Notas Curso de Estadística

Maikol Solís

Índice general

Capítulo 1

Introducción

Capítulo 2

Estimación de densidades

2.1. Histograma

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

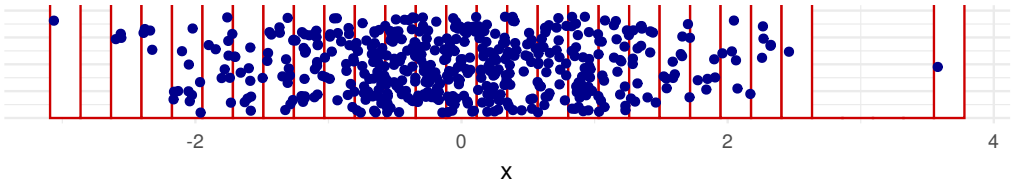
2.1.1. Construcción Estadística

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

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

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

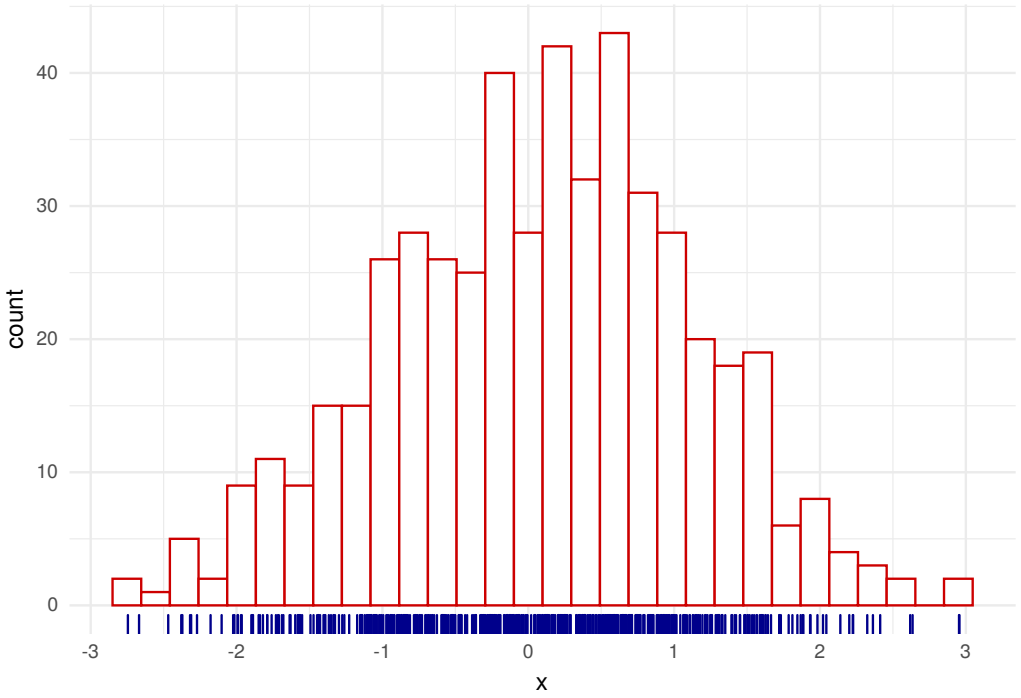
- Cuente cuántas observaciones caen en cada segmento. n_j.



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

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

- Dibuje el histograma.



Formalmente el histograma es el

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

donde I es la indicadora.

2.1.2. Construcción probabilistica

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

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

Esto se puede aproximar como

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

Acomodando un poco la expresión

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

2.1.3. Propiedades estadísticas

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

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

2.1.4. Sesgo

El cálculo del sesgo es el

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

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

Entonces

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

Entonces,

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

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

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

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

2.1.5. Varianza

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

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

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

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

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

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

2.1.6. Error cuadrático medio

El error cuadrático medio del histograma es el

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

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

Solución. Prueba segunda igualdad:

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

Ahora note que:

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

Entonces:

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

□

Retomando los términos anteriores se tiene que

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

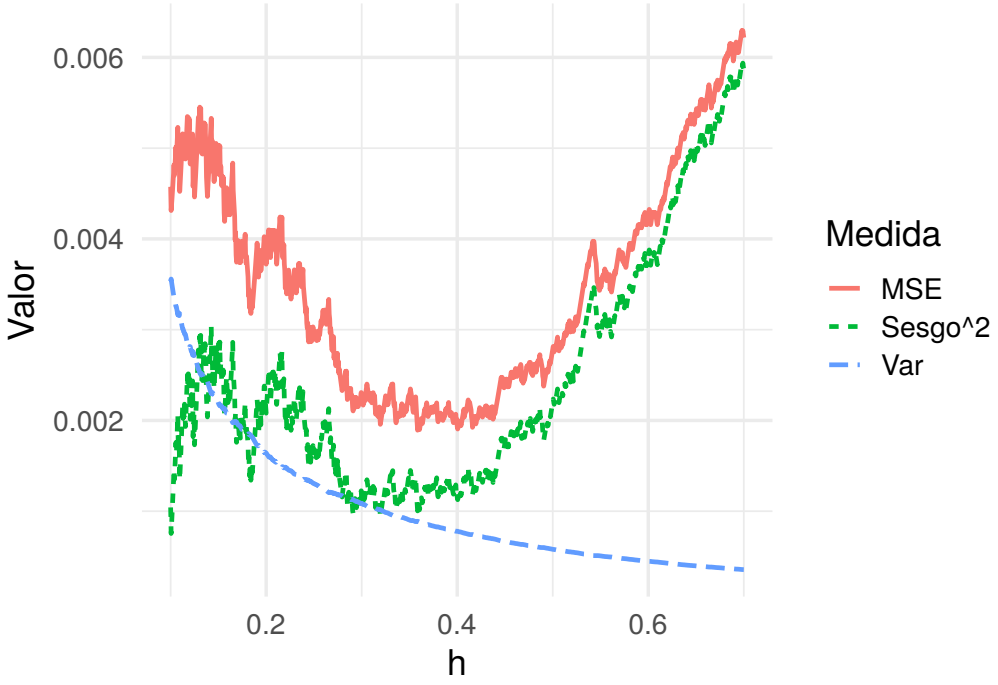
Nota: . Si $h \to 0$ y $nh \to \infty$ entonces $\text{MSE}\left(\hat{f}_h(x)\right) \to 0$. Es decir, conforme usamos más observaciones, pero el ancho de banda de banda no decrece tan rápida, entonces el error cuadrático medio converge a 0.

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

La fórmula anterior tiene la siguiente particularidad

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

Note que la figura siguiente tiene esa propiedad.



2.1.7. Error cuadrático medio integrado

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

La solución a esto es integrar el MSE.

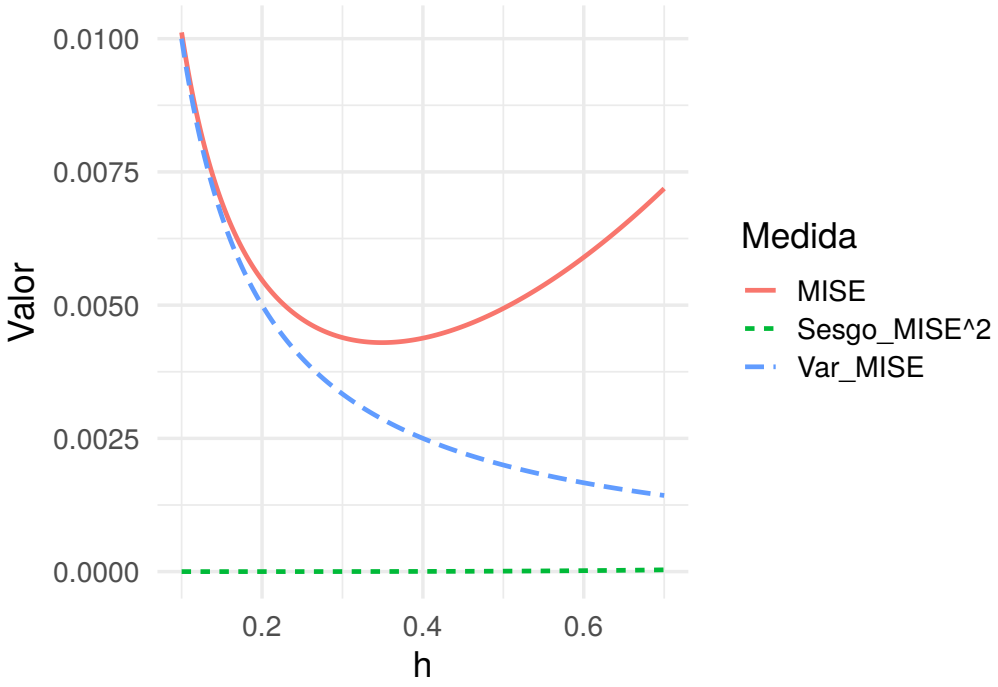
$$\begin{aligned}
\text{MISE}\left(\hat{f}_h(x)\right) &= \text{E}\left[\int_{-\infty}^{\infty}\left\{\hat{f}_h(x) - f(x)\right\}^2 dx\right] \\
&= \int_{-\infty}^{\infty}\text{E}\left[\left\{\hat{f}_h(x) - f(x)\right\}^2\right] dx \\
&= \int_{-\infty}^{\infty}\text{MSE}(\hat{f}_h(x))\, dx
\end{aligned}$$

Además,

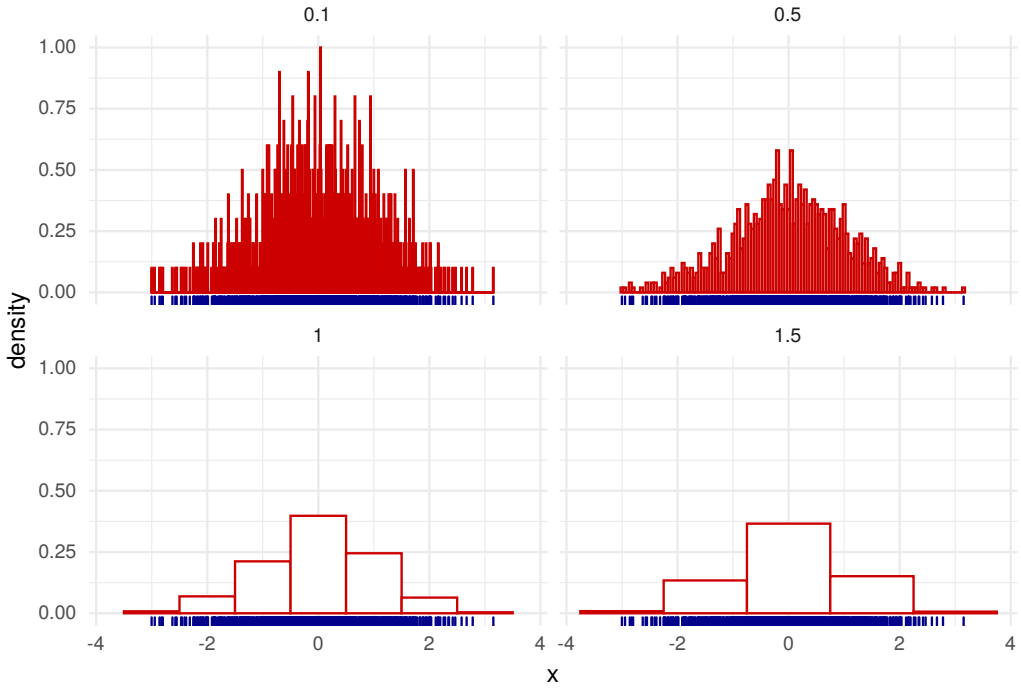
$$
\begin{aligned}
\mathrm{MISE}(\hat{f}_h(x)) &= \int_{-\infty}^{\infty} \frac{1}{nh} f(x)dx \\
&+ \int_{-\infty}^{\infty} \sum_j I(x \in B_j) \left\{ \left(j - \frac{1}{2}\right) h - x \right\}^2 \left[f' \left(\left\{ j - \frac{1}{2} \right\} h \right) \right]^2 dx \\
&= \frac{1}{nh} + \sum_j \left[f'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

2.1.8. Ancho de banda óptimo para el histograma

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



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



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

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

implica que

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

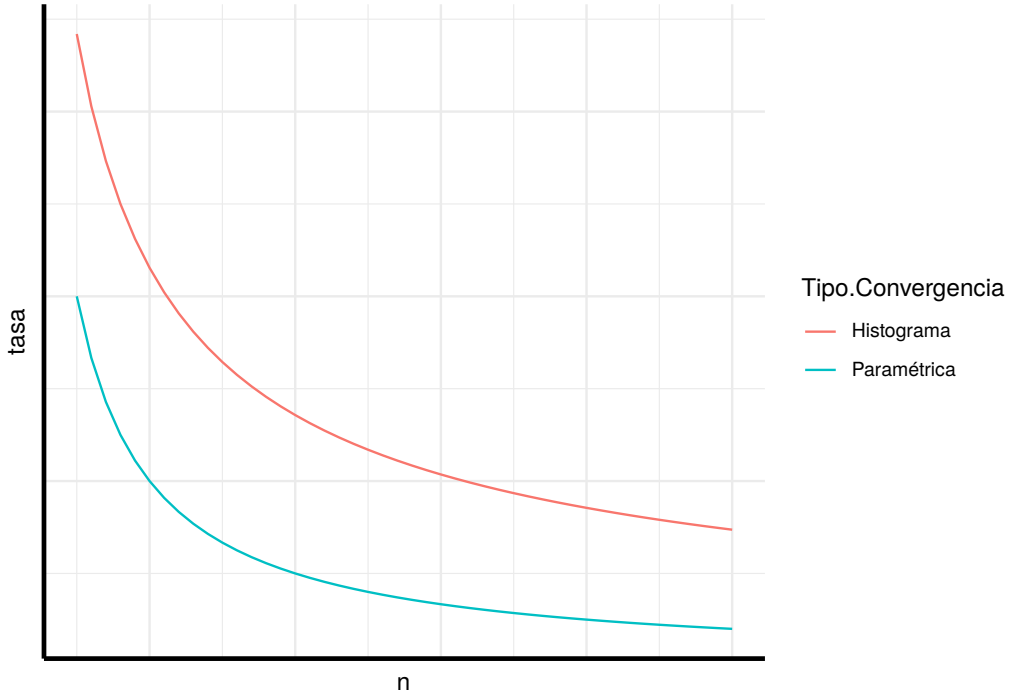
y que por lo tanto

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

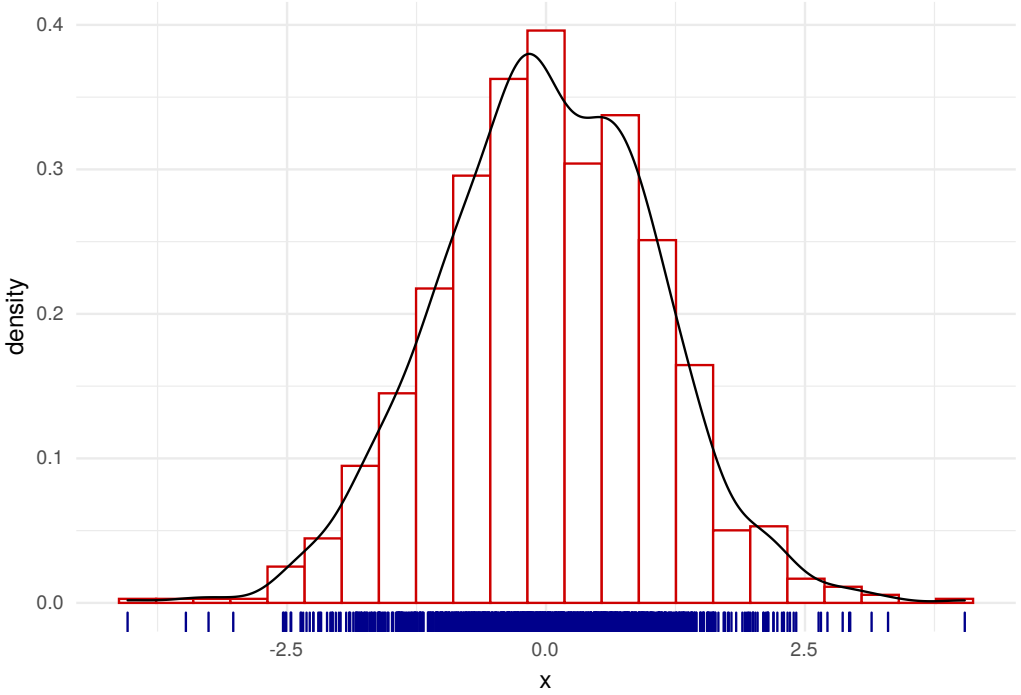
Nota: (Recuerde de Estadística I). Si $X_1, \ldots, X_2 \sim f_\theta$ i.i.d, con $\text{Var}(X) = \sigma^2$, recuerde que el estimador $\hat{\theta}$ de θ tiene la característica que

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

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



Finalmente, podemos encontrar el valor óptimo de esta datos dado por



$h =$`h_opt_MISE`

2.2. Estimación No-paramétrica de densidad

2.2.1. Primera construcción

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

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

Considere la distribución empírica como

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

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

para todo x in $\mathbb{R}$.

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

La respuesta es si (más o menos).

Suponga que $h > 0$ tenemos la aproximación

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

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

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

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

Podemos rescribirlo de la forma,

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

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

2.2.2. Otra construcción

Con el histograma construimos una serie de segmentos fijo B_j y 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 histograma se escribe como

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

Ahora tratemos de modificar ligeramente esta expresión notando dos cosas

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

con $u = \frac{x-xi}{h}$

2.

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

\end{enumerate}

Finalmente se tiene que

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

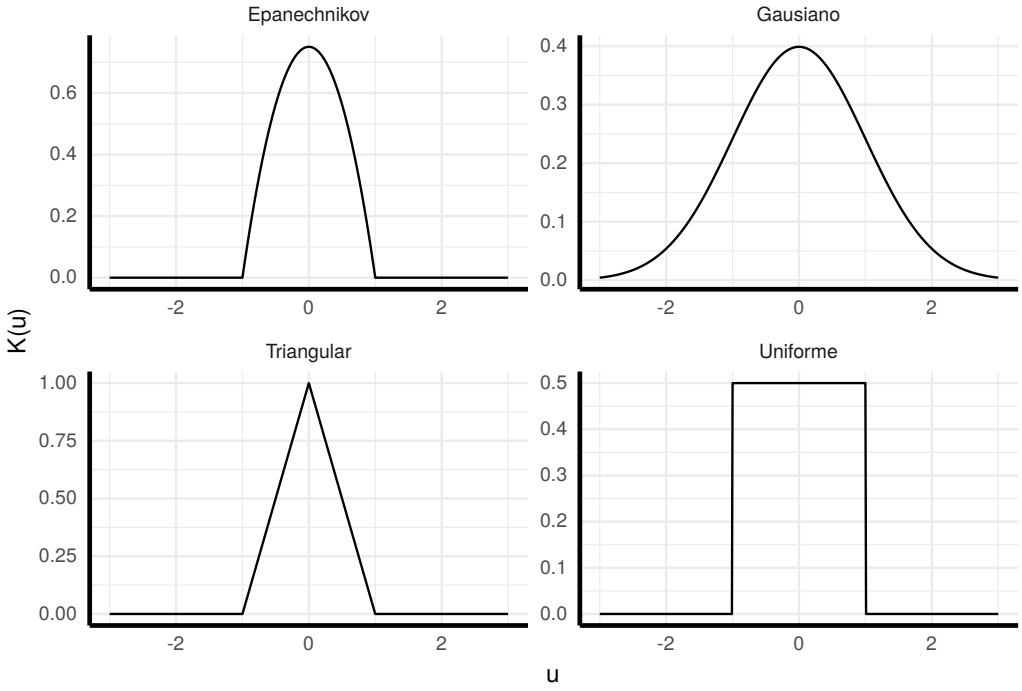
Nota: . ¿Qué pasaría si cambiaríamos la función K del histograma por una más general?

