Ejemplo Poisson

En el fichero Datos-Encuesta-USA.RData aparecen datos del General Social Survey hecho en USA a finales de los 90, hemos seleccionado sólo 5 de las variables presentes en la encuesta, concretamente:

- *FEMALE*: Indica si la persona encuestada es hombre (0) o mujer (1).
- *CHILDS*: Número de hijos que tiene la persona encuestada.
- *YEAR*: Año en el que la persona participó en la encuesta (entre 1972 y 1998).
- *AGE*: Edad de la persona en el momento de participar en la encuesta.
- *DEG*: Nivel de estudios, variable que va de 0 a 4, donde 3 y 4 quiere decir grado universitario o superior.

Lee los datos con ayuda de R.

```
load("DatosEncuestaUSA.RData")
```

El objetivo es analizar el número de hijos que tienen las mujeres que tenían una edad de 40 años en el instante de participación en la encuesta, interesa analizar las mujeres que participaron en la encuesta en la década de 1990.

Con el siguiente código R puedes seleccionar la variable de interés, hijos, de mujeres con 40 años y que particiaron en los años 90 en la encuesta (la encuesta no incluye años más allá de 1999).

```
hijos<-Y$CHILDS[Y$FEMALE==1 & Y$YEAR>=1990 & Y$AGE==40 & !is.na(Y$DEG)]
hijos<-hijos[!is.na(hijos)]
```

En el objeto hijos aparecen el número de hijos que tienen las mujeres selccionadas. En concreto se trata de una muestra de tamaño $n = 155$.

Es muy usual en la práctica de la estadística usar el modelo *Poisson* para representar una variable aleatoria que es un conteo, veremos más adelante cómo usar técnicas gráficas y procedimientos para analizar suposiciones como ésta (asumir un determinado modelo paramétrico para uos datos). Por el momento, supongamos que

$$N_i = \text{ Número de hijos que tiene cada mujer de esta población}$$

esta variable es aleatoria, y representamos su aleatoriedad mediante una $Po(\lambda)$, con λ el número esperado de hijos que tienen las mujeres de esta poblacion.

Es decir la masa de probabilidad, suponiendo λ conocido es:

$P(X = x) = \frac{e^{-\lambda}\lambda^x}{x!}$, $x = 0, 1, 2, ..., \infty$

El objetivo de este ejercicio es estimar puntualmente λ.

Estimador máximo verosímil

Escribe la función de verosimilitud para λ. Recuerda que es:

$$L(\lambda|n_1, \ldots, n_{155}) = f(n_1, \ldots, n_{155}|\lambda)$$

Escribe esta función para el caso del modelo Poisson. Encuenta el valor de λ que lo maximiza. Puedes usar técnicas matemáticas para encontrar el máximo, o calcularlo con ayuda de R (dibujando la función, donde el argumento es λ, y usando la función *optimize* de R para buscar el valor que maximiza esta función).

Para buscar el máximo usando R tienes que programar primero la función verosimilitud en R (yo la he llamado *vero.pois*), y después usar los siguientes argumentos:

optimize(f = , interval = , ..., lower = min(interval), upper = max(interval), maximum = FALSE, tol = .Machine$double.eps^0.25)

Tienes que especificar que optimize calcule un máximo, y que busque en un intervalo (por ejemplo prueba con el intervalo 0, 10), además hay que pasarle el resto de argumentos de la función *vero.pois*.

