampling)
## Chain 4:          0.050043 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   3497    1
## lp__ -32.61    0.01 0.73 -33.48 -32.10   3903    1
##
## Samples were drawn using NUTS(diag_e) at Mon Jun 27 08:33:29 2022.
## 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] 0.3069652

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

##      10%      90%
## 0.2261308 0.3905893

shinystan::launch_shinystan(fit)

```

Ejercicio 9.1. Replique los resultados anteriores pero para el caso de 2 monedas y comente los resultados.

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, 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.
- Härdle, Wolfgang y col. (2004). *Nonparametric and Semiparametric Models*. Berlin, Heidelberg: Springer Berlin Heidelberg.
- 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.
- 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.