Esta función debería cumplir las siguientes características

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

Por ejemplo:

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

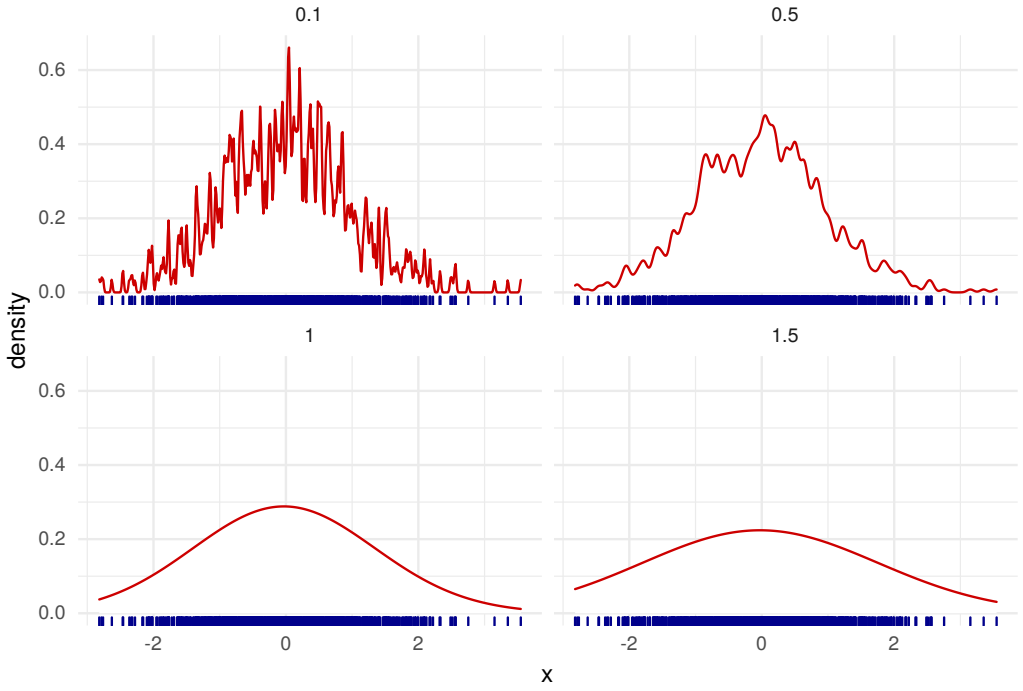


Entonces se tendría que la expresión general para un estimador por núcleos es

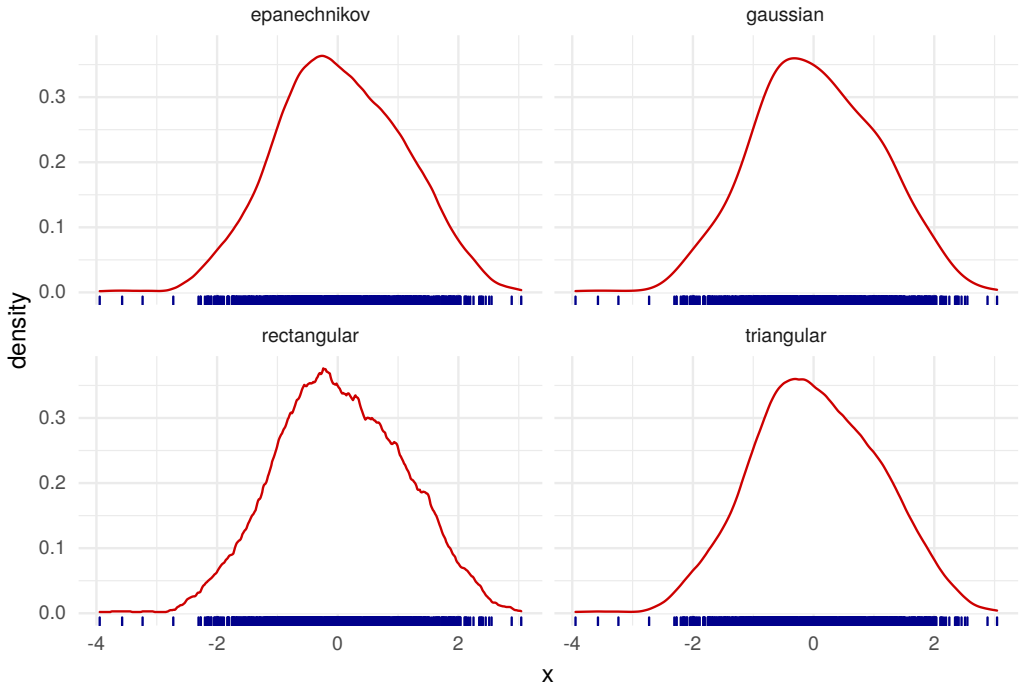
$$\hat{f}_h(x) = \frac{1}{nh} \sum_{i=1}^{n} K\left(\frac{x - x_i}{h}\right)$$

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

Nuevamente sería el ancho de banda ya que



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

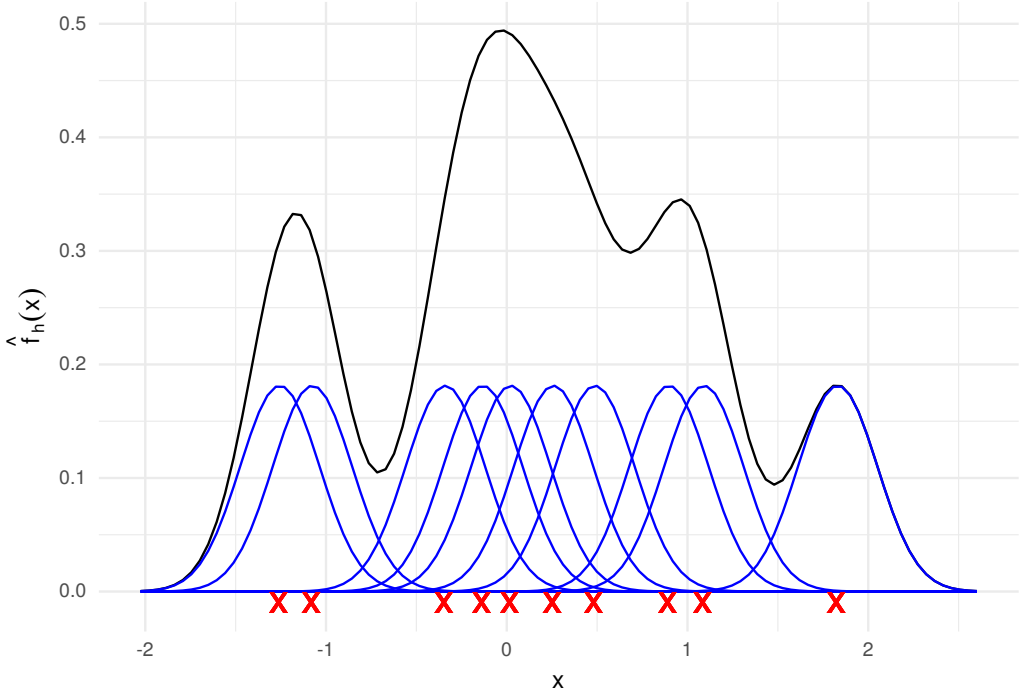


Recordemos nuevamente la fórmula

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

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

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



2.3. Propiedades Estadísticas

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

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

Entonces:

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

donde

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

2.3.1. Varianza

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

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

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

2.3.2. Sesgo

Para el sesgo tenemos

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

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

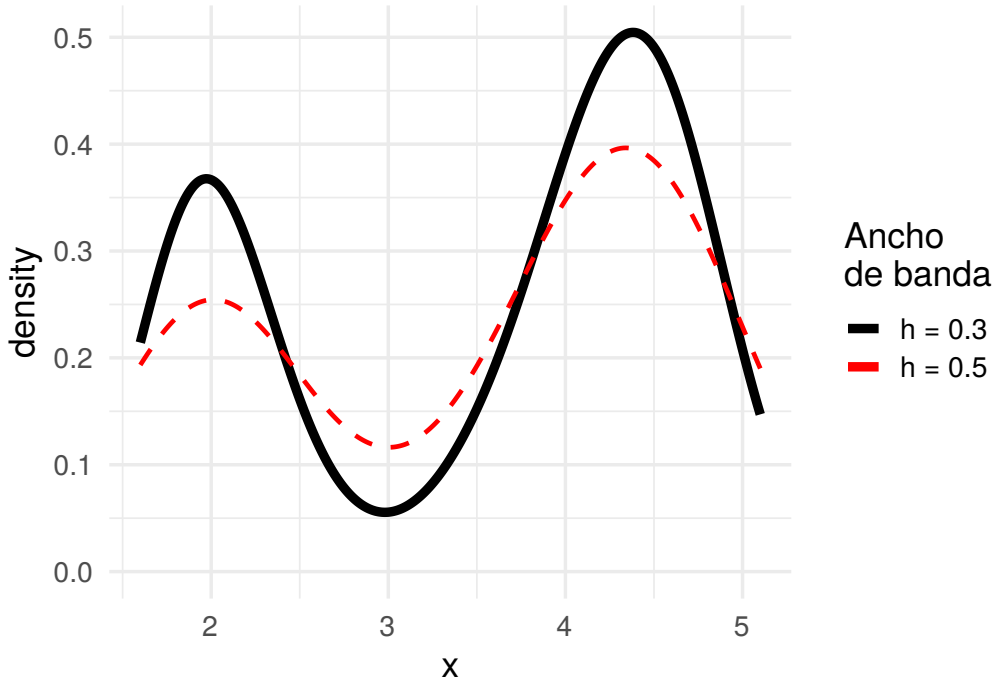
$$\text{Sesgo}\left(\hat{f}_h(x)\right) = \frac{h^2}{2}f''\mu_2(K) + o(h^2), \text{ si } h \to 0$$

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

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

Solución.

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



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

2.3.3. Error cuadrático medio y Error cuadrático medio integrado

El error cuadrático medio se escribe

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

Y el error cuadrático medio integrado se escribe como,

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

2.3.4. Ancho de banda óptimo

Minimizando el MISE con respecto a h obtenemos

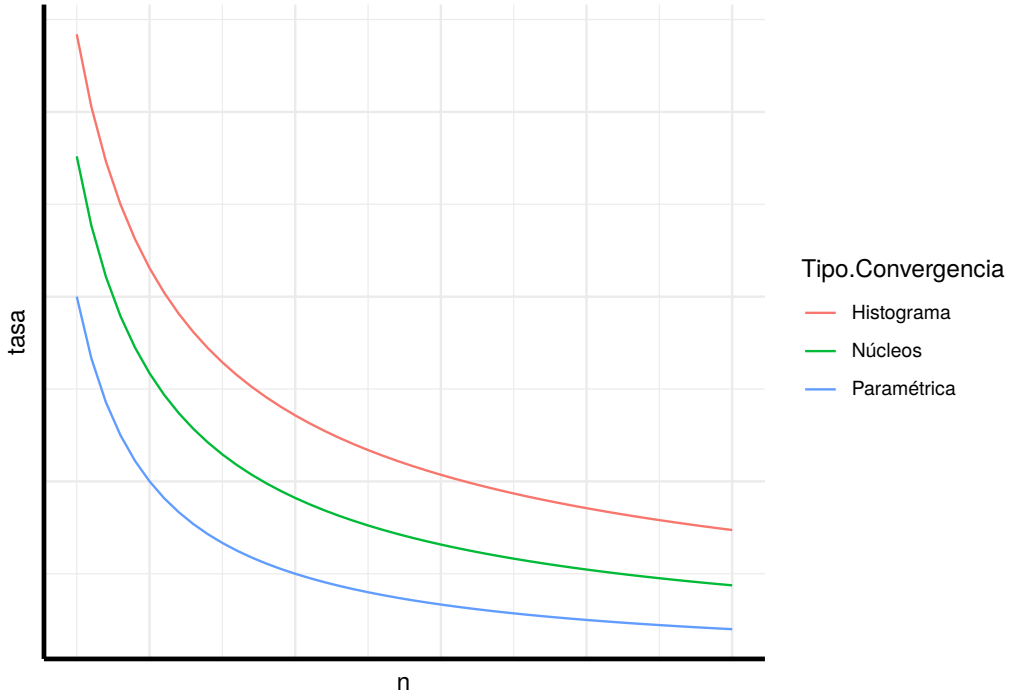
$$h_{opt} = \left(\frac{\|K\|_2^2}{\|f''\|_2^2 \left(\mu_2(K)\right)^2 n} \right)^{1/5} = O\left(n^{-1/5}\right).$$

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

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



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

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

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

2.4. Escogiendo el ancho de banda

Nota: . La principal característica del ancho de banda

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

ES QUE ¡NO FUNCIONA!

Veremos dos métodos para determinar un h que funcione:

- Referencia normal.
- Validación cruzada.

2.4.1. Referencia normal

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

Más adelante veremos otro método para densidades más generales.

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

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

donde

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

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

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

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

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

El rango intercuantil IQR se define como

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

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

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

Entonces,

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

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

Podemos sustituir la varianza empírica de la fórmula inicial y tenemos

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

Combinando ambos estimadores, podemos obtener,

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

2.4.2. Validación Cruzada

Defina el *error cuadrático integrado* como

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

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

Nuestro objetivo es minimizar el ISE con respecto a h.

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

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

Vamos a resolver esto en dos pasos partes

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

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

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

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

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

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

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

donde

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

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

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

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

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

Finalmente tenemos la función,

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

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

Nota: . Para efectos prácticos es mejor utilizar el criterio

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

y luego calcular numéricamente la integral.

2.4.3. Intervalos de confianza para estimadores de densidad no paramétricos

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

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

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

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

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

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

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

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

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

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

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

Existe una versión más general para determinar la banda de confianza de toda la función. Por favor revisar la página 62 de (Härdle y col. 2004).

2.5. Laboratorio

Comenzaremos con una librería bastante básica llamada `KernSmooth`.

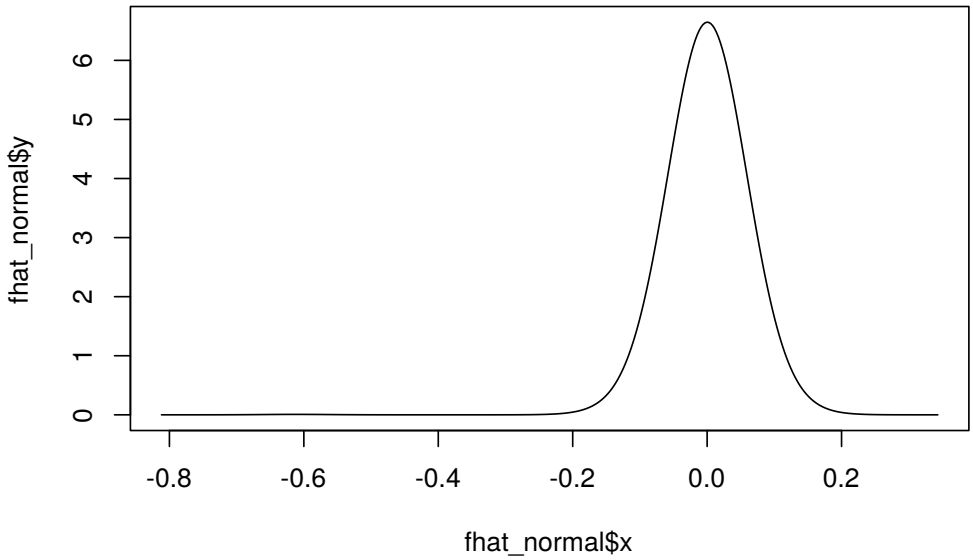
2.5.1. Efecto de distintos Kernels en la estimación

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

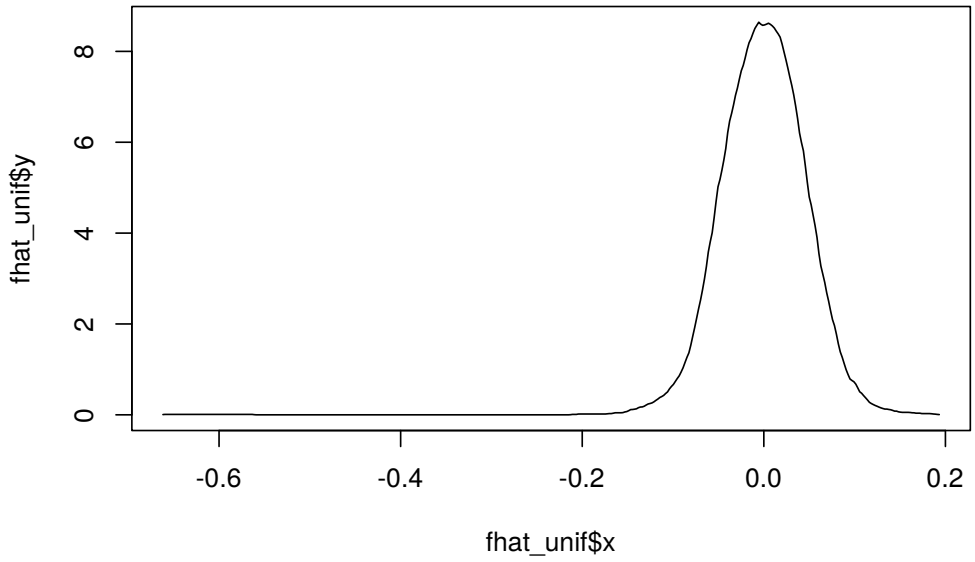
```
summary(x)
```

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

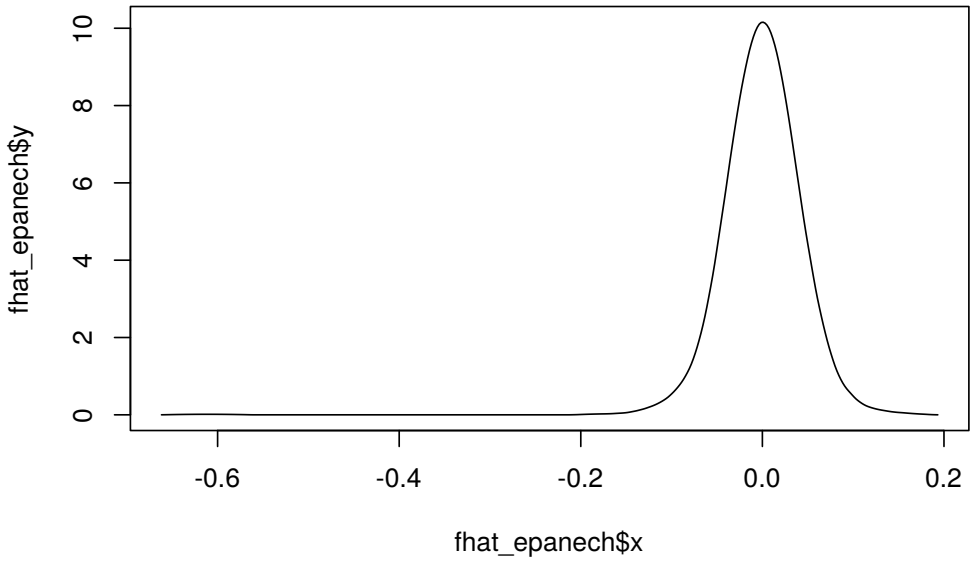
```
library(KernSmooth)

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



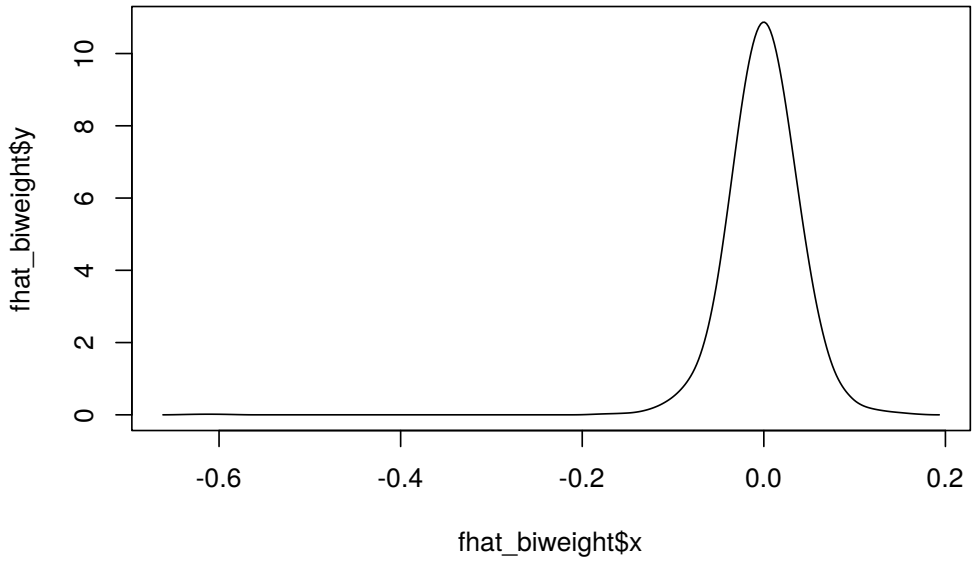

```
fhat_unif <- bkde(x, kernel = "box", bandwidth = 0.05)
plot(fhat_unif, type = "l")
```



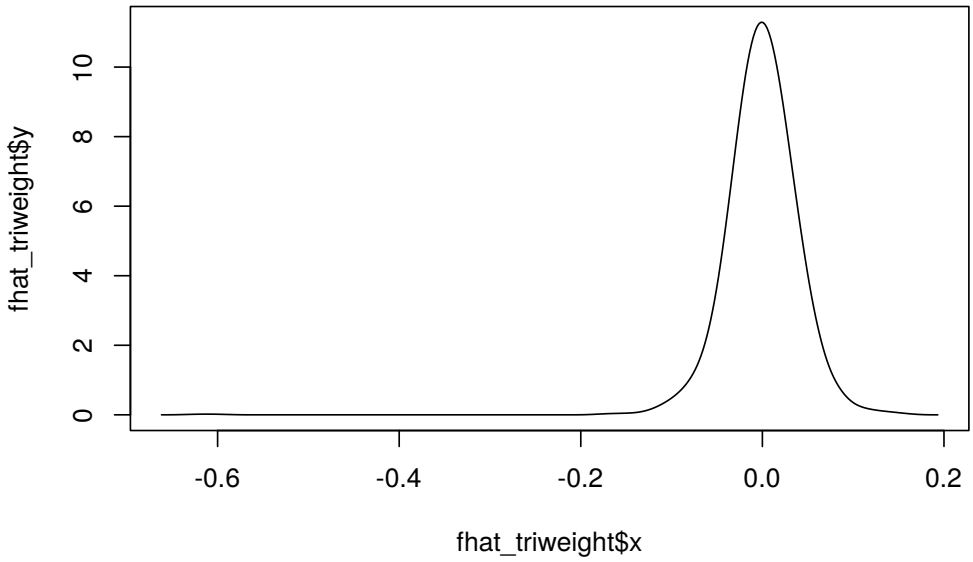

```
fhat_epanech <- bkde(x, kernel = "epanech", bandwidth = 0.05)
plot(fhat_epanech, type = "l")
```




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



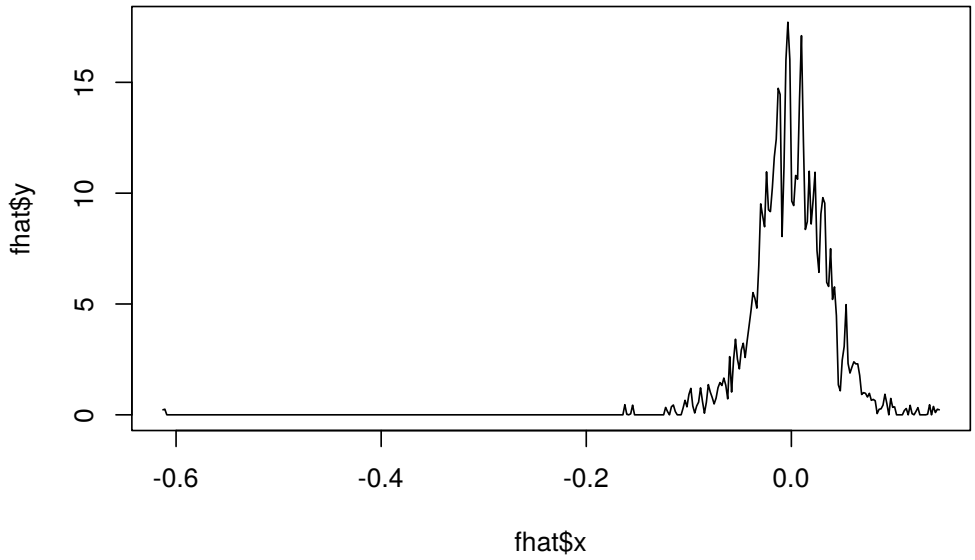