Respuestas

Primero calculamos matemáticamente el estimador verosímil. Para ello partimos de la premisa de que la función verosímil es igual a la función de densidad conjunta. Entendemos que las muestras han sido escogidas aleatoriamente, por lo que las consideramos independientes entre sí. Debido a esto, la función de densidad conjunta es igual al producto de las densidades marginales:

$$L(\lambda|n_1,\ldots,n_{155}) = f(n_1,\ldots,n_{155}|\lambda) = f(N_1 = n_1,\ldots,N_{155} = n_{155}) = \prod_{i=1}^{k=155} f(N_i = n_i)$$

Sustituyendo la función de densidad por la Poisson, obtenemos el siguiente resultado como función de verosimilitud:

$$\prod_{i=1}^{k=155} f(N_i = n_i) = \prod_{i=1}^{k=155} \frac{e^{-\lambda}\cdot\lambda^{n_i}}{n_i!}$$

Desarrollando el producto de densidades para obtener una función definitiva para la verosimilitud:

$$\prod_{i=1}^{k=155} \frac{e^{-\lambda}\cdot\lambda^{n_i}}{n_i!} = \frac{e^{-\lambda\cdot k}\cdot\lambda^{\sum_{i=1}^{k=155} n_i}}{\prod_{i=1}^{k=155} n_i!}$$

A modo de simplificar los cálculos, aplicamos logaritmos y sus propiedades. De esta manera conseguimos una función más sencilla

$$ln(\frac{e^{-\lambda\cdot k}\cdot\lambda^{\sum_{i=1}^{k=155} n_i}}{\prod_{i=1}^{k=155} n_i!}) =$$

$$ln(\ e^{-\lambda\cdot k}\cdot\lambda^{\sum_{i=1}^{k=155} n_i}) - ln(\prod_{i=1}^{k=155} n_i!) =$$

$$[ln\ e^{-\lambda\cdot k} + ln\lambda^{\sum_{i=1}^{k=155} n_i}] - [\sum_{i=1}^{k=155} ln\ n_i!] =$$

$$-\lambda\cdot k + \sum_{i=1}^{k=155} n_i\ \cdot ln\ \lambda - \sum_{i=1}^{k=155} ln\ n_i!$$

Pasamos la fórmula sin logaritmos a R.

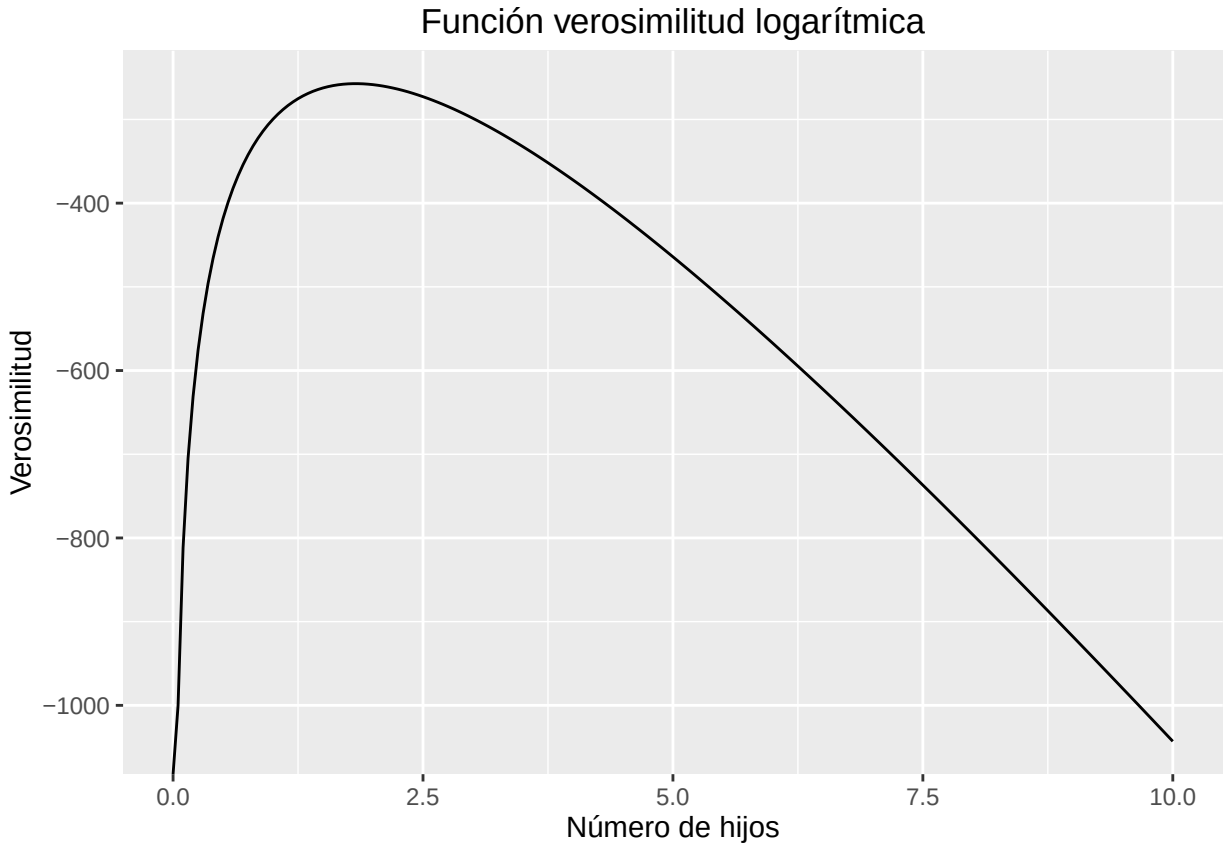
```
vero.pois <- function(lambda){(exp(-lambda*k)*lambda^(sum(n)))/prod(factorial(n))}
```