```
fhat_triweight <- bkde(x, kernel = "triweight", bandwidth = 0.05)
plot(fhat_triweight, type = "l")
```



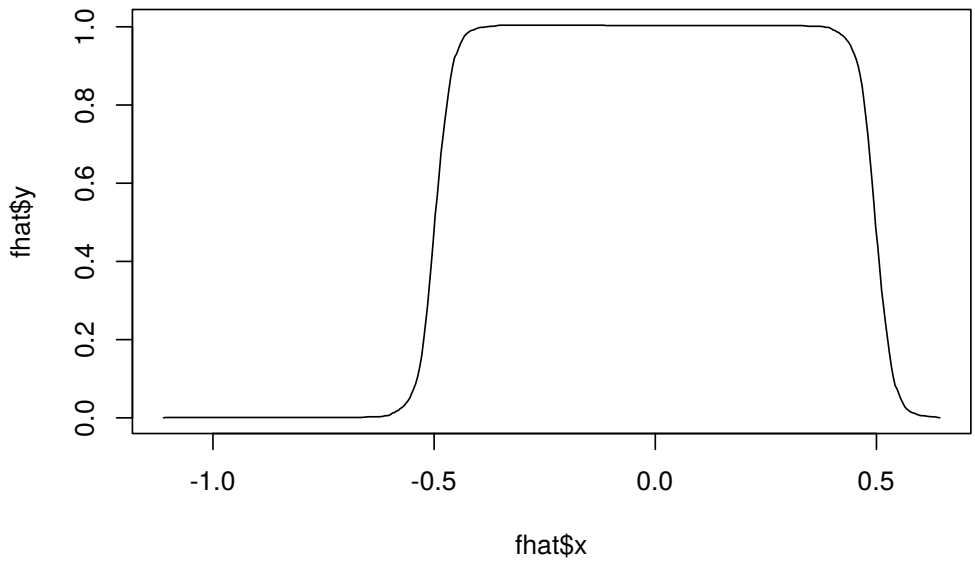
2.5.2. Efecto del ancho de banda en la estimación

** Kernel uniforme **

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

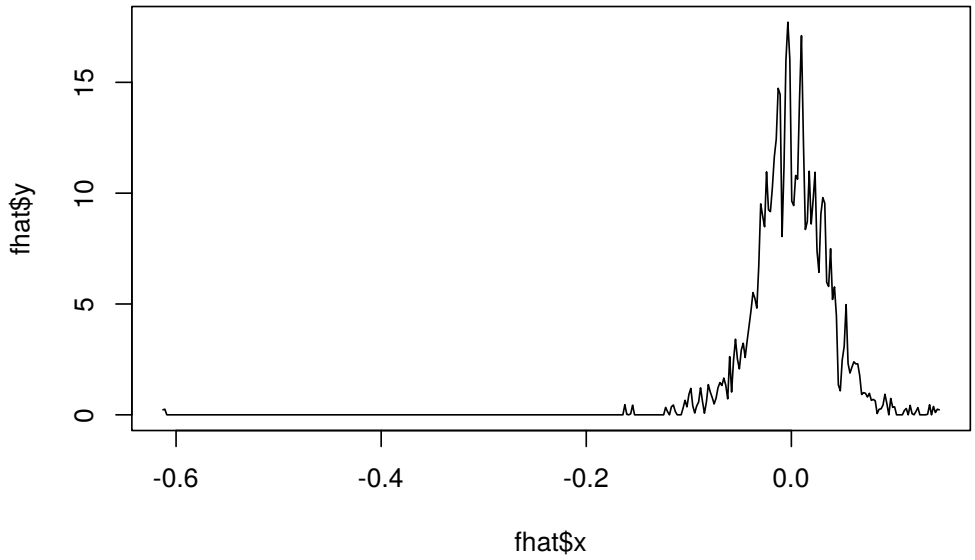



```
fhat <- bkde(x, kernel = "box", bandwidth = 0.5)
plot(fhat, type = "l")
```

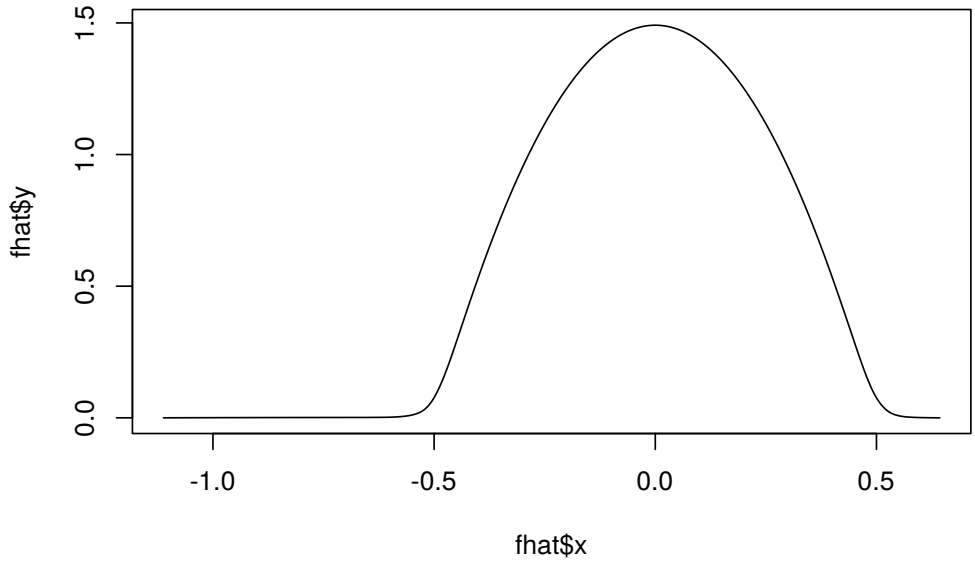


** Kernel Epanechnikov **

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




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

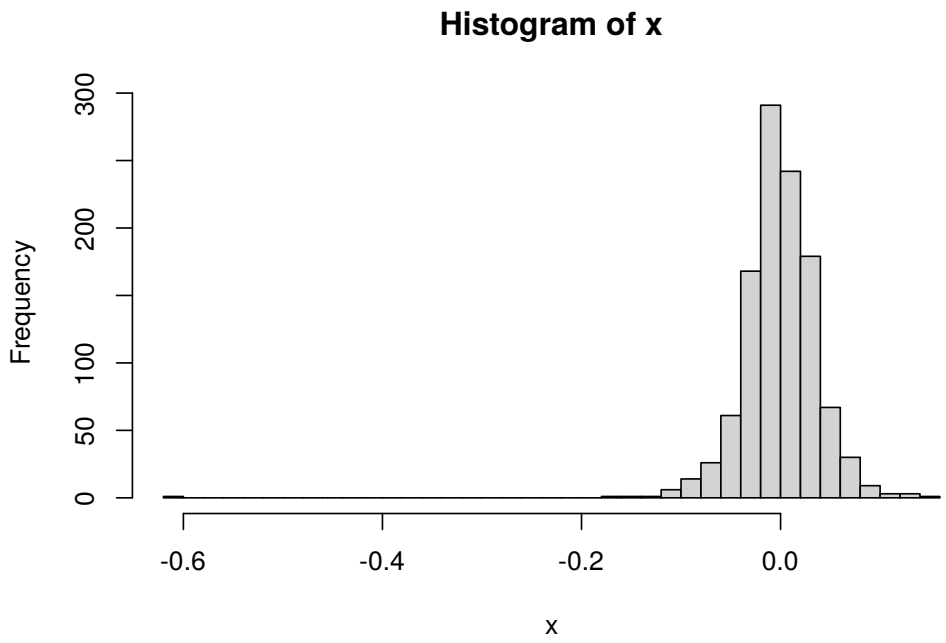



```
suppressMessages(library(tidyverse))
library(gganimate)

fani <- tibble()

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

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

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

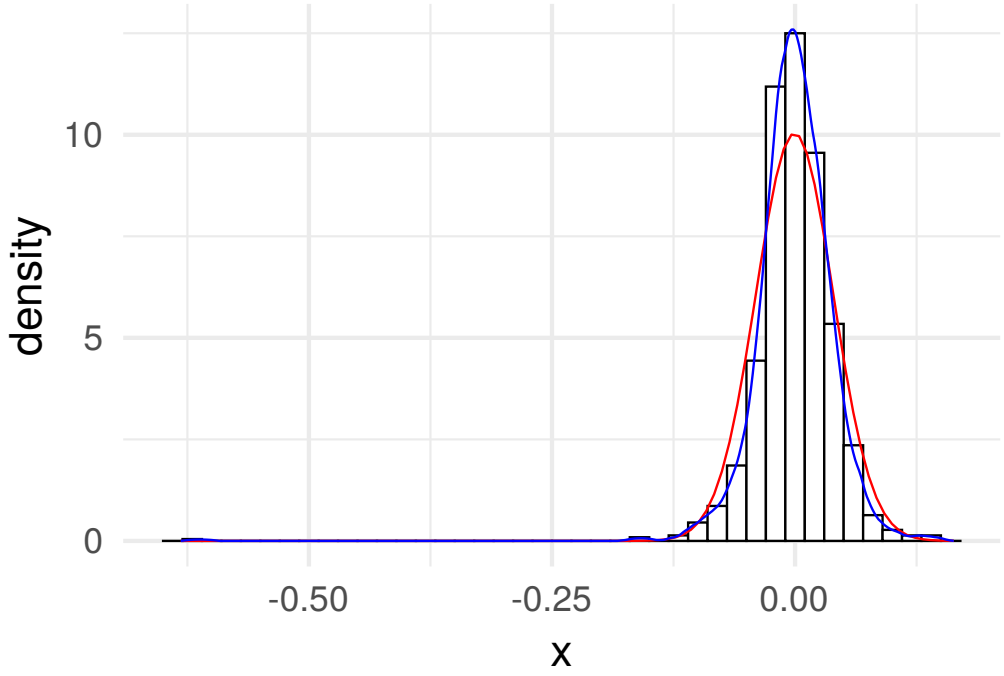
Nota: . - Construya una variable llamada `u` que sea una secuencia de -0.15 a

0.15 con un paso de 0.01 - Asigne `x` a los datos `stockrel` y calcule su media y varianza. - Usando la función `dnorm` construya los valores de la distribución de los datos usando la media y varianza calculada anteriormente. Asigne a esta variable `f\_param`. - Defina un ancho de banda `h` en 0.02 - Construya un histograma para estos datos con ancho de banda `h`. Llame a esta variable `f\_hist` - Usando el paquete `KernSmooth` y la función `bkde`, construya una función que calcule el estimador no paramétrico con un núcleo Epanechivok para un ancho de banda h. Llame a esta variable `f\_epa`. - Dibuje en el mismo gráfico la estimación paramétrica y no paramétrica.

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

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

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

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

h <- 0.02

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

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




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

x_df <- data.frame(x)

library(ggplot2)

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



2.5.3. Ancho de banda óptimo

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

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

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

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

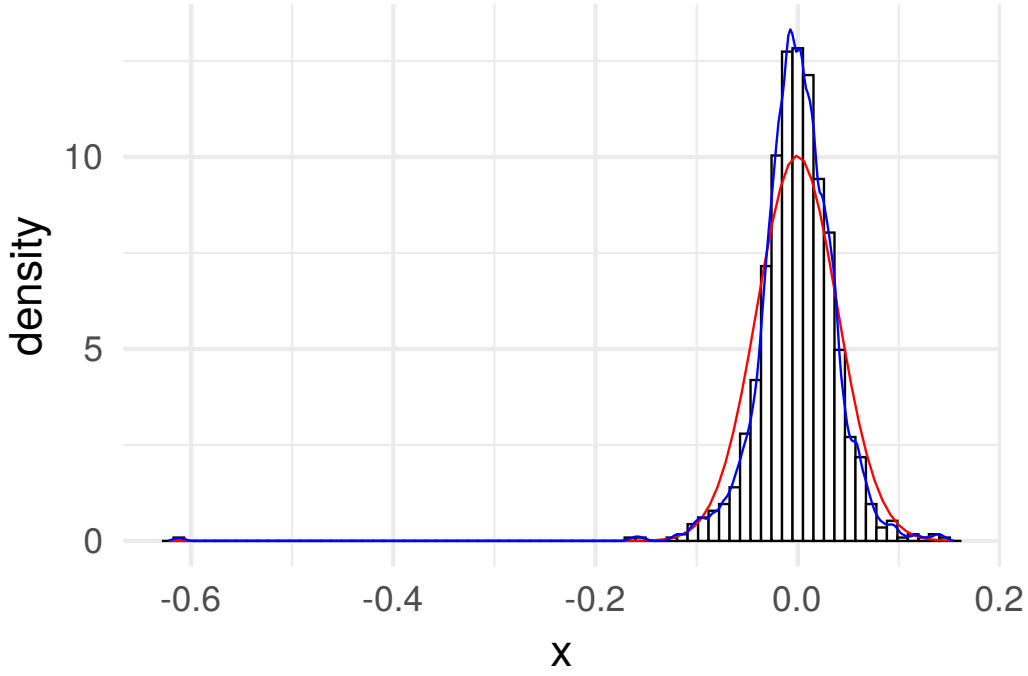
Un estimador más robusto es

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

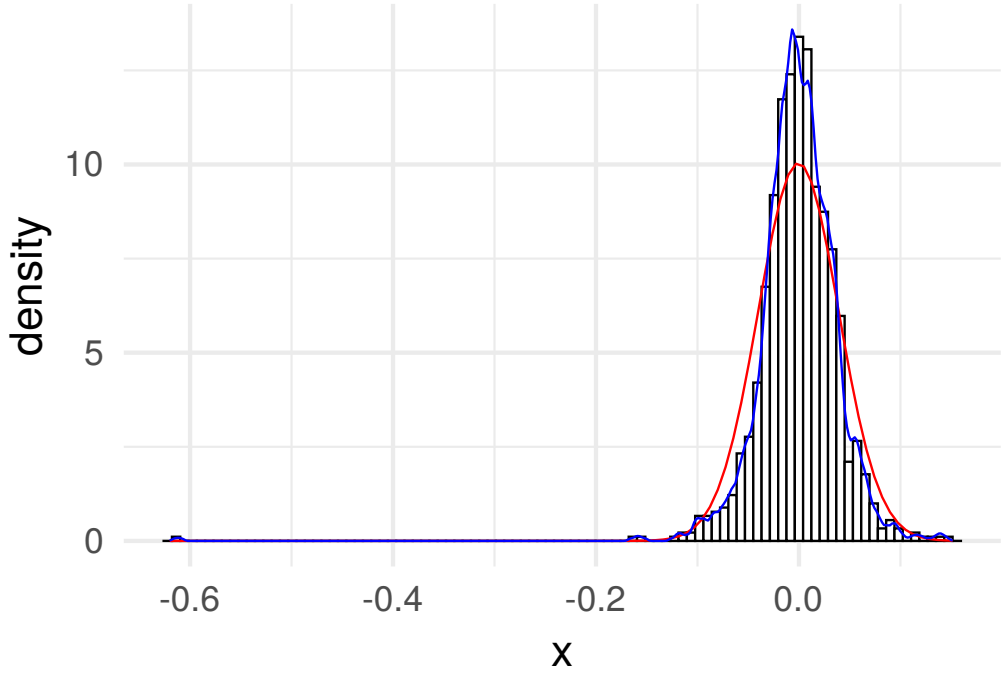
¿Por qué es $IQR/1{,}34$?

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

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

h <- h_normal

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

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

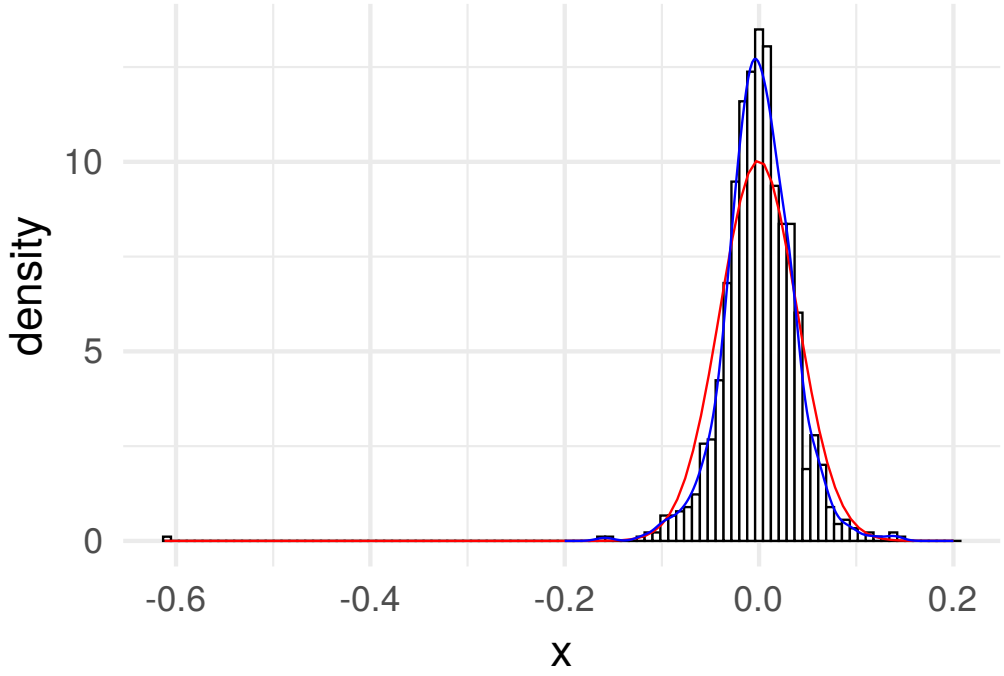



```
h_iqr <- 1.06 * min(s, IQR(x)/1.34) * n^(-1/5)

h <- h_iqr

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

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



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

```
library(np)

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

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

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

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

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



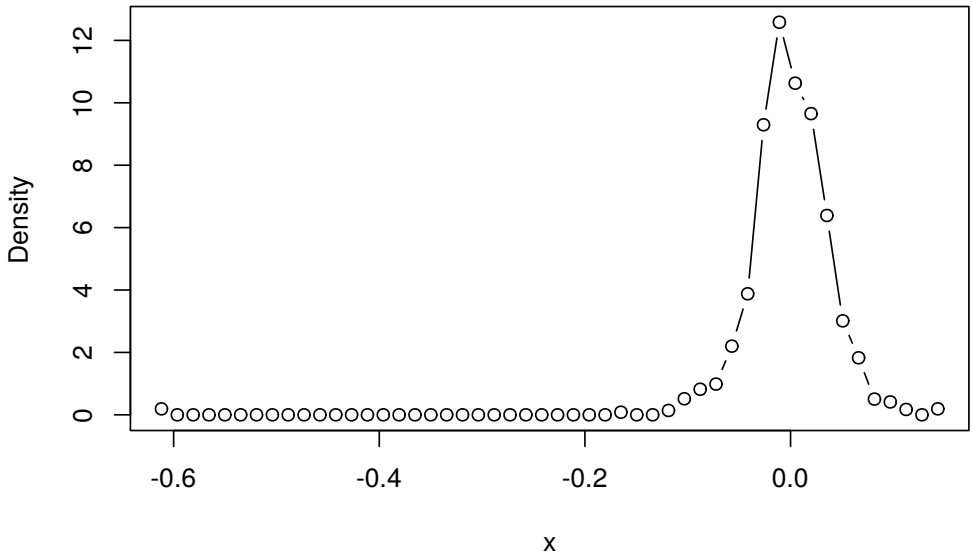
2.5.4. Validación cruzada

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

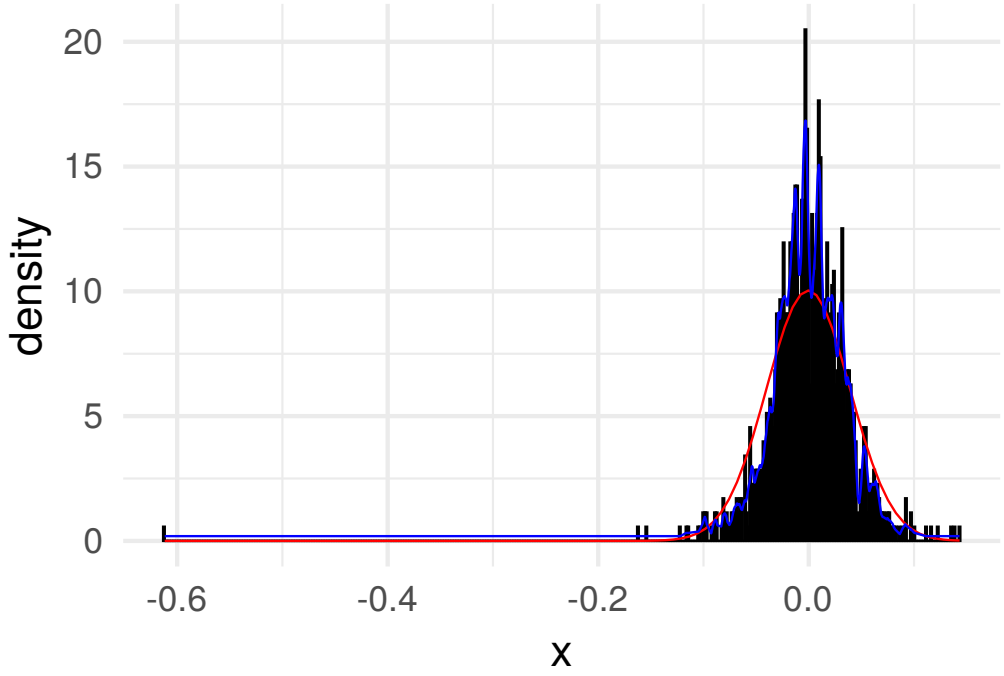
```
h_cv_np_ls <- npudensbw(dat = x, bwmethod = "cv.ls",
    ckertype = "epa", ckerorder = 2)
```

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

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

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




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

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



2.5.5. Temas adicionales

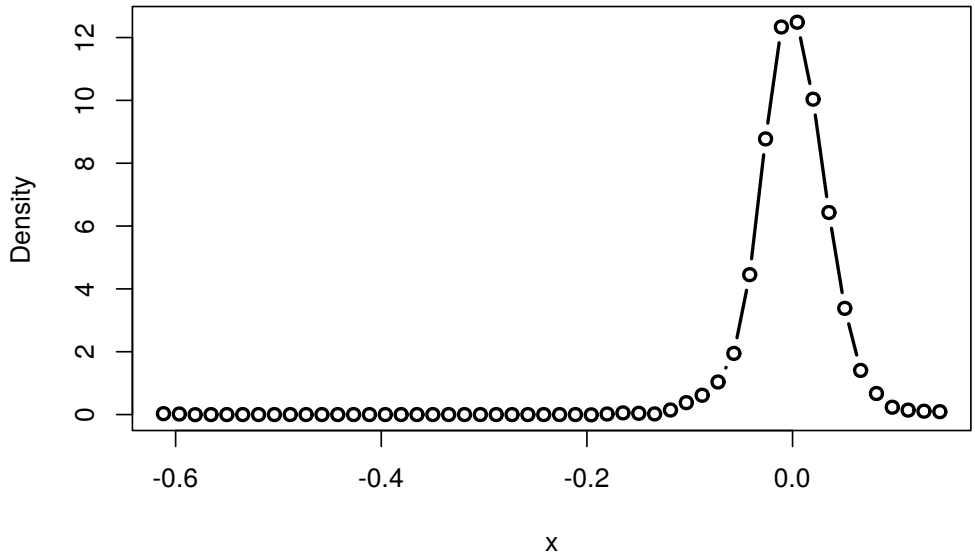
** Reducción del sesgo ** Como lo mencionamos en el texto, una forma de mejorar el sesgo en la estimación es suponer que la función de densidad es más veces diferenciable.

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

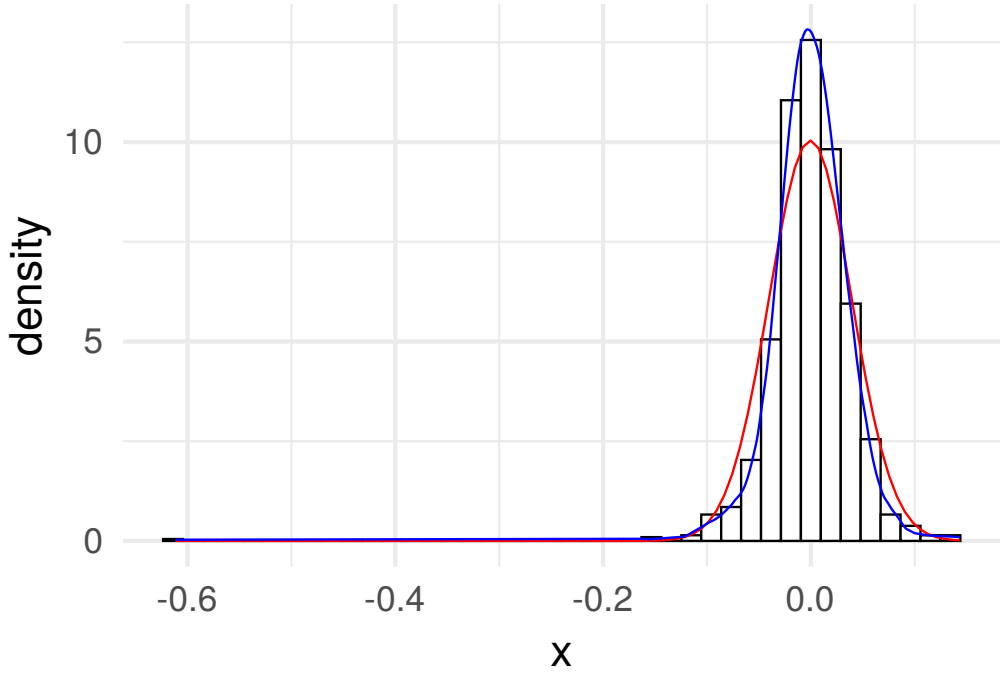
```
h_cv_np_ls <- npudensbw(dat = x, bwmethod = "cv.ls",
    ckertype = "epa", ckerorder = 4)
```

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

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

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




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

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

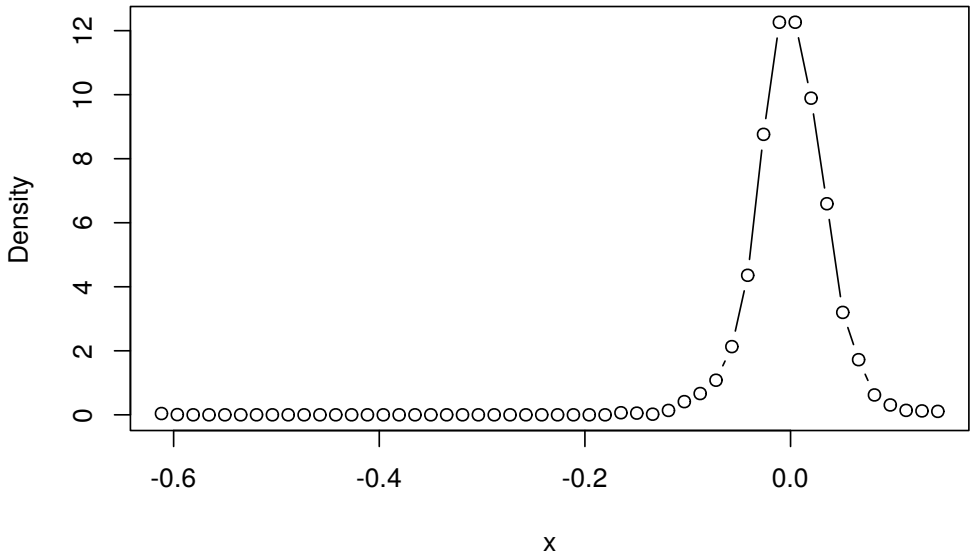


Otra forma de estimar el ancho de banda Otra forma de estimar ancho de bandas óptimos es usando máxima verosimilitud. Les dejo de tarea revisar la sección 1.1 del artículo de (Hall 1987) para entender su estructura.

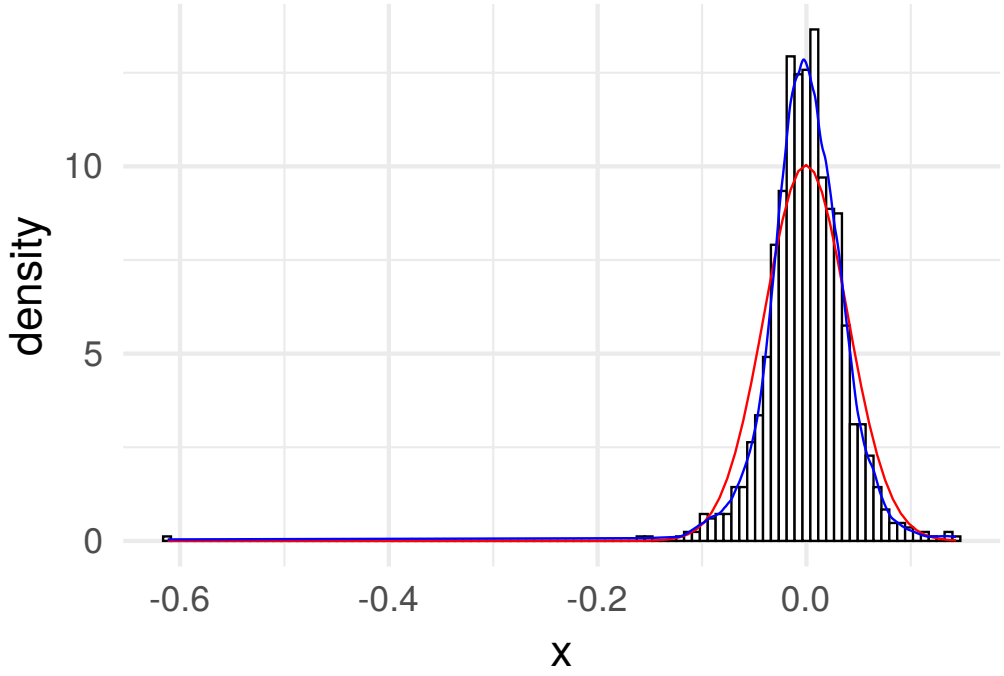
```
h_cv_np_ml <- npudensbw(dat = x, bwmethod = "cv.ml",
    ckertype = "epanechnikov")
```

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

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

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



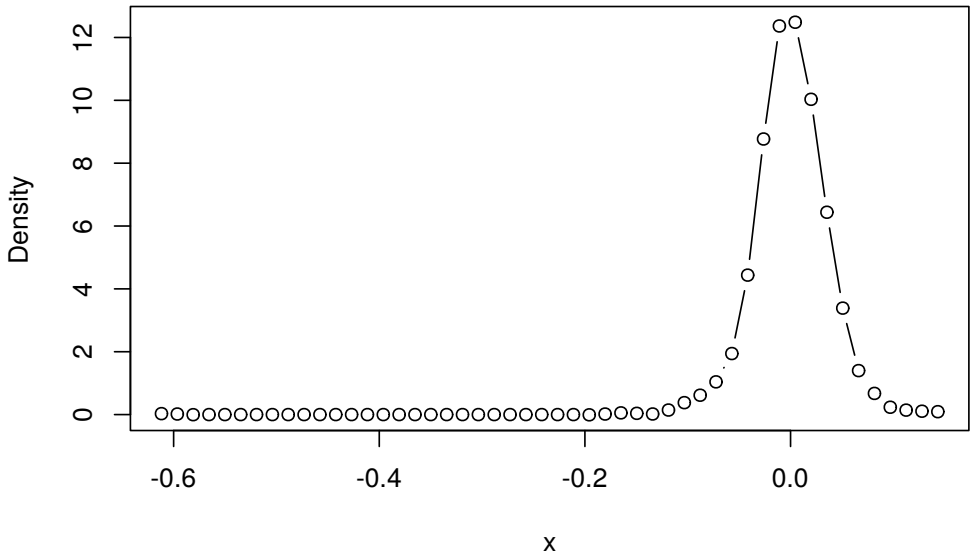

```
dens.np.df <- data.frame(x = dens.np$eval[, 1], y = dens.np$dens)

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



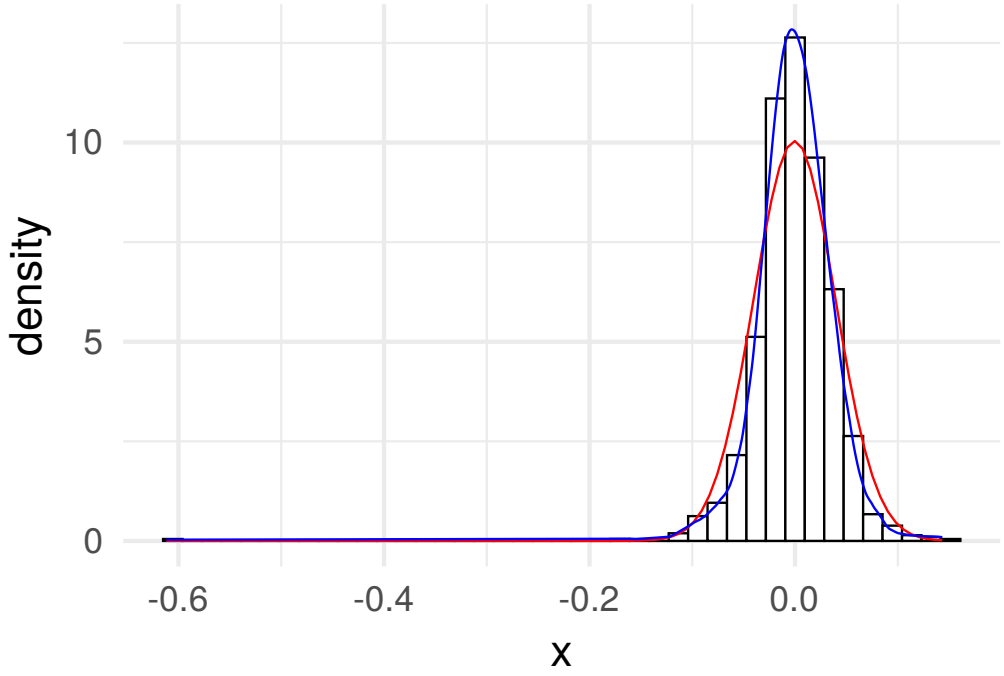

```
h_cv_np_ml <- npudensbw(dat = x, bwmethod = "cv.ml",
    ckertype = "epanechnikov", ckerorder = 4)
```

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

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

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




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

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




```
fani <- tibble()

for (b in seq(0.001, 0.05, length.out = 40)) {
    f <- npudens(tdat = x, ckertype = "epanechnikov",
        bandwidth.compute = FALSE, bws = b)
    fani <- fani %>% bind_rows(tibble(xreal = sort(x),
        x = f$eval$x, y = f$dens, bw = b))
}

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

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

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

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

2.6. Ejercicios

Del libro de (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

Jacknife y Bootstrap

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

Primero se conoce que

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

y esto nos permite escribir el intervalo de confianza como

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

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

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

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

¿Cómo podemos encontrar ese intervalo de confianza?

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

3.1. Caso concreto

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

Cargamos la base de la siguiente forma:

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

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

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

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

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

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

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

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

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

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

3.2. Jacknife

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

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

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

para algún a and b.

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

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

por lo tanto

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

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

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

Es claro que en este contexto, se tiene que

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

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

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

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

x
429.7098
428.1905
429.6023
429.3756
430.1087
428.1905
429.6023
429.3756
430.0764
429.7098

Definamos el sesgo *jackife* como

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

donde

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

Ejercicio 3.2. En nuestro caso tendríamos lo siguiente:

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

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

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

Observe que b_{jack} tiene la siguiente propiedad

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

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

Podemos usar los T_i para generar muestras adicionales para estimar el parámetro θ.

En este caso defina el siguiente estimador:

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

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

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

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

¿Qué conclusión se obtiene de este cálculo?

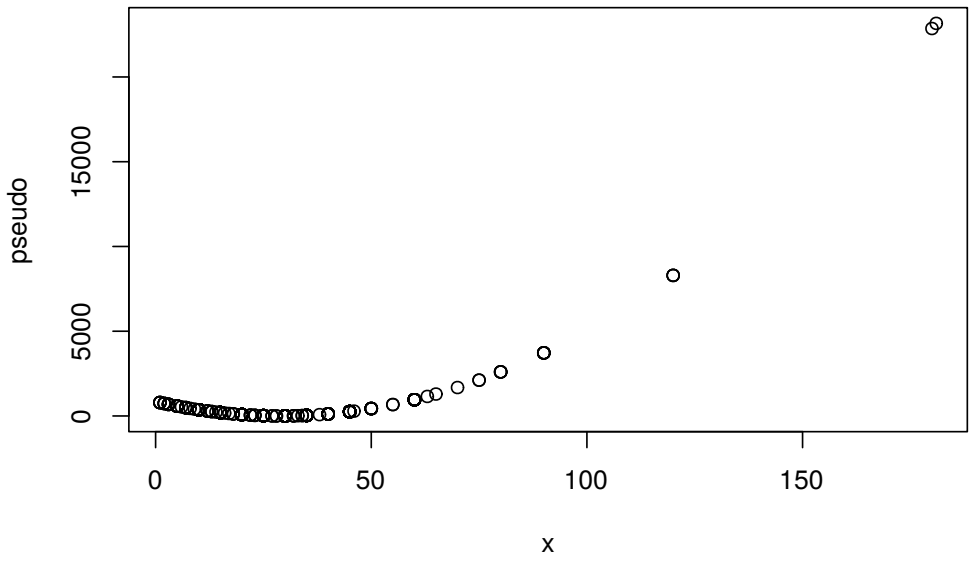
Ejercicio 3.4. Los pseudo-valores se estiman de forma directa como,

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

pseudo[1:10]
```

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

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

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



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

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

donde

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

Nota: . Sin embargo, se puede demostrar fácilmente que se pueden usar pseudovalores para construir una prueba normal de hipótesis. Dado que cada pseudovalor es independiente e idénticamente distribuido (iid), se deduce que su promedio se ajusta a una distribución normal a medida que el tamaño de la muestra aumenta. El promedio de los pseudovalores es solo T_{jack} y el valor esperado de ese promedio, debido a la construcción a la imparcialidad del estimador, es el parámetro bajo investigación, θ. Por lo tanto, tenemos que

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

Ejercicio 3.5.

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

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

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

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

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

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

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

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

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

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

3.3. Bootstrap

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

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

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

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

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

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

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

En simple pasos la técnica es

1. Seleccione $X_1^*, \ldots, X_n^* \sim \widehat{F}_n$
2. Estime $T_n^* = g(X_1^*, \ldots, X_n^*)$
3. Repita los Pasos 1 y 2, B veces para obtener $T_{n,1}^*, \ldots, T_{n,B}^*$
4. Estime
$$v_{\mathrm{boot}} = \frac{1}{B}\sum_{b=1}^{B}\left(T_{n,b}^* - \frac{1}{B}\sum_{r=1}^{B} T_{n,r}^*\right)^2$$

Por la ley de los grandes números tenemos que

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

además llamaremos,

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

En pocas palabras lo que tenemos es que

$$\begin{aligned} \text{Mundo Real: } F &\implies X_1, \ldots, X_n \implies T_n = g(X_1, \ldots, X_n) \\ \text{Mundo Bootstrap: } \widehat{F}_n &\implies X_1^*, \ldots, X_n^* \implies T_n^* = g(X_1^*, \ldots, X_n^*) \end{aligned}$$

En términos de convergencia lo que se tiene es que

$$\text{Var}_F(T_n) \overbrace{\approx}^{O(1/\sqrt{n})} \text{Var}_{\widehat{F}_n}(T_n) \overbrace{\approx}^{O(1/\sqrt{B})} v_{boot}$$

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

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

Por lo tanto, todos los puntos originales $X_1, \ldots, X_n$ 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

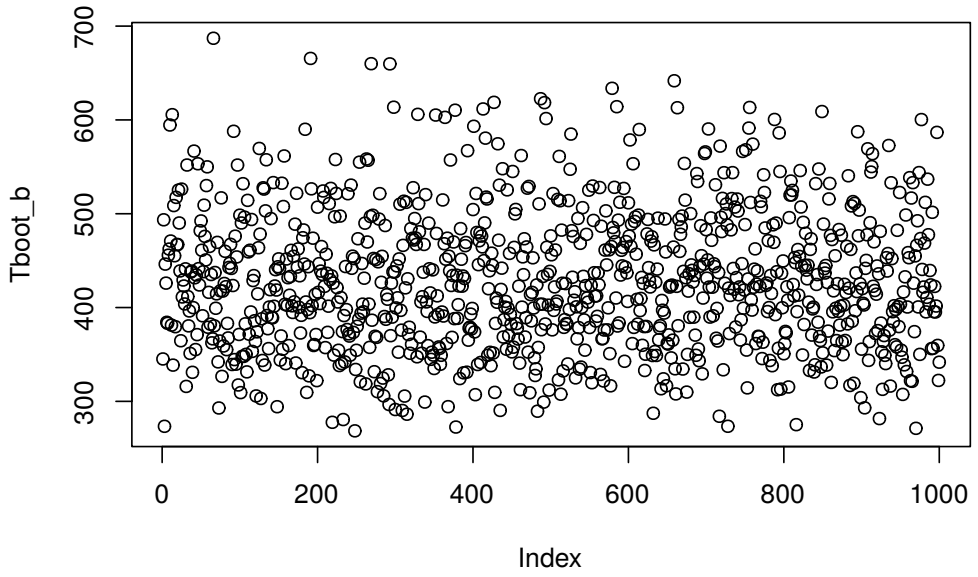
1. Seleccione una muestra con remplazo $X_1^*, \ldots, X_n^*$ de $X_1, \ldots, X_n$.

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

```
B <- 1000
Tboot_b <- NULL

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

Tboot_b[1:10]
```

```
##  [1] 345.1819 493.5279 273.3998 446.3071 426.0340 384.2662 383.2132 455.813
##  [9] 462.3363 594.5774
```

```
plot(Tboot_b)
```



Por supuesto podemos encontrar los estadísticos usuales para esta nueva muestra

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

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

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

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

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

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

3.3.1. Intervalos de confianza

3.3.1.1. Intervalo Normal

Este es el más sencillo y se escribe como

$$T_n \pm z_{\alpha/2}\widehat{\mathrm{Se}}_{\mathrm{boot}} \tag{3.4}$$

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

Ejercicio 3.7. 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 $\widehat{\theta}_n = T\left(\widehat{F}_n\right)$ y defina la cantidad pivotal $R_n = \widehat{\theta}_n - \theta$.
Sea $H(r)$ la función de distribución del pivote:

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

Además considere $C_n^\star = (a, b)$ 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

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

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

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.

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

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

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

Nota: . Según la notación anterior note que

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

Con estas observaciones It follows that an approximate $1 - \alpha$ confidence interval is $C_n = (\widehat{a}, \widehat{b})$ where

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

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

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

Ejercicio 3.8. 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
```

3.3.2. Intervalo pivotal studentizado

Una mejora del intervalo anterior sería normalizar los estimadores previamente

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

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: . Esto requerira estimar la varianza de $T^*_{n,b}$ para cada b.

Con esto se puede obtener cantidades $Z^*_{n,1}, \ldots, Z^*_{n,B}$ que debería ser próximos a Z_n.

Sea z^*_α del α cuantíl de $Z^*_{n,1}, \ldots, Z^*_{n,B}$, entonces $\mathbb{P}(Z_n \leq z^*_\alpha) \approx \alpha$.

Define el intervalo

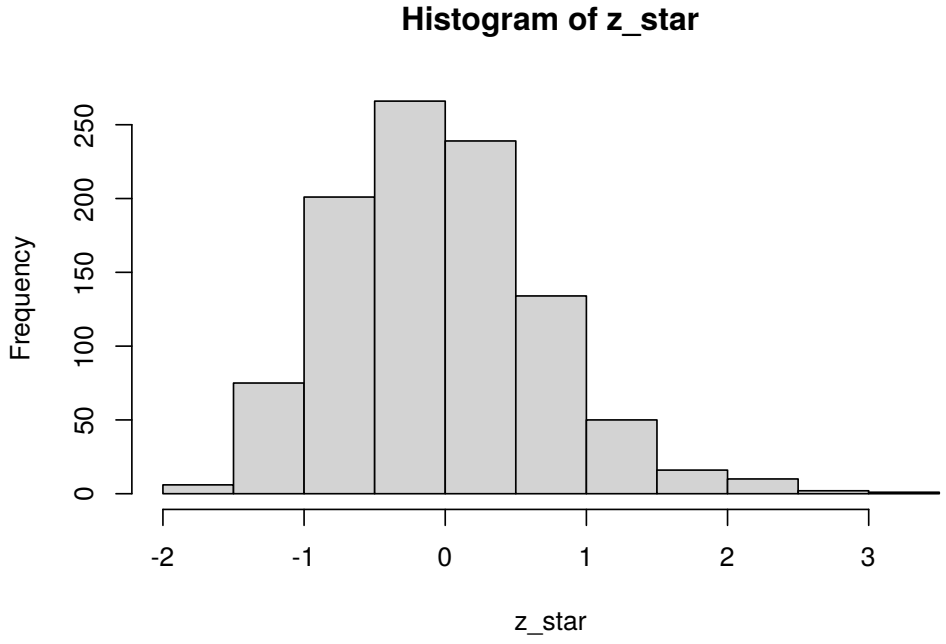
$$C_n = \left(T_n - z^*_{1-\alpha/2}\widehat{\text{se}}_{\text{boot}}, T_n - z^*_{\alpha/2}\widehat{\text{se}}_{\text{boot}}\right)$$

Justificado por el siguiente cálculo:

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

Ejercicio 3.9. Note que para este caso tenemos que hacer bootstrap para cada estimador bootstrap calculado.

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

for (b in 1:B) {
    xb <- sample(x, size = n, replace = TRUE)
    Tboot_b[b] <- var(xb)
    for (m in 1:B) {
        xbm <- sample(xb, size = n, replace = TRUE)
        Tboot_bm[b] <- var(xbm)
    }
    sdboot_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%
## 309.8684 521.6372
```

3.3.3. Resumiendo

Resumiendo todos lo métodos de cálculo de intervalos obtenemos

```
knitr::kable(data.frame(Metodo = c("Jacknife", "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
Jacknife	285.1679	573.3289
Bootstrap Normal	283.8315	574.6653
Bootstrap Pivotal	271.2827	551.4989
Bootstrap Pivotal Estudentizado	309.8684	521.6372

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 Jacknife y Bootstrap (con todos sus intervalos de confianza). Dada que la distancia es una medida que puede ser influenciada por distancias muy cortas o muy largas, se puede calcular el logaritmo de esta variable para eliminar la escala de la distancias.

2. Verifique que esta última variable se podría estimar paramétricamente con una distribución normal. Repita los cálculos anteriores tomando como cuantiles los de una normal con media 0 y varianza 1.