Pasamos también la función obtenida de aplicar logaritmos.

```
log.vero.pois <- function(lambda){-lambda*k + sum(n)*log(lambda) - sum(log(factorial(n)))}
```

Representamos la función

```r
library(ggplot2)
k = length(hijos)
n = hijos
upper_interv = 10
lower_interv = 0
steps = 0.05
lambda_interv = seq(from = lower_interv, to = upper_interv, by = steps)
valores_lambda = log.vero.pois(lambda_interv)
verosimilitud_df = data.frame(log.vero.pois_x = lambda_interv, log.vero.pois_y = valores_lambda)
ggplot(verosimilitud_df, aes(log.vero.pois_x, log.vero.pois_y)) +
  geom_line() +
  labs(x="Número de hijos", y="Verosimilitud") +
  ggtitle("Función verosimilitud logarítmica")
```



Observando el gráfico podemos ver en torno a qué valores se encuentra el máximo verosímil de la función.

Usamos la función optimize para calcular el máximo verosímil

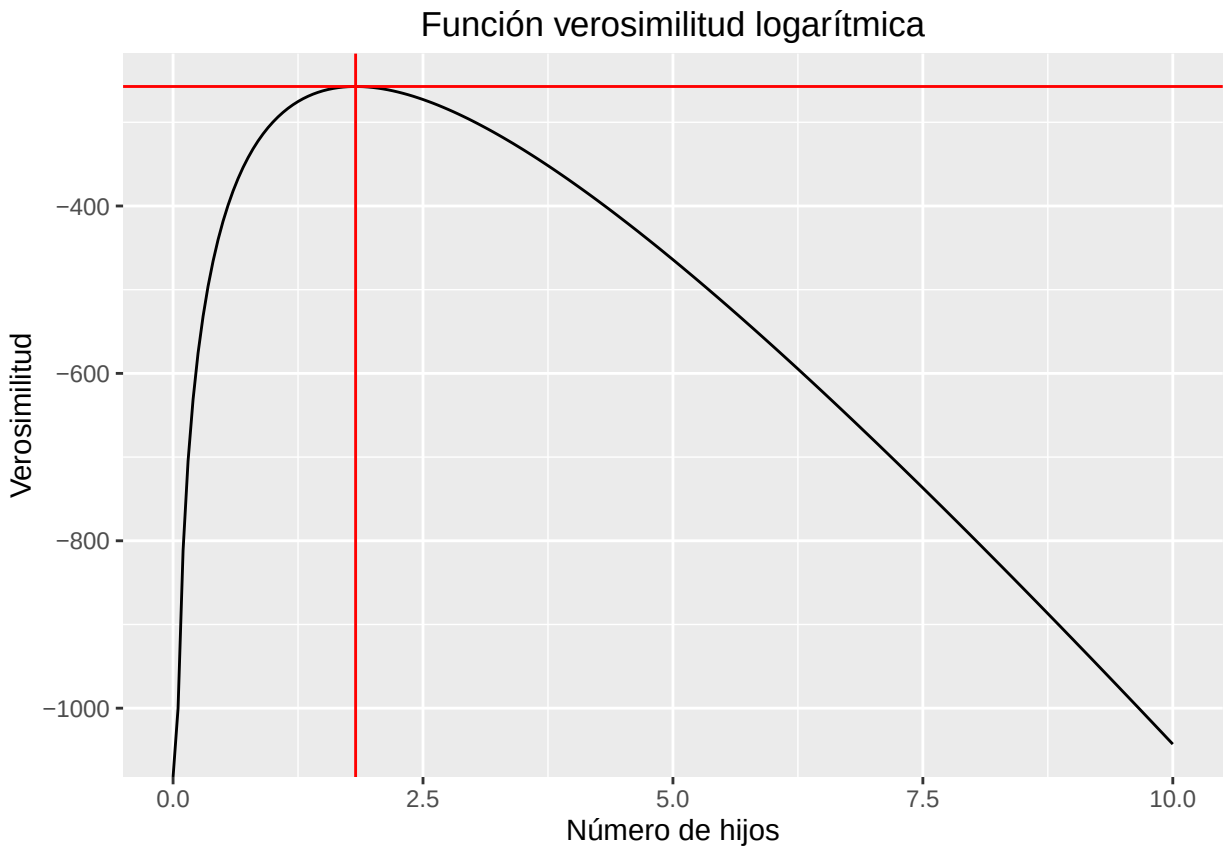
```r
max_verom=optimize(log.vero.pois,lambda_interv, lower = min(lambda_interv), upper = max(lambda_interv),
```

El máximo verosímil es 1.8258108

Representamos el máximo en la función verosímil.

```r
library(ggplot2)
k = length(hijos)
n = hijos
upper_interv = 10
lower_interv = 0
```