3. Compare los intervalos calculados y comente los resultados.

4. Del libro (Wasserman 2006) **Sección 3:** 2, 3, 7, 9, 11.

Capítulo 4

Estimación de densidades con Bayes

4.1. Introducción a la estimación Bayesiana

4.1.1. Preliminares

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

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

Además para el caso de la binomial tenemos que

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

y la distribución beta se escribe de la forma

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

donde

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

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

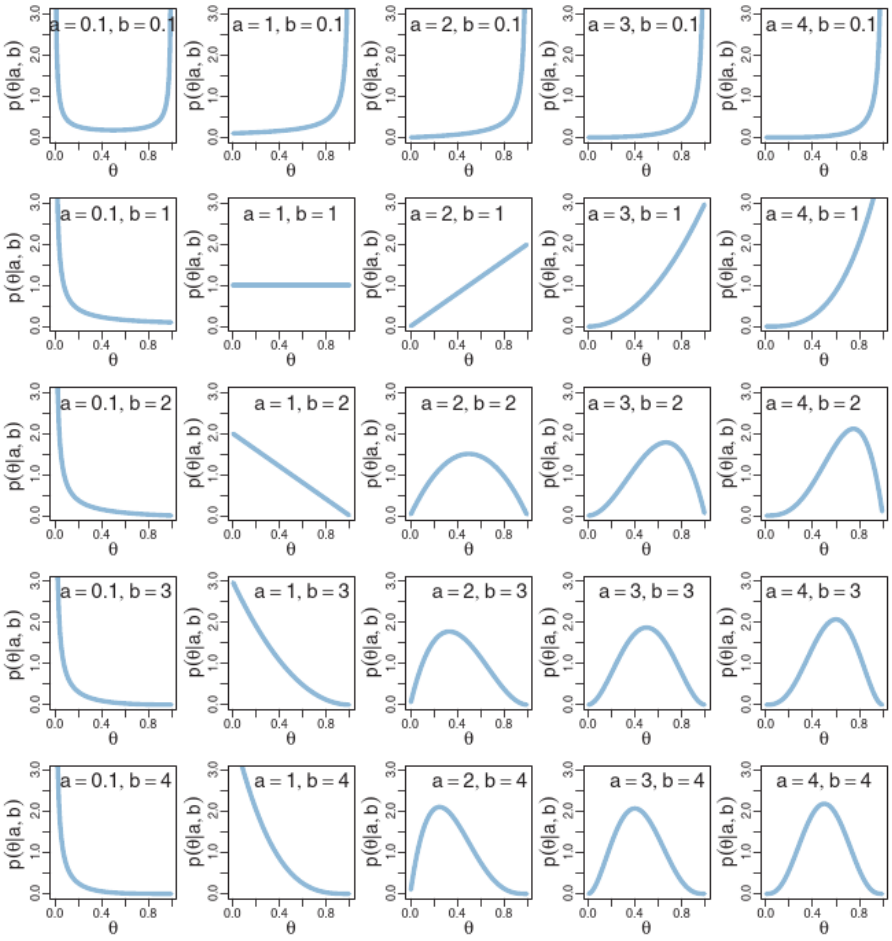


Figura 4.1: Tomado de Kruschke (2014)

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

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

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

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

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

4.1.2. Ejemplo sencillo

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

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

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

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

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

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

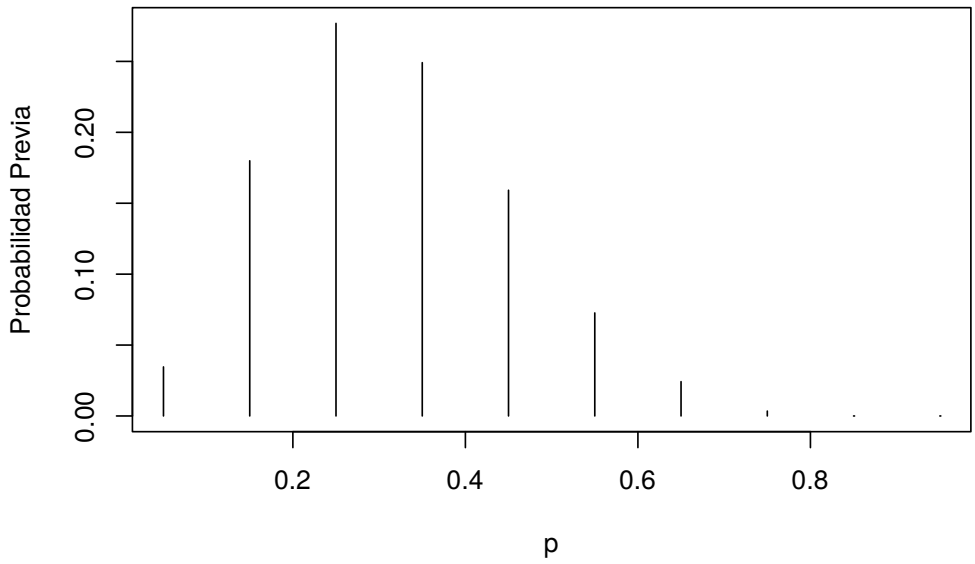
```
(prior <- c(1, 5.2, 8, 7.2, 4.6, 2.1, 0.7, 0.1, 0,
    0))
```

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

```
(prior <- prior/sum(prior))
```

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

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



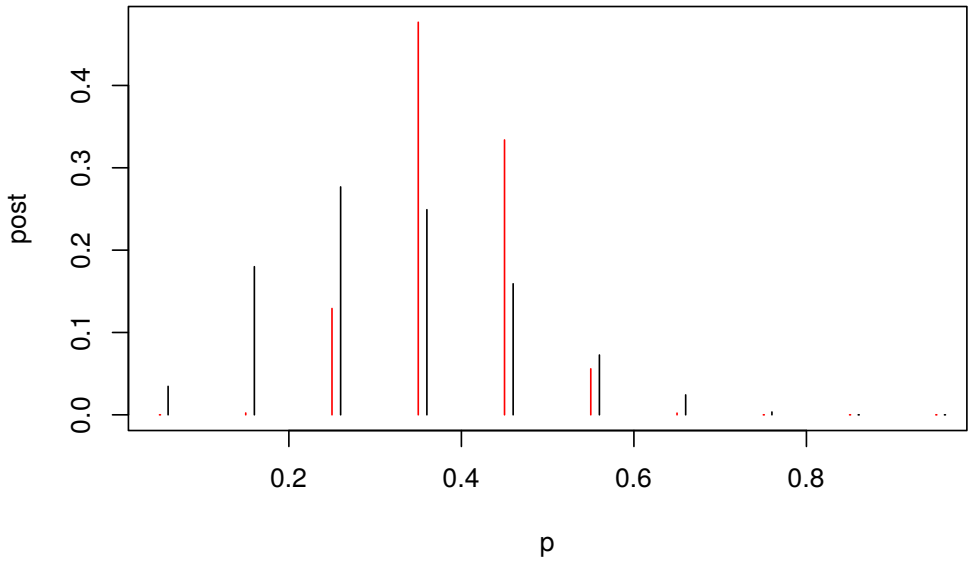
El paquete `LearnBayes` tiene la función `pdisc` que estima la distribución posterior para una previa discreta binomial. Recuerde que el valor 11 representa la cantidad de estudiantes con más de 8 horas de sueño y 16 lo que no duermen esa cantidad.

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

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

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

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



¿Qué se puede deducir de estos resultados?

4.1.3. Datos reales

Continuemos el ejercicio pero esta vez usando datos reales.

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

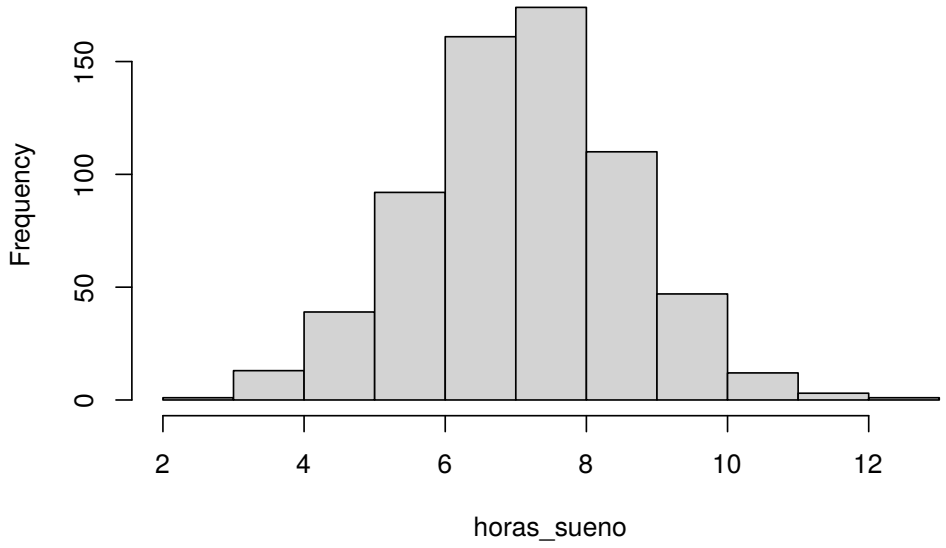
```
data("studentdata")
```

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

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

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

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



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

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

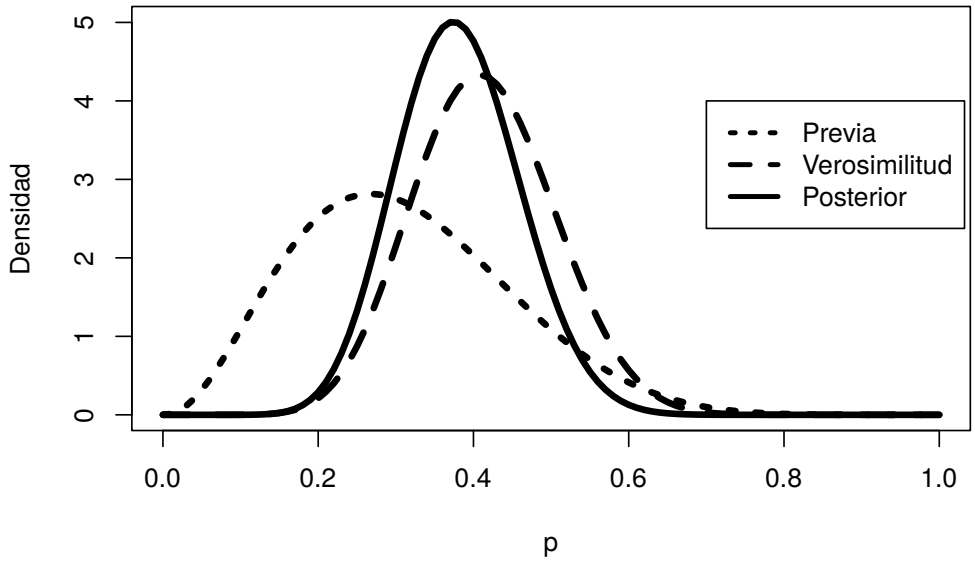
El ajuste de los parámetros de la Beta depende mucho de la información previa que se tenga del modelo. Una forma fácil de estimarlo es a través de cuantiles con los cuales se puede reescribir estos parámetros. En particular, suponga que se cree que el 50 % de las observaciones la proporción será menor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

Para esto ajustaremos los siguientes parámetros

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

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

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

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



En particular, si estamos interesados en $\mathbb{P}(p >= ,5|$ data $)$ se puede estimar con

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

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

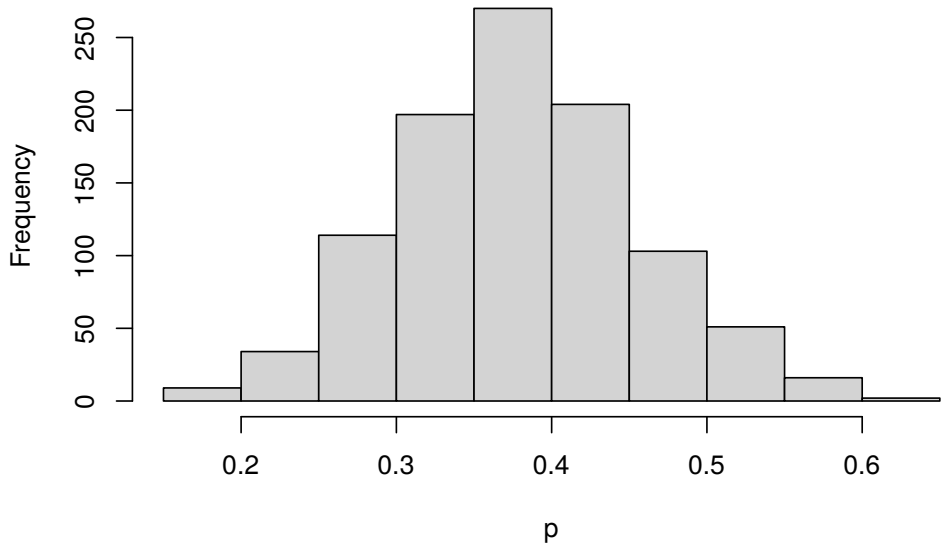
y el intervalo de confianza correspondiente a esta distribución sería

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

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

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

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



La probabilidad que este valor sea mayor que 0.5 es

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

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

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

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

4.2. Previa de histograma

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

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

El método que usaremos será el siguiente:

- Elija una cuadrícula de valores de p sobre un intervalo que cubra la densidad posterior.
- Calcule el producto de la probabilidad $L(p)$ y el $f(p)$ sobre esa grilla.
- Normalice dividiendo cada producto por la suma de los productos. En esto paso, estamos aproximando la densidad posterior por una probabilidad discreta Distribución en la grilla.
- Usando el comando `sample` de `R`, tome una muestra aleatoria con reemplazo de la distribución discreta.

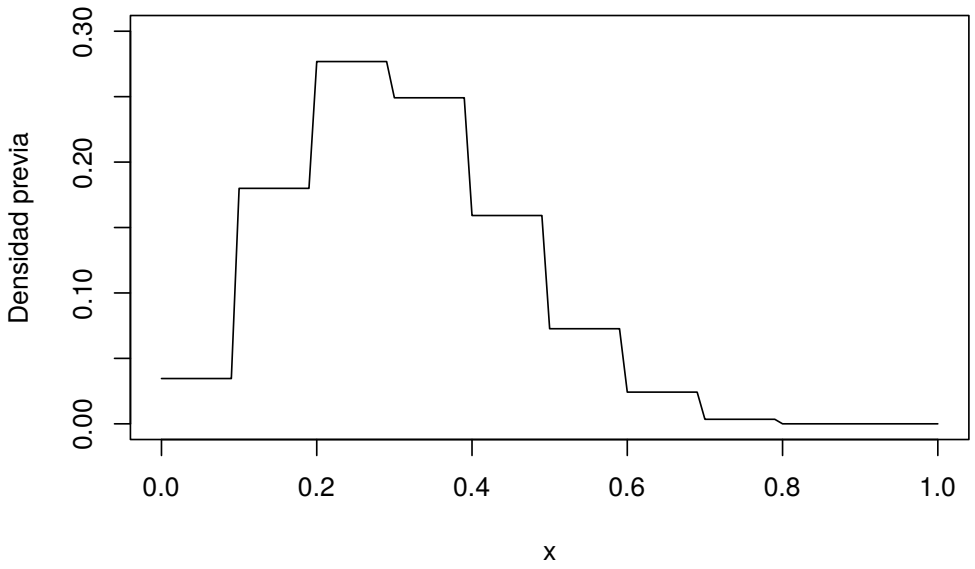
El resultado nos debe arrojar una muestra de la distribución posterior sobre la grilla

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

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

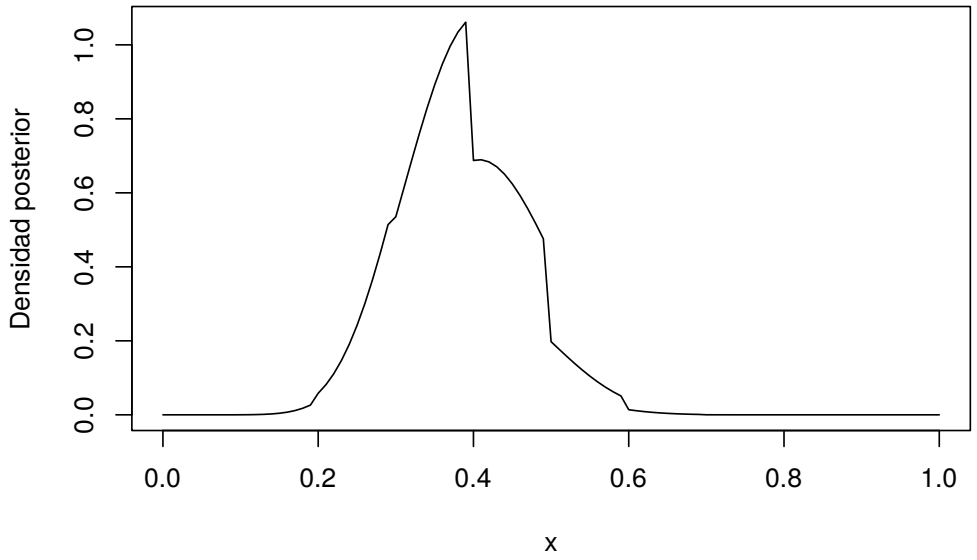
Con la función `histprior` construye los valores de p sobre una grilla.

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



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

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

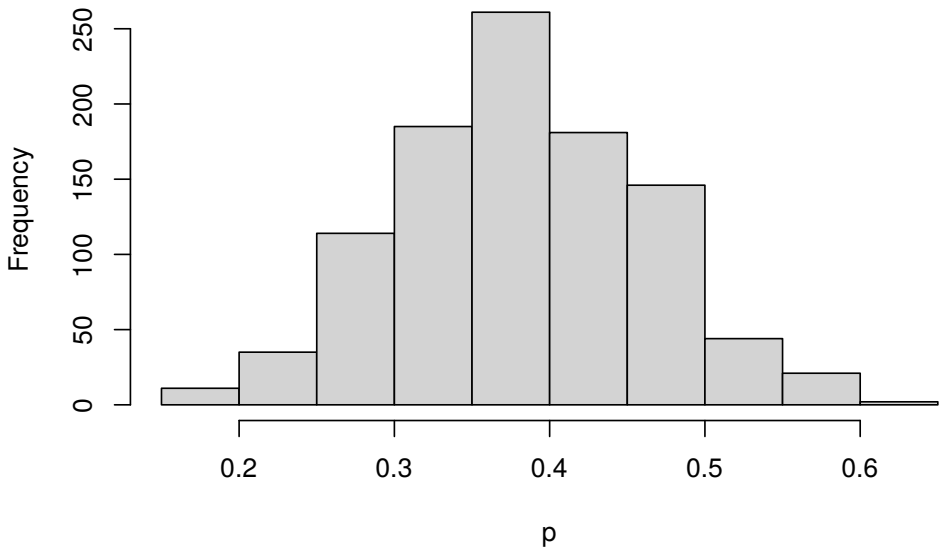


Para conseguir la distribución posterior, solo debemos de construirla para una secuencia ordenada de valores p

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

Finalmente basta con tomar el muestreo de la posterior

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



4.3. Métodos Monte Carlo

4.4. Una moneda

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

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

4.4.1. Ejemplo del viajero

Suponga que tenemos un viajero que quiere estar en 7 lugares distintos (suponga que están en línea recta) y la probabilidad de pasar a un lugar a otro

se decide tirando una moneda no sesgada (50 % a la derecha y 50 % a la izquierda).

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

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

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

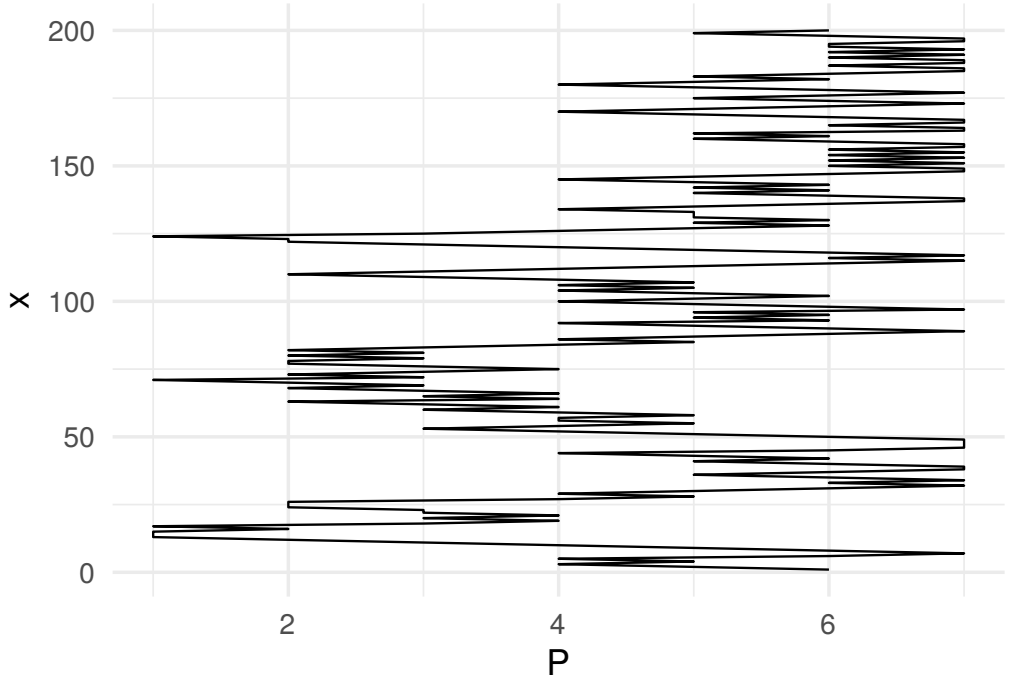
       Tira un número aleatorio entre 0 y 1 r

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

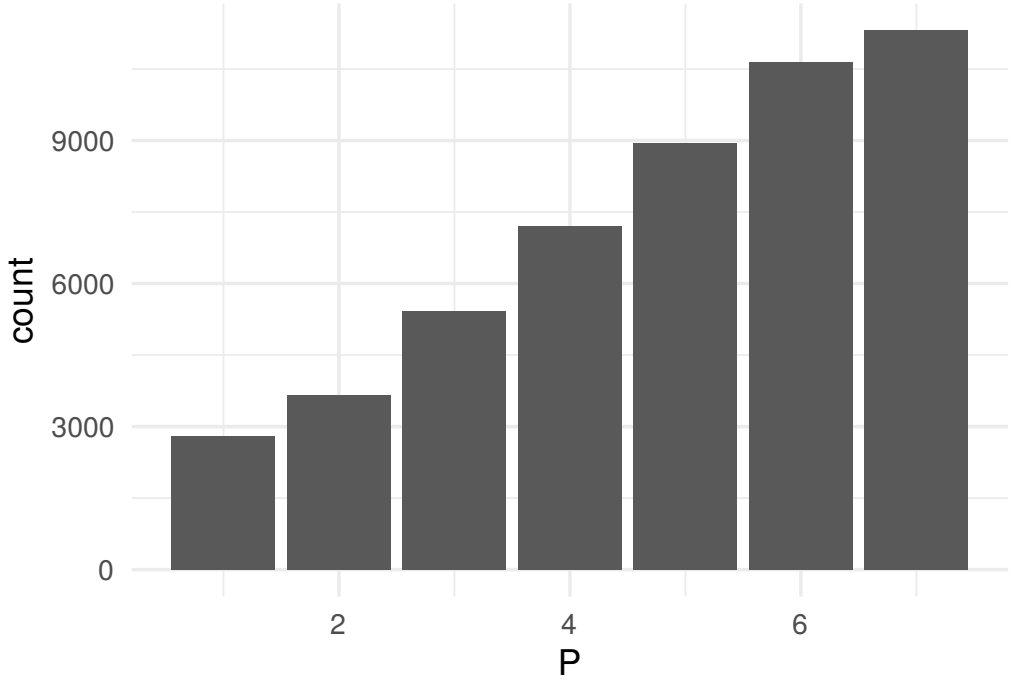
```
P <- 1:7

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

n_pasos <- 50000
trayectoria <- numeric(n_pasos)

trayectoria[1] <- pos_actual

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

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

```
    } else if (moneda == 0 & (pos_actual - 1) >= 1) {
        pos_nueva <- pos_actual - 1
    }

    p_moverse <- min(pos_nueva/pos_actual, 1)

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

    if (hay_movimiento) {
        pos_actual <- pos_nueva
    }

    trayectoria[k] <- pos_nueva
}
```

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

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




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




```
mean(trayectoria)
```

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

```
sd(trayectoria)
```

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

4.4.2. Cadenas de Markov

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

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

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

Entonces

- $\mathbb{P}(\theta \to \theta+1) = 0{,}5\,\text{mín}\left(\frac{P(\theta+1)}{P(\theta)}/, 1\right)$
- $\mathbb{P}(\theta+1 \to \theta) = 0{,}5\,\text{mín}\left(\frac{P(\theta)}{P(\theta+1)}/, 1\right)$

Entonces la razón entre estas dos probabilidades es

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

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

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

La matriz T tiene las propiedades

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

4.4.3. El algoritmo de Metropolis-Hasting

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 $f(\theta|$ datos $)$.

En palabras simples el algoritmo de Metropoli Hasting es

1. Simule un valor θ^* de una densidad de propuesta $p\left(\theta^*|\theta^{t-1}\right)$
2. Estime la razón
$$R = \frac{f\left(\theta^*\right) L\left(\theta^{t-1}|\theta^*\right)}{f\left(\theta^{t-1}\right) L\left(\theta^*|\theta^{t-1}\right)}$$
3. Estima la probabilidad de aceptación $p_{\text{moverse}} = \text{mín}\{R, 1\}$.
4. Tome θ^t tal que $\theta^t = \theta^*$ con probabilidad p_{moverse} ; en otro caso $\theta^t = \theta^{t-1}$

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

Si esta es independiente de las elecciones anteriores entonces,

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

Otras formas es escoger

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

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

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

Una última opción es tomar

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

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

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

$$\mathbb{P}\left(\theta_\star|\theta^{(t)}\right) = L\left(\theta_\star|\theta^{(t)}\right) \cdot \text{mín}\left\{1, \frac{f\left(\theta_\star\right) L\left(\theta^{(t)}|\theta_\star\right)}{f\left(\theta^{(t)}\right) L\left(\theta_\star|\theta^{(t)}\right)}\right\} \tag{4.1}$$

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

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

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

$$f\left(\theta_j\right) = \underbrace{\sum_{i=1}^{M} f\left(\theta_i\right) \mathbb{P}\left(\theta_j|\theta_i\right)}_{\text{Probabilidad total}} = \sum_{i=1}^{M} f\left(\theta_j\right) \mathbb{P}\left(\theta_i|\theta_j\right)$$

$$f(\boldsymbol{\theta})^\top T = f(\boldsymbol{\theta}) \tag{4.3}$$

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

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

4.4.5. Extensión al caso del viajero

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

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

Para hacer este ejemplo concreto, el viajero no conoce cuál es su probabilidad de salto θ pero sabe que ha tirado la moneda N veces y observado z exitos. 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}

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

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

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

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

```r
# 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 49.4 y tasa de rechazo del 50.59

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

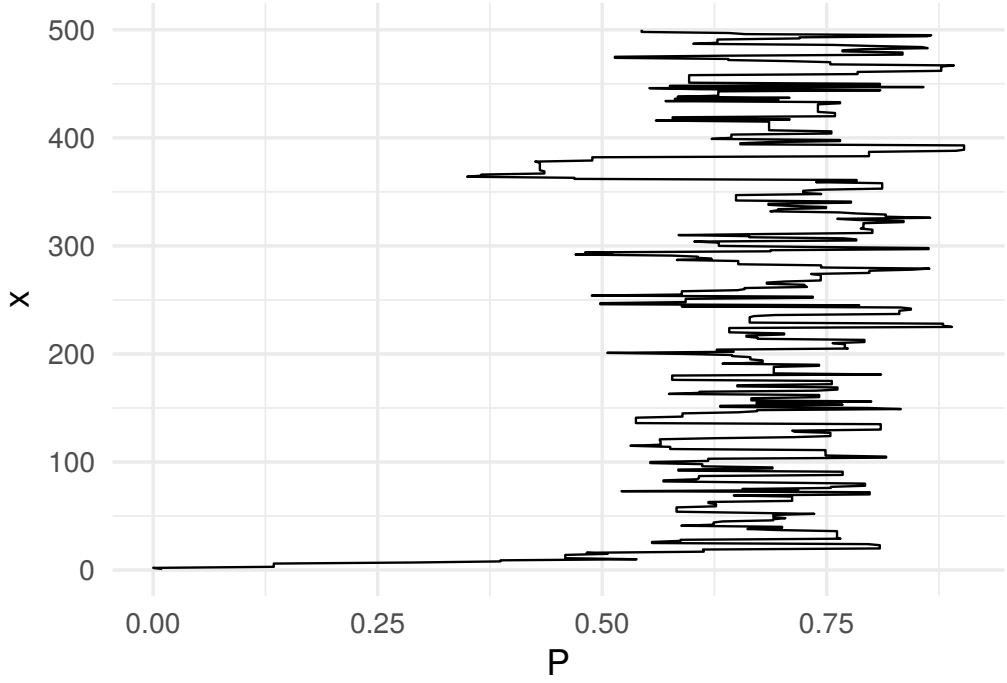
```
mean(trayectoria[500:n_pasos])
```

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

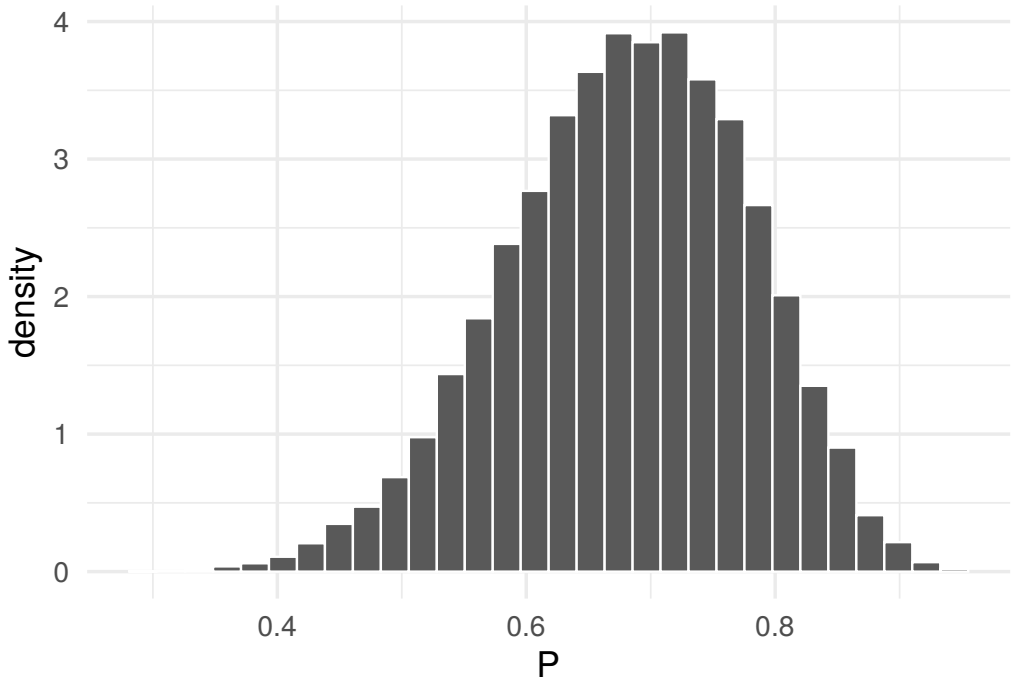
```
sd(trayectoria[500:n_pasos])
```

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

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

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




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



4.5. Dos monedas

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

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

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

Tenemos:

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

3. Verosimilitud: Bernoulli.

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

4. Previa: Beta independiente para cada θ.

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

La distribución posterior se puede escribir como

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

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

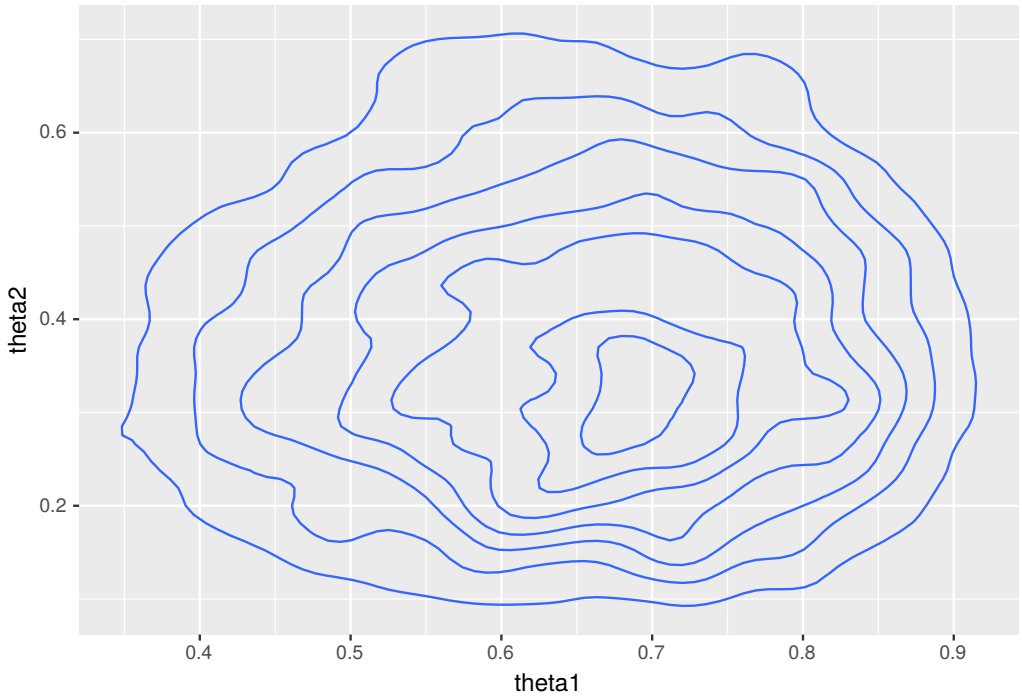
Tratemos de encontrar los parámetros para la distribución posterior usando un algoritmo de Metropolis-Hasting. Función tomada de Kruschke-Notes

```
metro_2coins <- function(
  z1, n1,              # z = successes, n = trials
  z2, n2,              # z = successes, n = trials
  size  = c(0.1, 0.1), # sds of jump distribution
  start = c(0.5, 0.5), # value of thetas to start at
  num_steps = 5e4,     # number of steps to run the algorithm
  prior1 = dbeta,      # function describing prior
  prior2 = dbeta,      # function describing prior
  args1 = list(),      # additional args for prior1
  args2 = list()      # additional args for prior2
  ) {

  theta1           <- rep(NA, num_steps)  # trick to pre-alocate memory
  theta2           <- rep(NA, num_steps)  # trick to pre-alocate memory
  proposed_theta1  <- rep(NA, num_steps)  # trick to pre-alocate memory
```

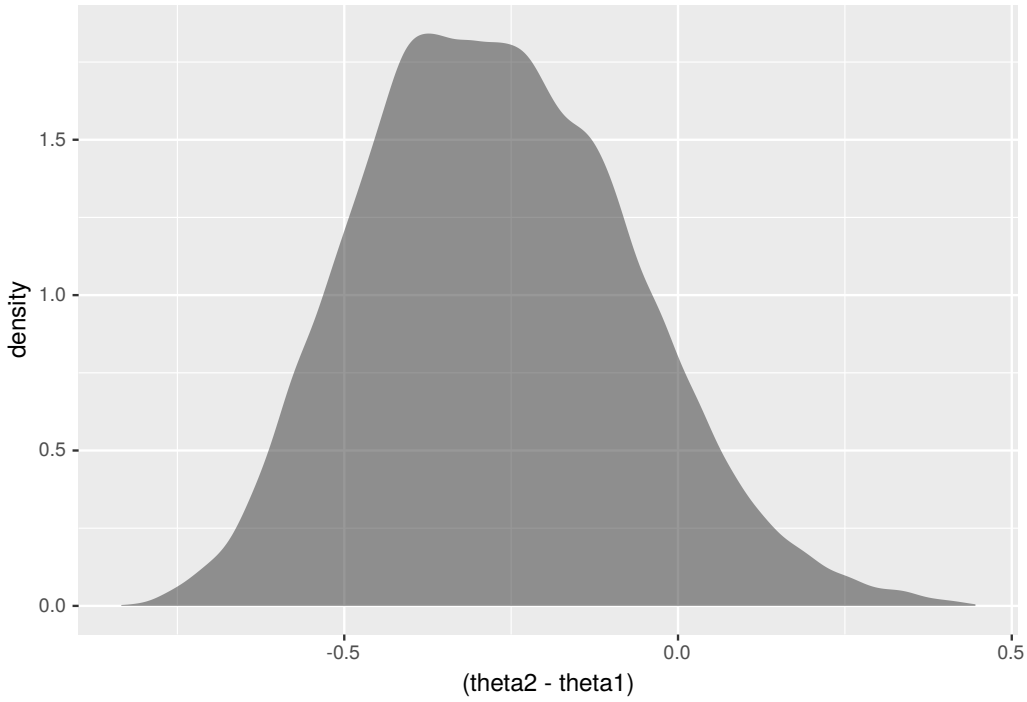
```
proposed_theta2   <- rep(NA, num_steps)  # trick to pre-alocate memory
move              <- rep(NA, num_steps)  # trick to pre-alocate memory
theta1[1]         <- start[1]
theta2[1]         <- start[2]

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
  )
}
```

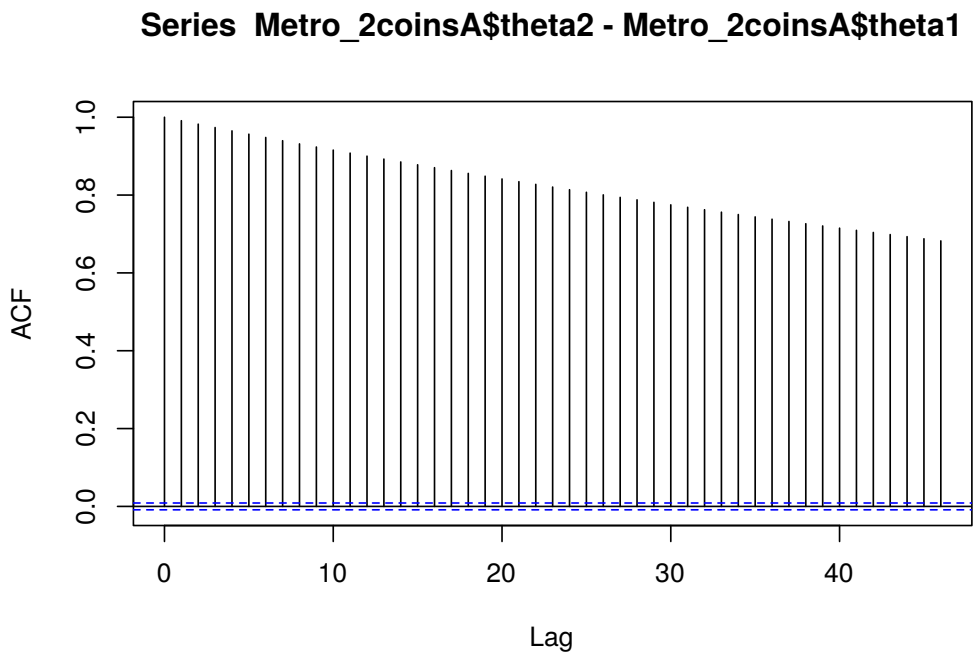
```
Metro_2coinsA <- metro_2coins(z1 = 6, n1 = 8, z2 = 2,
    n2 = 7, size = c(0.02, 0.02), args1 = list(shape1 = 2,
        shape2 = 2), args2 = list(shape1 = 2, shape2 = 2))

Metro_2coinsA %>% gf_density2d(theta2 ~ theta1)
```



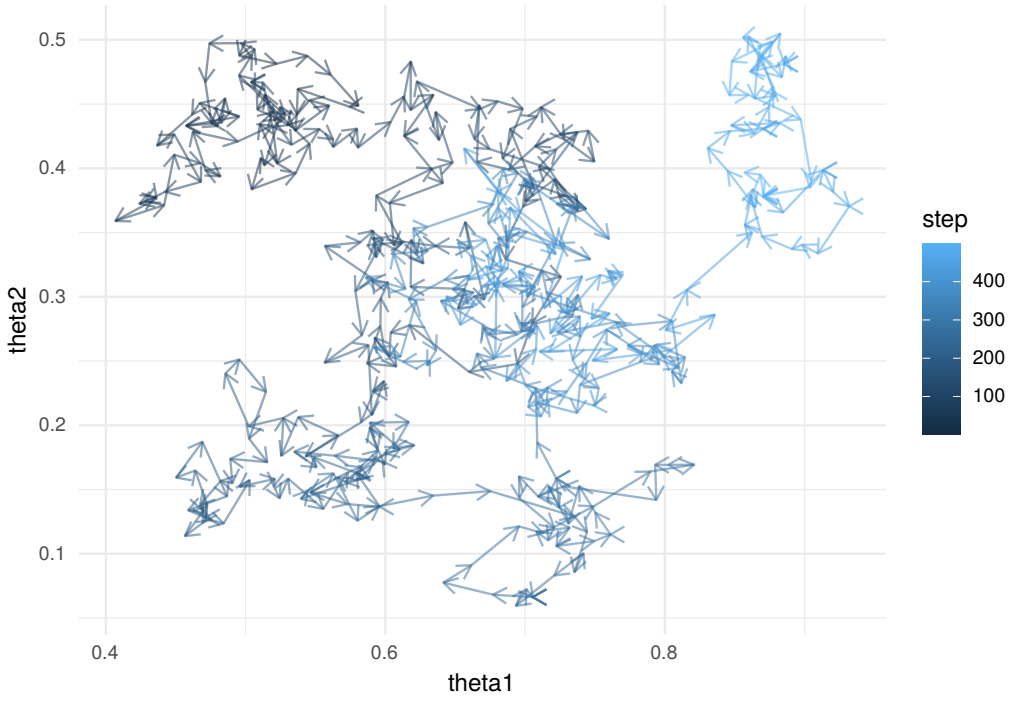

```
Metro_2coinsA %>% gf_density(~(theta2 - theta1))
```




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

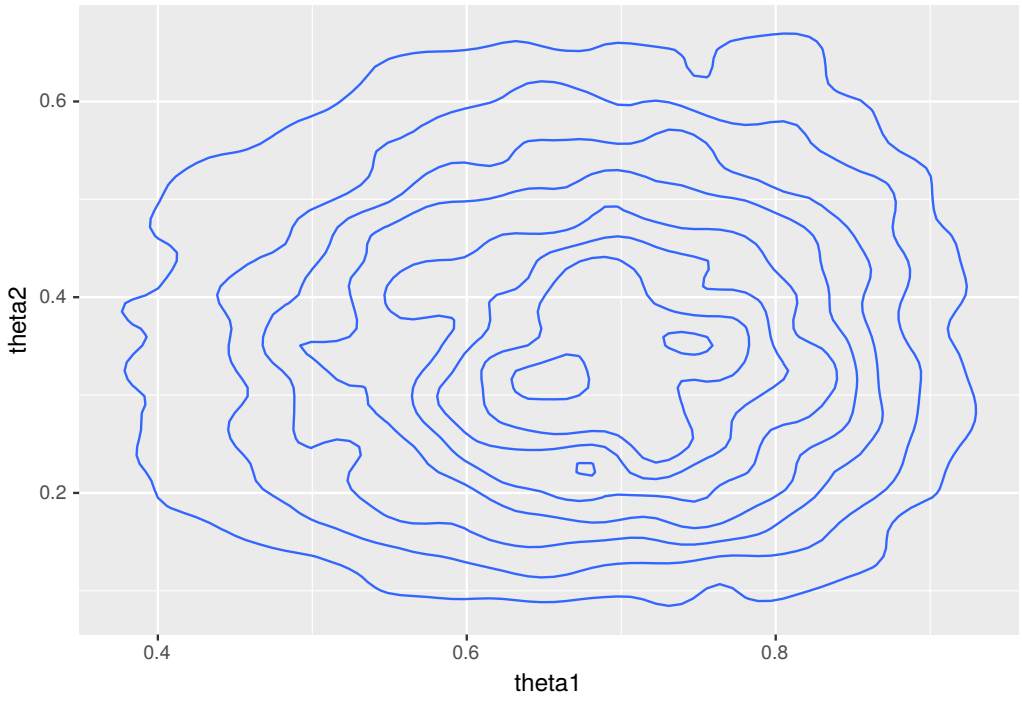



```
Metro_2coinsA %>% filter(step < 500) %>% gf_path(theta2 ~
    theta1, color = ~step, alpha = 0.5, arrow = arrow(type = "open",
    angle = 30, length = unit(0.1, "inches"))) + theme_minimal()
```