```r
steps = 0.05
lambda_interv = seq(from = lower_interv, to = upper_interv, by = steps)
valores_lambda = log.vero.pois(lambda_interv)
verosimilitud_df = data.frame(log.vero.pois_x = lambda_interv, log.vero.pois_y = valores_lambda)
ggplot(verosimilitud_df, aes(log.vero.pois_x, log.vero.pois_y)) +
  geom_line() +
  geom_hline(yintercept = max_verom$objective, color = 'red') +
  geom_vline(xintercept = max_verom$maximum, color = 'red') +
  labs(x="Número de hijos", y="Verosimilitud") +
  ggtitle("Función verosimilitud logarítmica")
```



Estimador Bayesiano

Para el modelo Poisson existe una distribución a priori conjugada, la densidad Gamma sobre λ, de parámetros a y b.

$f(\lambda) = \frac{b^a}{\Gamma(a)} \lambda^{a-1} exp\,(-b\lambda)$, $0 < \lambda < \infty$

Supón que usamos una distribución inicial sobre $\lambda \sim Ga(2, 1)$. Calcula la distribución a posteriori. Dibuja con ayuda de R ambas funciones (código muy parecido al que usamos para el modelo Beta-Binomial), pero ahora hay que usar la distribución adecuada como distribución a posteriori, con los parámetros que tocan.

Proporciona un estimador puntual para λ usando esta distribución a posteriori.