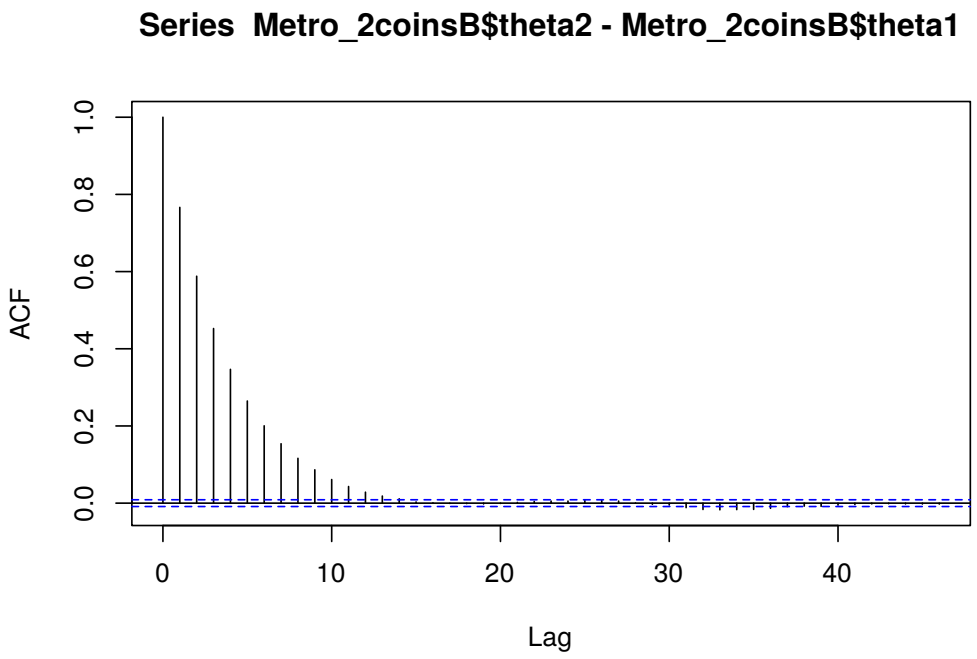



```
library(gganimate)
Metro_2coinsAplot <- Metro_2coinsA %>% filter(step <
    500) %>% gf_path(theta2 ~ theta1, color = ~step,
    alpha = 0.5, arrow = arrow(type = "open", angle = 30)) +
    theme_minimal() + transition_reveal(step)

animate(Metro_2coinsAplot, fps = 1)
```

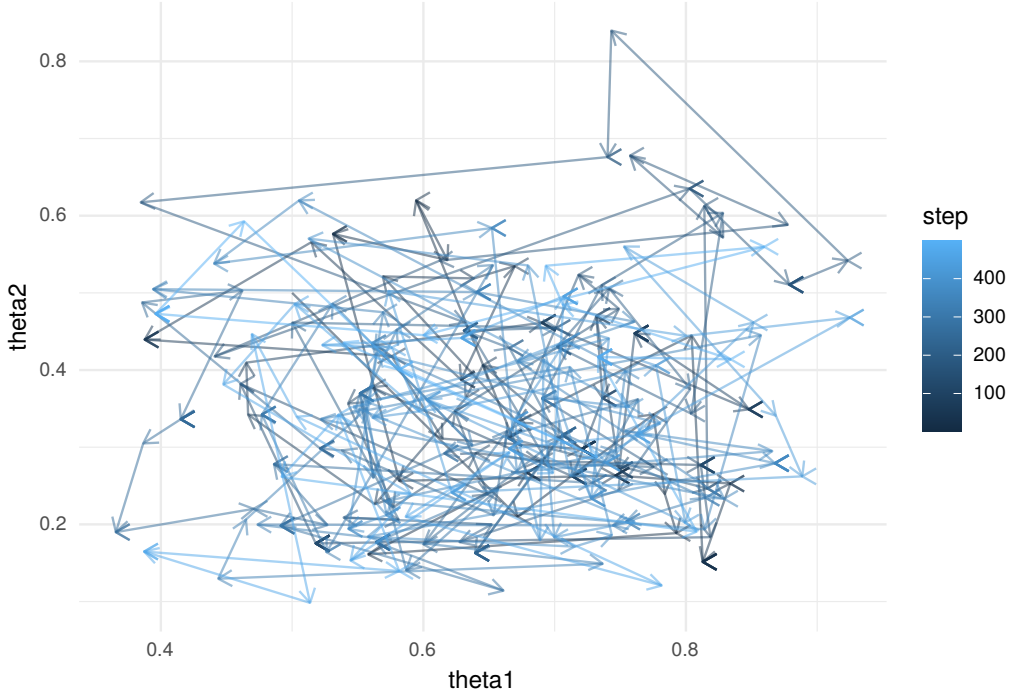
```
Metro_2coinsB <- metro_2coins(z1 = 6, n1 = 8, z2 = 2,
    n2 = 7, size = c(0.2, 0.2), args1 = list(shape1 = 2,
        shape2 = 2), args2 = list(shape1 = 2, shape2 = 2))

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)
```

4.5.1. Muestreo de Gibbs

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

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

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 cuanta los datos y fijando los otros parámetros. Es decir,

$$[\theta_1|\theta_2,\ldots,\theta_M,datos]$$
$$[\theta_2|\theta_1,\theta_3,\ldots,\theta_M,datos]$$
$$[\theta_M|\theta_1,\ldots,\theta_{M-1},datos]$$

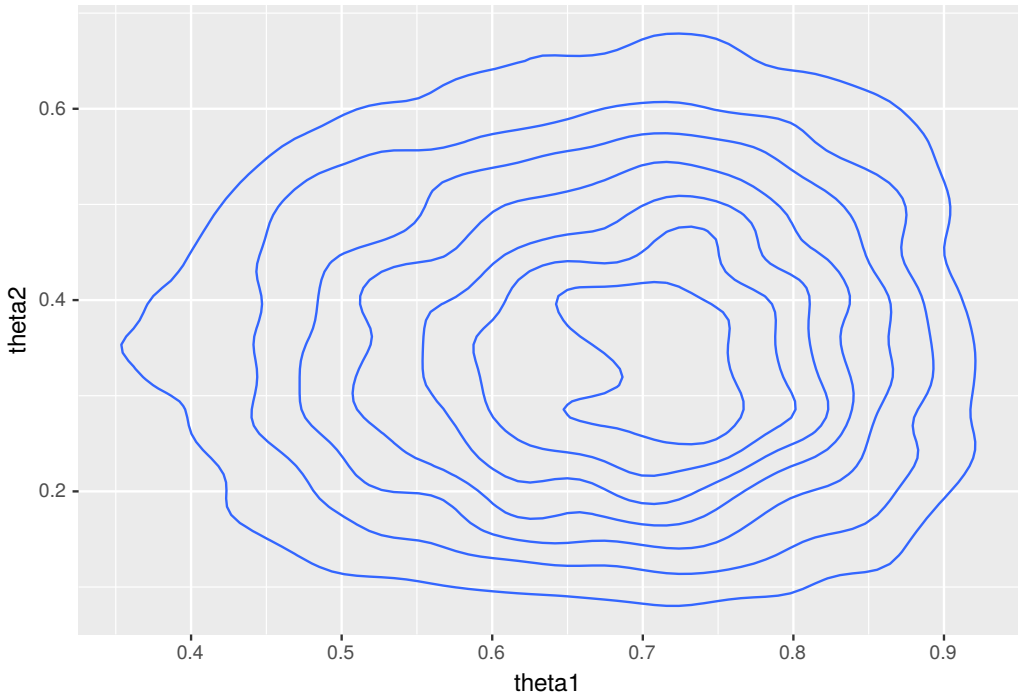
El tratamiento teórico puede ser consultado https://www.ece.iastate.edu/~namrata/EE527_Spring08/l4c.pdf#page=16

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

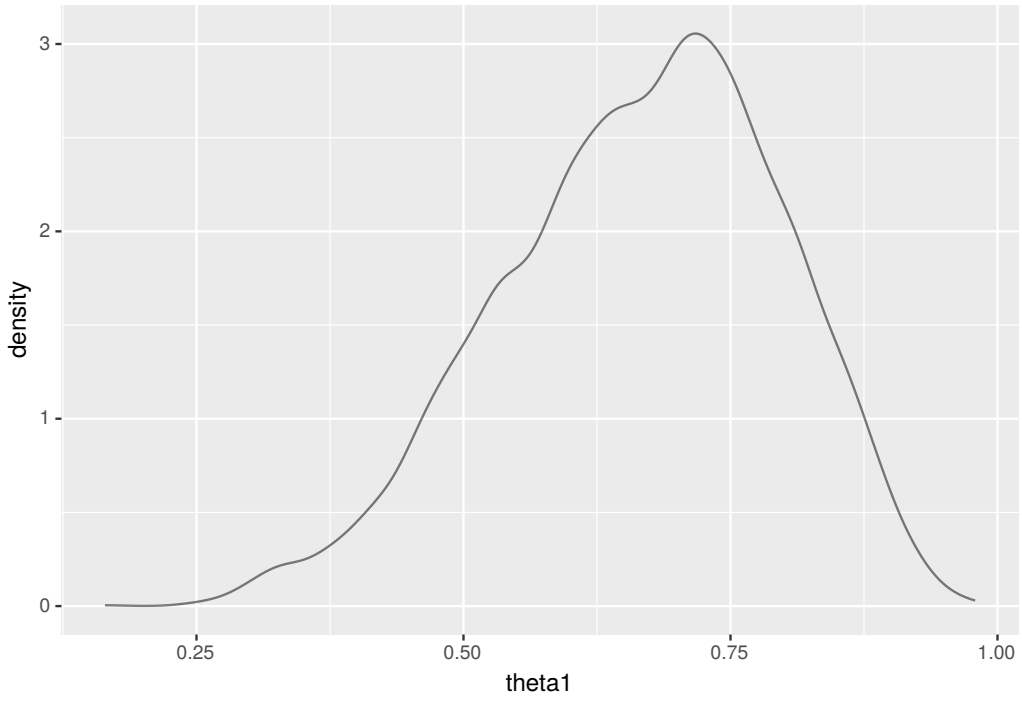
```
gibbs_2coins <- function(
  z1, n1,              # z = successes, n = trials
  z2, n2,              # z = successes, n = trials
  start = c(0.5, 0.5), # value of thetas to start at
  num_steps = 1e4,     # number of steps to run the algorithm
  a1, b1,              # params for prior for theta1
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

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

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

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

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

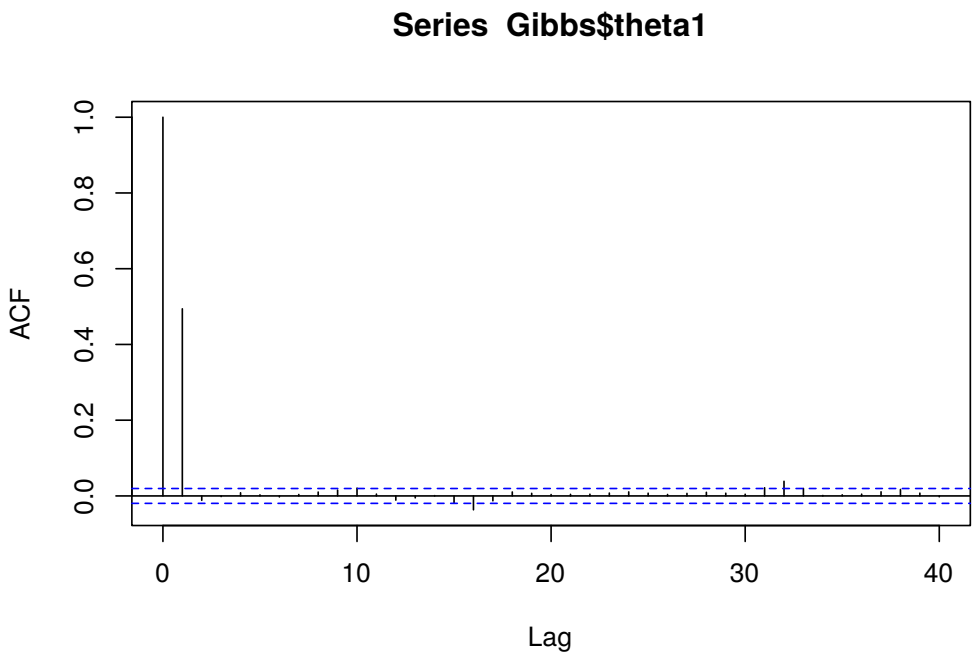
```
Gibbs <- gibbs_2coins(z1 = 6, n1 = 8, z2 = 2, n2 = 7,
    a1 = 2, b1 = 2, a2 = 2, b2 = 2)

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



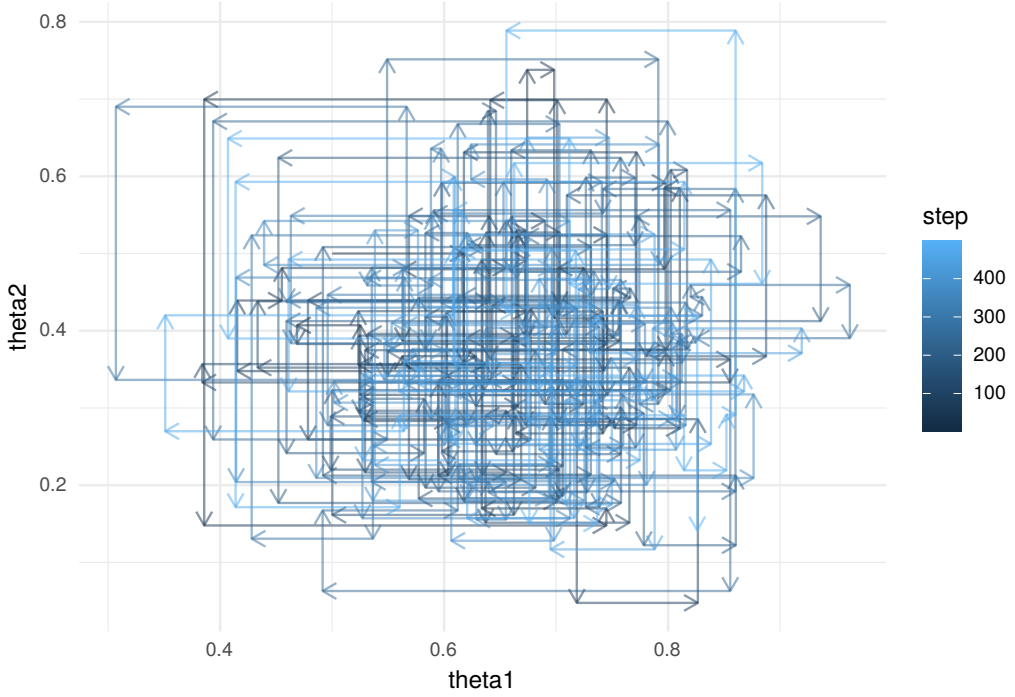

```
Gibbs %>% gf_dens(~theta1)
```




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



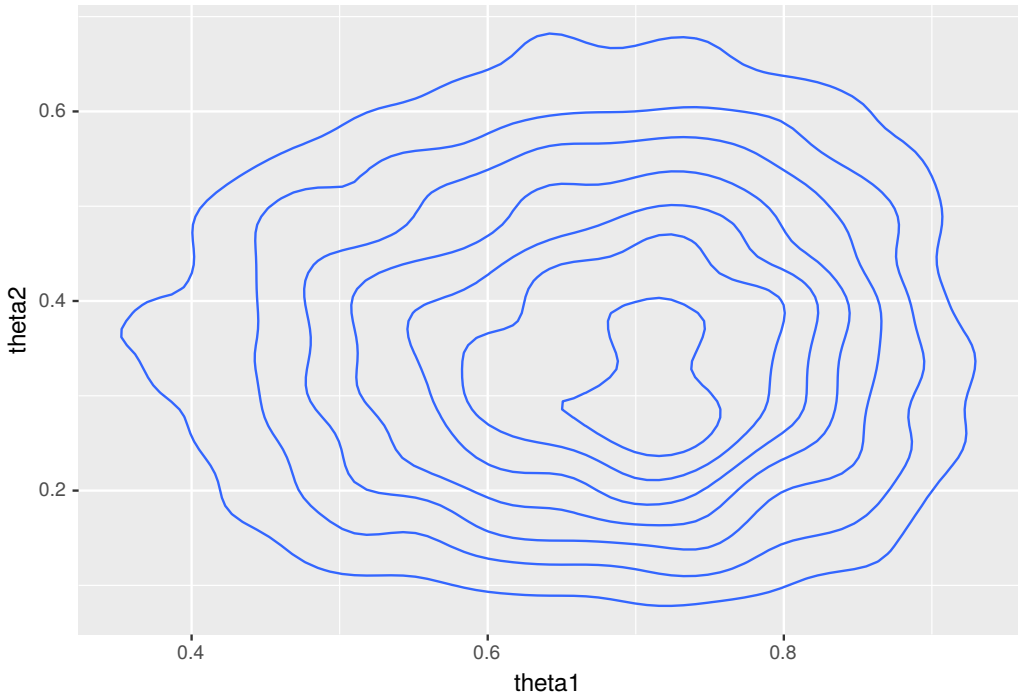

```
Gibbs %>% filter(step < 500) %>% gf_path(theta2 ~ theta1,
    color = ~step, alpha = 0.5, arrow = arrow(type = "open",
        angle = 30, length = unit(0.1, "inches"))) +
    theme_minimal()
```



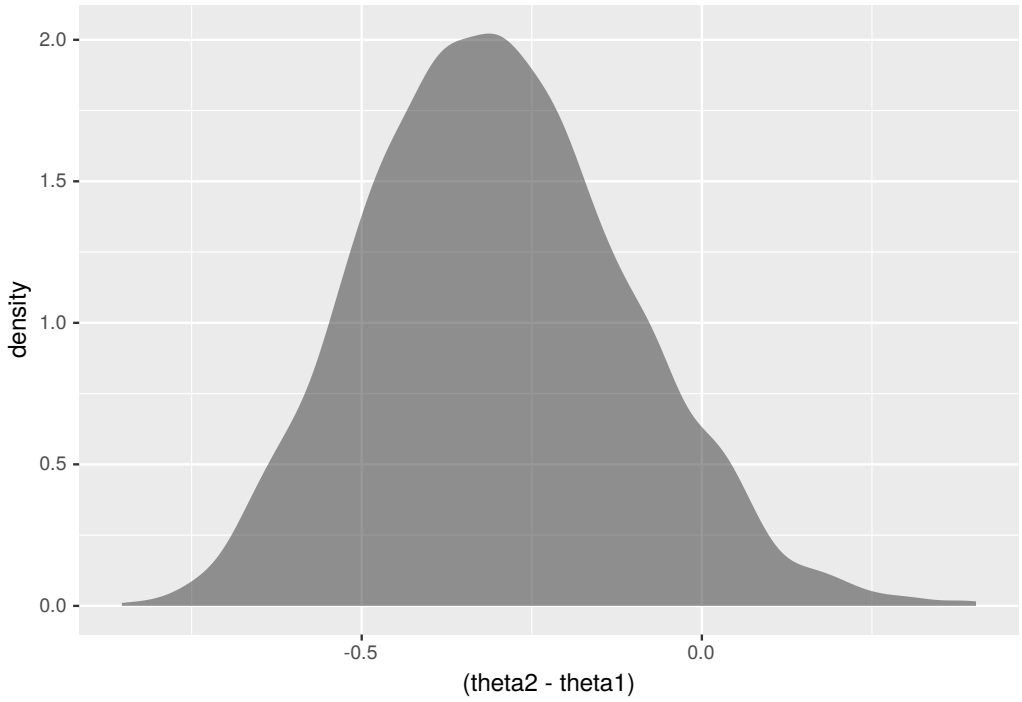

```
Gibbsplot <- Gibbs %>% filter(step < 500) %>% gf_path(theta2 ~
    theta1, color = ~step, alpha = 0.5, arrow = arrow(type = "open",
    angle = 30)) + theme_minimal() + transition_reveal(step)

animate(Gibbsplot, fps = 1)
```

Ciclos completos

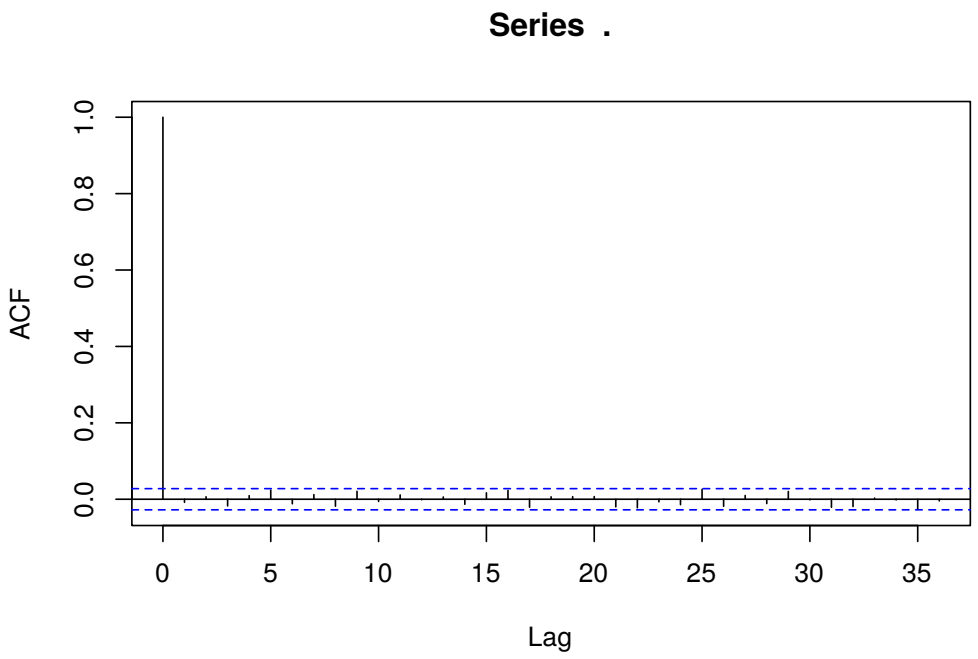
```
Gibbs %>% filter(step%%2 == 0) %>% gf_density2d(theta2 ~
    theta1)
```



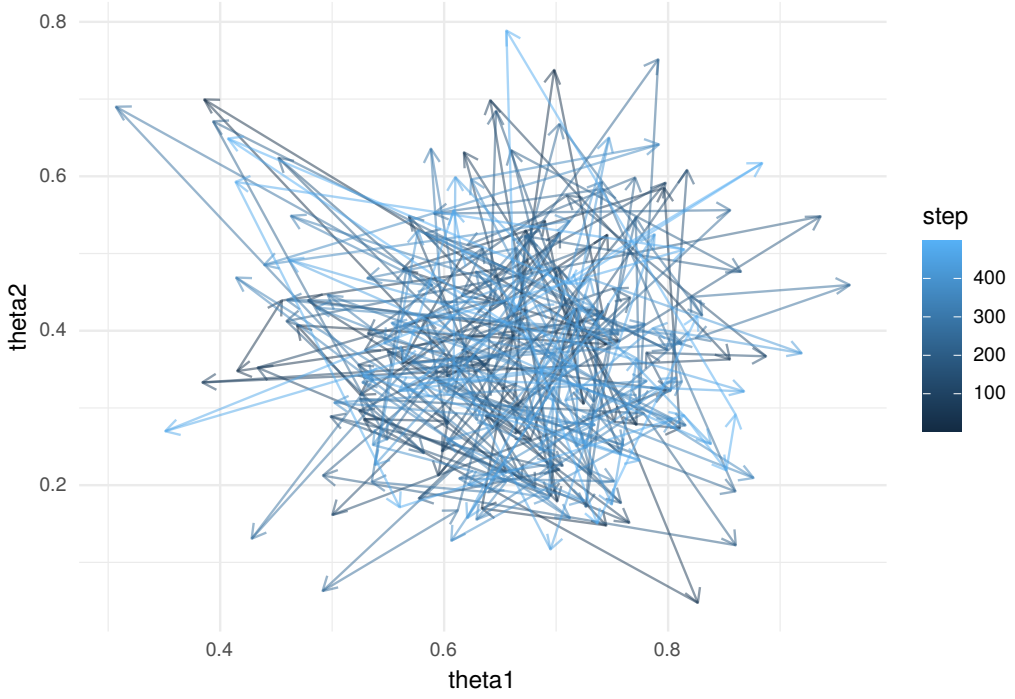

```
Gibbs %>% filter(step%%2 == 0) %>% gf_density(~(theta2 -
    theta1))
```