Respuestas

Para calcular el estimador puntual Bayesiano para λ, primero tenemos que calcular el estimador Bayesiano o

función a posteri. Nos apoyamos en la probabilidad condicionada para ello:

$$f(\lambda|N=n) = \frac{f(\lambda)\cdot f(n|\lambda)}{f(n)}$$

Queremos obtener una estimación para lambda, a partir de la función anterior. Para ello, se multiplca la función de densidad a priori por la verosimilitud, entre la función de las muestras. Y, representa las muestras tomadas.

De cara al cálculo de la distribución a priori, el denominador sólo tiene efectos de normalización ya que no depende del parámetro a estimar, por lo que podríamos obviarlo, quedando una distribución a posteriori similar al producto de la función a priori por la función de verosimilitud $f(y|\lambda)$

$$f(\lambda|N=n) \sim f(\lambda)\cdot f(n|\lambda)$$

Tenemos los siguientes términos:

Distribución a priori

$$f(\lambda) = \frac{b^a}{\Gamma(a)}\cdot\lambda^{a-1}\cdot e^{-b\cdot\lambda},\ para\ 0<\lambda<\infty$$

Función de verosimilitud, con $k = 155$ (tamaño de las muestras)

$$f(n|\lambda) = \frac{e^{-\lambda\cdot k}\cdot\lambda^{\sum_{i=1}^{k=155} n_i}}{\prod_{i=1}^{k=155} n_i!}$$

El denominador no depende del parámetro a estimar, λ, por lo que podemos trabajar con una función de verosimilitud como la siguiente:

$$f(n|\lambda) \sim e^{-\lambda\cdot k}\cdot\lambda^{\sum_{i=1}^{k=155} n_i}$$

Distribución a posteriori

$$f(\lambda|N=n) \sim \frac{b^a}{\Gamma(a)}\cdot\lambda^{a-1}\cdot e^{-b\cdot\lambda}\cdot e^{-\lambda\cdot k}\cdot\lambda^{\sum_{i=1}^{k=155} n_i}$$

Operando:

$$f(\lambda|N=n) \sim \frac{b^a}{\Gamma(a)}\cdot\lambda^{a-1+\sum_{i=1}^{k=155} n_i}\cdot e^{-b\cdot\lambda-\lambda\cdot k}$$

Como podemos observar, el resultado final es otra función Gamma con diferentes términos en los exponentes. En este caso, la función a posteriori, sigue la misma distribución que la priori, una Gamma.

Para obtener los parámetros de la Gamma a posterior, juntamos términos y tenemos que:

$$a_{post} = a + \sum_{i=1}^{k=155} n_i$$

$$b_{post} = b + k$$

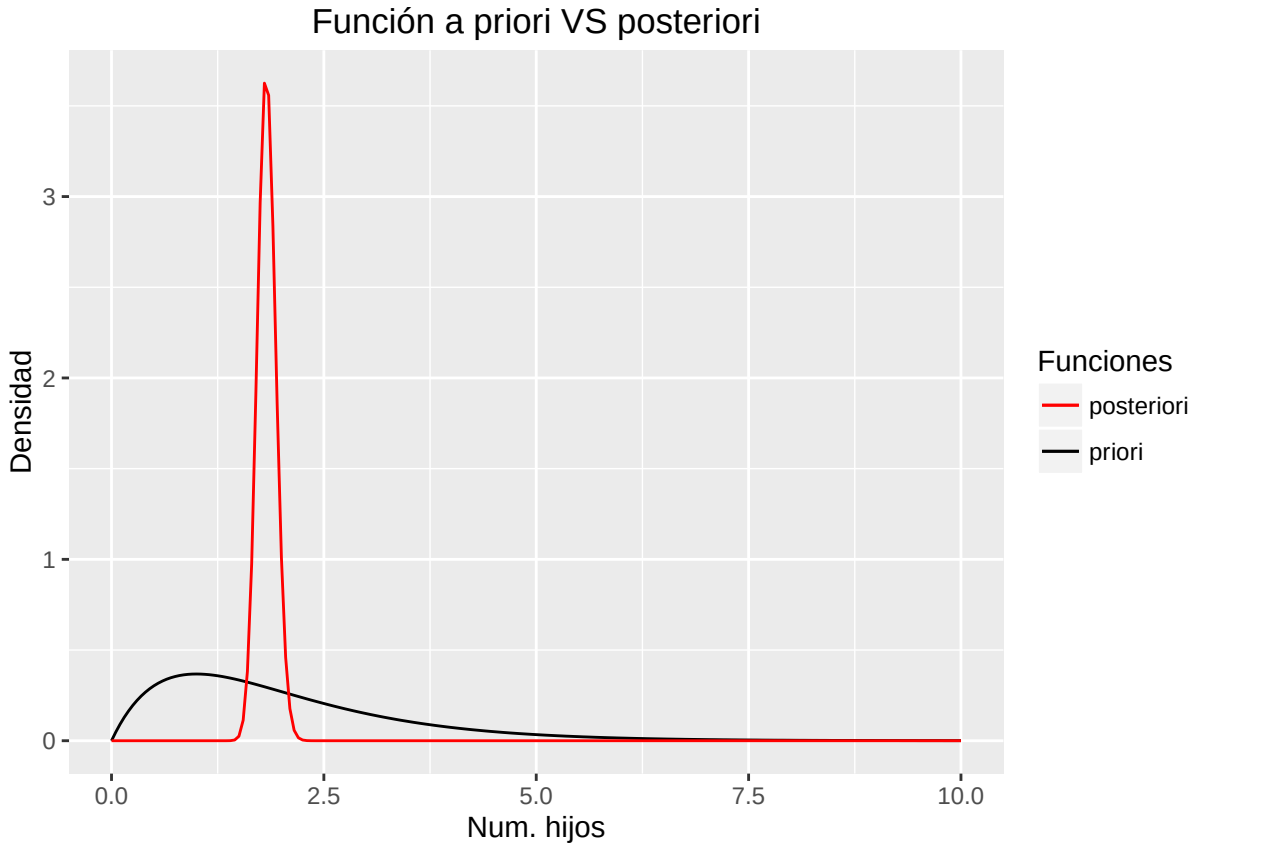
```
a <- 2
b <- 1
apost <- a + sum(n)
bpost <- b + k

upper_interv = 10
lower_interv = 0
```

```r
steps = 0.05
lambda_interv = seq(from = lower_interv, to = upper_interv, by = steps)

f_priori <- dgamma(lambda_interv,a,b)
f_posteriori <- dgamma(lambda_interv,apost,bpost)
priori_posteriori_df = data.frame(lambda_x = lambda_interv, priori = f_priori, posteriori = f_posterior

ggplot(priori_posteriori_df, aes(lambda_x)) +
  geom_line(aes(y=priori, color ='priori')) +
  geom_line(aes(y=posteriori, color = 'posteriori')) +
  scale_color_manual(name = 'Funciones', values=c(priori='black', posteriori = 'red'))+
  labs(x="Num. hijos", y="Densidad") +
  ggtitle("Función a priori VS posteriori")
```



Una vez obtenida la función a posterior, calcularemos un estimador puntual para la variable λ. En este caso, usaremos la media.

Teniendo como posteriori,

$$f(\lambda|N=n) \sim Ga(apost, bpost)$$

el estimador sería el siguiente:

$$E[\lambda] = \frac{apost}{bpost} = \frac{a + \sum_{i=1}^{k=155} n_i}{b+k}$$

Si lo calculamos

```
estimacion_media_lambda = (a+sum(n))/(b+k)
```

En este caso, obtenemos un valor estimado de 1.8269231. *Si nos fijamos en la función a posteriori, la media se encuentra en torno al valor que se acaba de calcular.*