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




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




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

animate(Gibbsplot2, fps = 1)
```

4.6. Uso de JAGS

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

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

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

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

```
## Rows: 50
## Columns: 1
## $ y <int> 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 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 lenguage no existen versiones vectorizadas de las funciones por lo que todo debe llenarse usando `for`'s. Una revisión completa de este lenguage la pueden econtrar en su manual de uso [1]

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

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

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

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

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

Veamos el resultado

```
bern_jags
```

```
## Inference for Bugs model at "/tmp/Rtmp1ygvHJ/model24b219a8212.txt", fit usi
##  3 chains, each with 2000 iterations (first 1000 discarded)
##  n.sims = 3000 iterations saved
##          mu.vect sd.vect   2.5%    25%    50%    75%  97.5%  Rhat n.eff
## theta      0.307   0.064  0.189  0.264  0.305  0.349  0.438 1.001  3000
## deviance  62.069   1.368 61.087 61.189 61.530 62.403 65.725 1.001  3000
##
## For each parameter, n.eff is a crude measure of effective sample size,
```

```
## and Rhat is the potential scale reduction factor (at convergence, Rhat=1).
##
## DIC info (using the rule, pD = var(deviance)/2)
## pD = 0.9 and DIC = 63.0
## DIC is an estimate of expected predictive error (lower deviance is better).
```

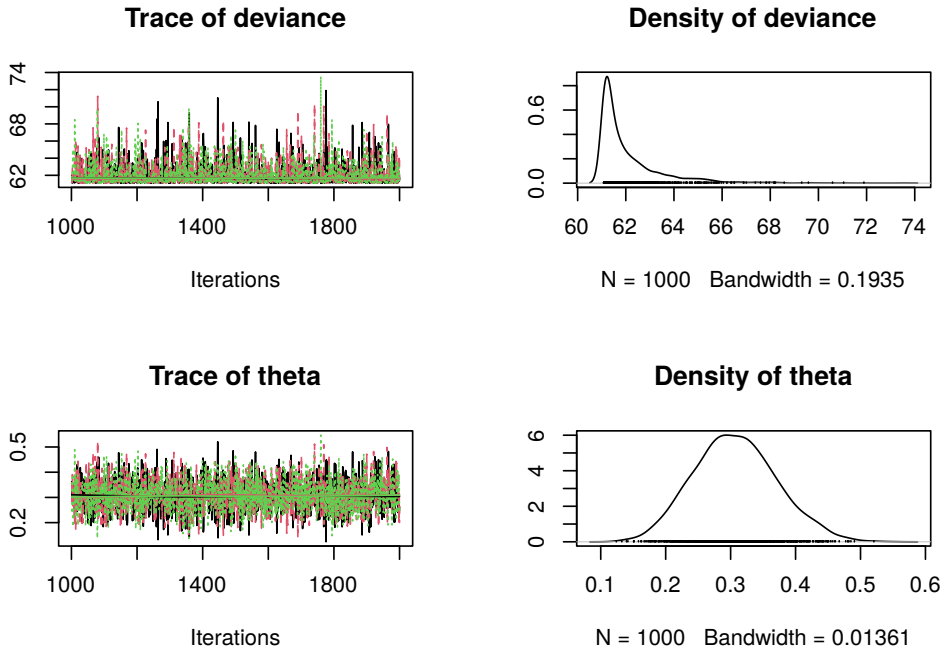
```
head(posterior(bern_jags))
```

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

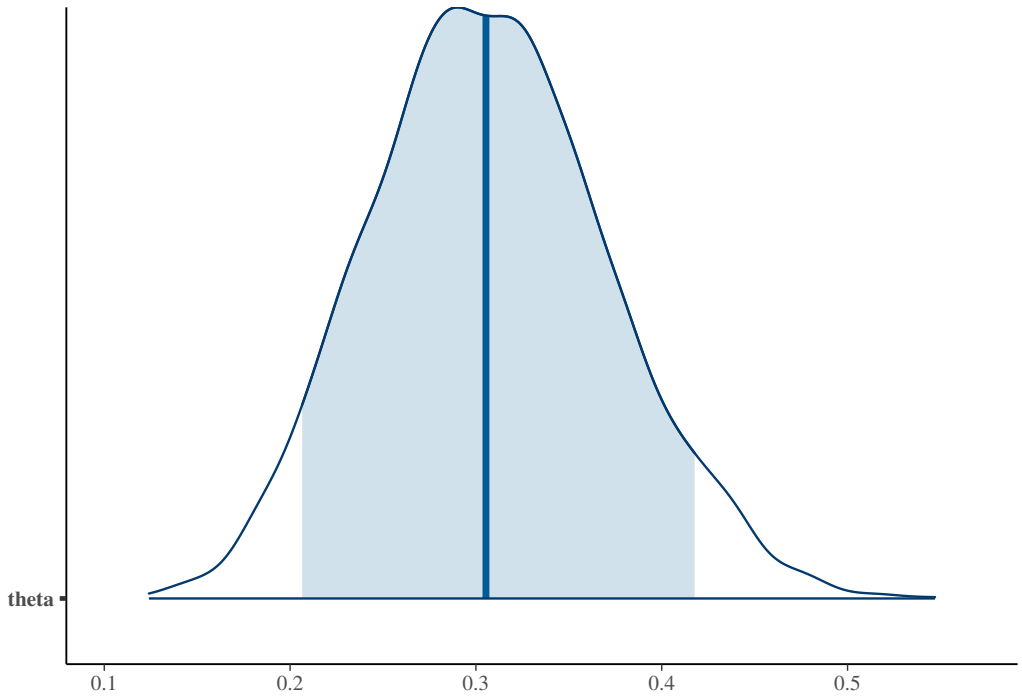
```
gf_dhistogram(~theta, data = posterior(bern_jags),
    bins = 50) %>% gf_dens(~theta, size = 1.5, alpha = 0.8) %>%
    gf_dist("beta", shape1 = 16, shape2 = 36, color = "red")
```

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

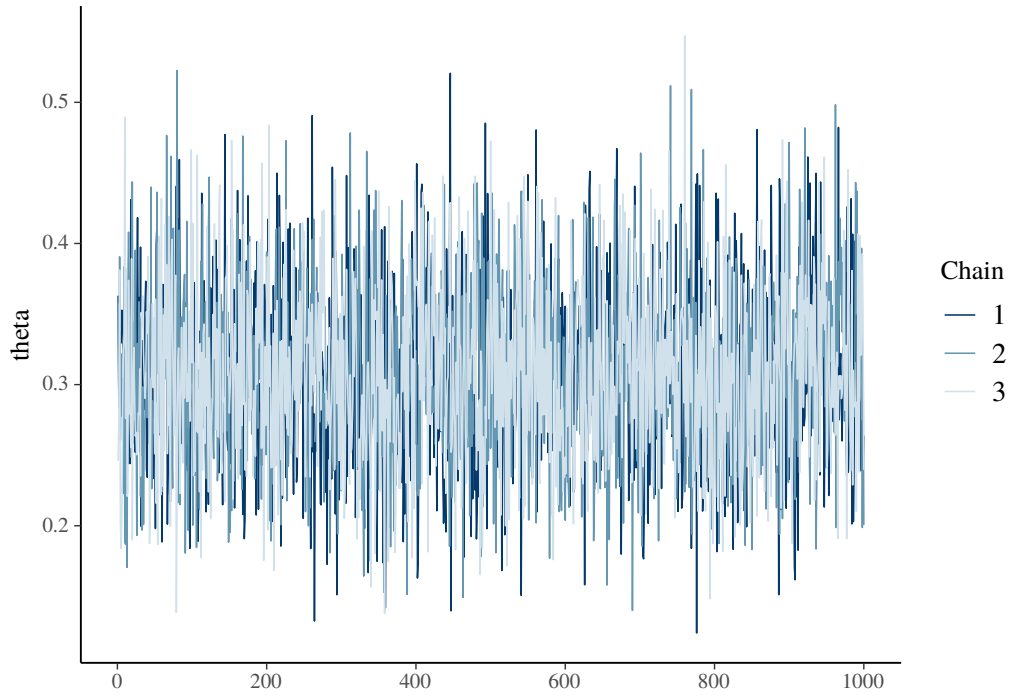
```
bern_mcmc <- as.mcmc(bern_jags)
plot(bern_mcmc)
```



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



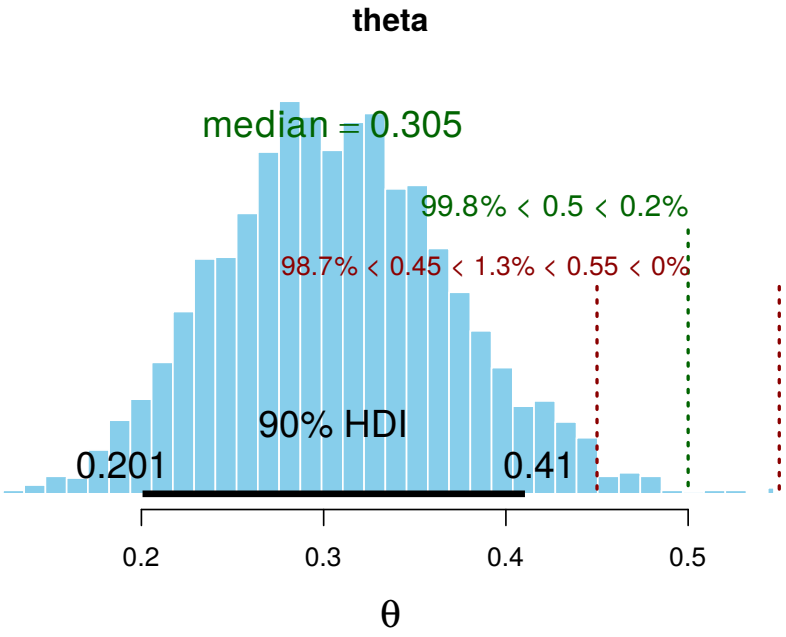

```
mcmc_trace(bern_mcmc, pars = "theta")
```




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

```
## Error: At least one layer must contain all faceting 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
1	Reginald
1	Reginald
1	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
```

```
    }
}
```

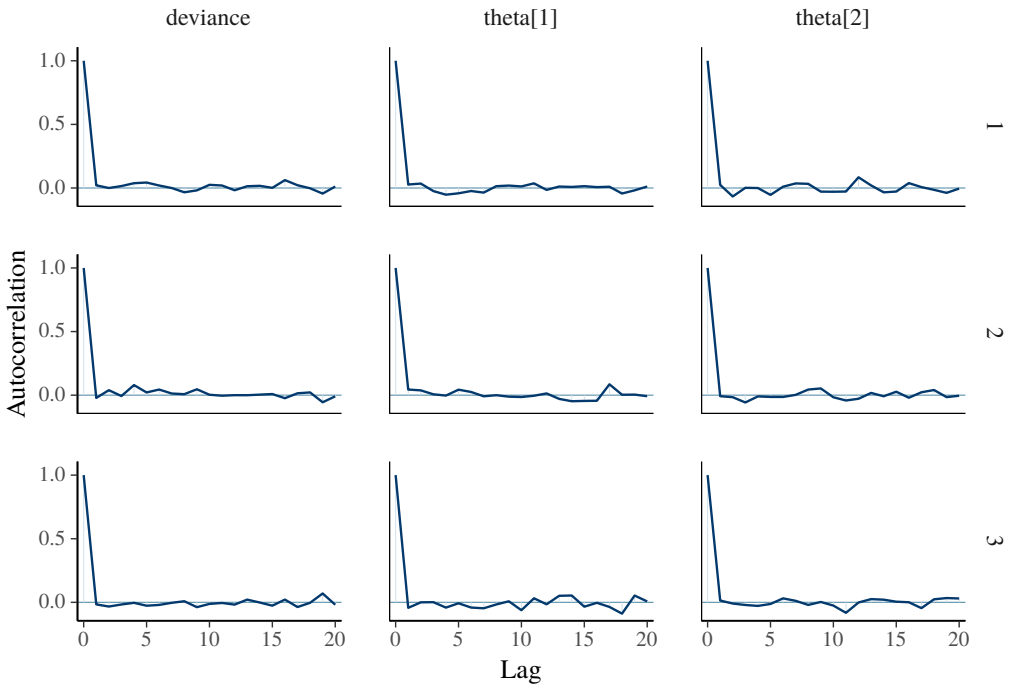
```
TargetList <- list(Nobs = nrow(Target), Nsub = 2, hit = Target$hit,
    subject = as.numeric(as.factor(Target$subject)))
TargetList
```

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

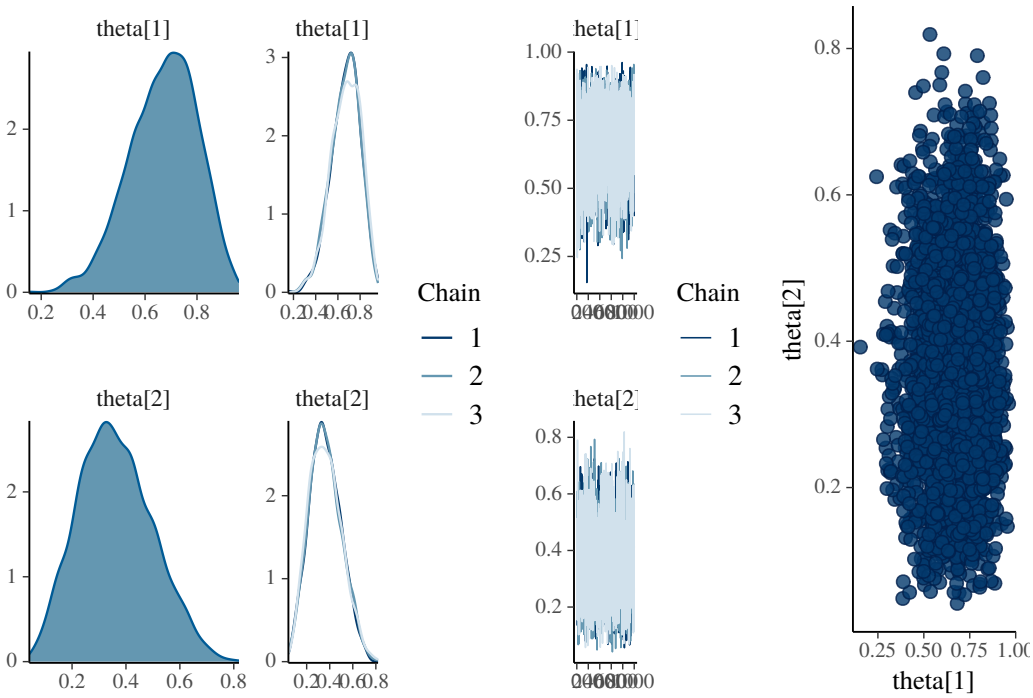
```
bern2_jags <- jags(data = TargetList, model = bern2_model,
    parameters.to.save = "theta")
```

```
## 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]"))
```



4.7. Uso de STAN

STAN[2] es otro tipo de lenguaje para definir modelos bayesiano. El lenguaje es un poco más sencillo, pero es particularmente útil para modelos bastante complejos.

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

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

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

[2] https://mc-stan.org/

```
}

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 1.4e-05 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take
## Chain 1: Adjust your expectations accordingly!
## Chain 1:
## Chain 1:
## Chain 1: Iteration:    1 / 5000 [  0%]  (Warmup)
## Chain 1: Iteration:  500 / 5000 [ 10%]  (Warmup)
## Chain 1: Iteration: 1000 / 5000 [ 20%]  (Warmup)
## Chain 1: Iteration: 1500 / 5000 [ 30%]  (Warmup)
## Chain 1: Iteration: 2000 / 5000 [ 40%]  (Warmup)
## Chain 1: Iteration: 2500 / 5000 [ 50%]  (Warmup)
## Chain 1: Iteration: 2501 / 5000 [ 50%]  (Sampling)
## Chain 1: Iteration: 3000 / 5000 [ 60%]  (Sampling)
## Chain 1: Iteration: 3500 / 5000 [ 70%]  (Sampling)
## Chain 1: Iteration: 4000 / 5000 [ 80%]  (Sampling)
## Chain 1: Iteration: 4500 / 5000 [ 90%]  (Sampling)
## Chain 1: Iteration: 5000 / 5000 [100%]  (Sampling)
## Chain 1:
```

```
## Chain 1:  Elapsed Time: 0.032881 seconds (Warm-up)
## Chain 1:                0.035488 seconds (Sampling)
## Chain 1:                0.068369 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 0.07 s
## Chain 2: Adjust your expectations accordingly!
## Chain 2:
## Chain 2:
## Chain 2: Iteration:    1 / 5000 [  0%]  (Warmup)
## Chain 2: Iteration:  500 / 5000 [ 10%]  (Warmup)
## Chain 2: Iteration: 1000 / 5000 [ 20%]  (Warmup)
## Chain 2: Iteration: 1500 / 5000 [ 30%]  (Warmup)
## Chain 2: Iteration: 2000 / 5000 [ 40%]  (Warmup)
## Chain 2: Iteration: 2500 / 5000 [ 50%]  (Warmup)
## Chain 2: Iteration: 2501 / 5000 [ 50%]  (Sampling)
## Chain 2: Iteration: 3000 / 5000 [ 60%]  (Sampling)
## Chain 2: Iteration: 3500 / 5000 [ 70%]  (Sampling)
## Chain 2: Iteration: 4000 / 5000 [ 80%]  (Sampling)
## Chain 2: Iteration: 4500 / 5000 [ 90%]  (Sampling)
## Chain 2: Iteration: 5000 / 5000 [100%]  (Sampling)
## Chain 2:
## Chain 2:  Elapsed Time: 0.033237 seconds (Warm-up)
## Chain 2:                0.034277 seconds (Sampling)
## Chain 2:                0.067514 seconds (Total)
## Chain 2:
##
## SAMPLING FOR MODEL '4584de91ce47196187979d2da8a67926' NOW (CHAIN 3).
## Chain 3:
## Chain 3: Gradient evaluation took 7e-06 seconds
## Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0.07 s
## Chain 3: Adjust your expectations accordingly!
## Chain 3:
## Chain 3:
## Chain 3: Iteration:    1 / 5000 [  0%]  (Warmup)
```

```
## Chain 3: Iteration:  500 / 5000 [ 10%]  (Warmup)
## Chain 3: Iteration: 1000 / 5000 [ 20%]  (Warmup)
## Chain 3: Iteration: 1500 / 5000 [ 30%]  (Warmup)
## Chain 3: Iteration: 2000 / 5000 [ 40%]  (Warmup)
## Chain 3: Iteration: 2500 / 5000 [ 50%]  (Warmup)
## Chain 3: Iteration: 2501 / 5000 [ 50%]  (Sampling)
## Chain 3: Iteration: 3000 / 5000 [ 60%]  (Sampling)
## Chain 3: Iteration: 3500 / 5000 [ 70%]  (Sampling)
## Chain 3: Iteration: 4000 / 5000 [ 80%]  (Sampling)
## Chain 3: Iteration: 4500 / 5000 [ 90%]  (Sampling)
## Chain 3: Iteration: 5000 / 5000 [100%]  (Sampling)
## Chain 3:
## Chain 3:  Elapsed Time: 0.03141 seconds (Warm-up)
## Chain 3:                0.032364 seconds (Sampling)
## Chain 3:                0.063774 seconds (Total)
## Chain 3:
##
## SAMPLING FOR MODEL '4584de91ce47196187979d2da8a67926' NOW (CHAIN 4).
## Chain 4:
## Chain 4: Gradient evaluation took 8e-06 seconds
## Chain 4: 1000 transitions using 10 leapfrog steps per transition would take
## Chain 4: Adjust your expectations accordingly!
## Chain 4:
## Chain 4:
## Chain 4: Iteration:    1 / 5000 [  0%]  (Warmup)
## Chain 4: Iteration:  500 / 5000 [ 10%]  (Warmup)
## Chain 4: Iteration: 1000 / 5000 [ 20%]  (Warmup)
## Chain 4: Iteration: 1500 / 5000 [ 30%]  (Warmup)
## Chain 4: Iteration: 2000 / 5000 [ 40%]  (Warmup)
## Chain 4: Iteration: 2500 / 5000 [ 50%]  (Warmup)
## Chain 4: Iteration: 2501 / 5000 [ 50%]  (Sampling)
## Chain 4: Iteration: 3000 / 5000 [ 60%]  (Sampling)
## Chain 4: Iteration: 3500 / 5000 [ 70%]  (Sampling)
## Chain 4: Iteration: 4000 / 5000 [ 80%]  (Sampling)
## Chain 4: Iteration: 4500 / 5000 [ 90%]  (Sampling)
## Chain 4: Iteration: 5000 / 5000 [100%]  (Sampling)
## Chain 4:
## Chain 4:  Elapsed Time: 0.030109 seconds (Warm-up)
```

```
## Chain 4:                0.018828 seconds (Sampling)
## Chain 4:                0.048937 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  3763    1
## lp__  -32.59    0.01 0.70 -33.43 -32.10  4316    1
##
## Samples were drawn using NUTS(diag_e) at Wed May 13 11:53:09 2020.
## For each parameter, n_eff is a crude measure of effective sample size,
## and Rhat is the potential scale reduction factor on split chains (at
## convergence, Rhat=1).
```

```
theta_draws <- extract(fit)$theta

mean(theta_draws)
```

```
## [1] 0.3062755
```

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

```
##       10%       90%
## 0.2278129 0.3897041
```

```
shinystan::launch_shinystan(fit)
```

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

4.8. Ejercicios

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

Bibliografía

Albert, Jim y col. (2009). *Bayesian computation with R*. New York, NY: Springer New York, págs. 1-298.

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*. Springer Series in Statistics. Berlin, Heidelberg: Springer Berlin Heidelberg, págs. xxviii+299.

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.

Kruschke, John K. (2014). «Doing Bayesian data analysis: A tutorial with R, JAGS, and Stan, second edition». En: *Doing Bayesian Data Analysis: A Tutorial with R, JAGS, and Stan, Second Edition*, págs. 1-759.

Quenouille, M. H. (ene. de 1949). «Approximate Tests of Correlation in Time-Series». En: *Journal of the Royal Statistical Society: Series B (Methodological)* 11.1, págs. 68-84.

Wasserman, Larry (2006). *All of Nonparametric Statistics*. Springer Texts in Statistics. New York, NY: Springer New York.