

EST-46115: Modelación Bayesiana

Profesor: Alfredo Garbuno Iñigo — Primavera, 2022 — Calibración basada en simulación .

Objetivo: En esta sección estudiaremos un mecanismo que puede ser utilizado para evaluar si un modelo está bien implementado. Esto aprovecha que la posterior es un procedimiento *calibrado* y recae en el hecho de que querríamos evaluar si nuestra implementación de un modelo es la correcta. Adicional, también querríamos identificar en qué parte del modelo se encuentra la deficiencia.

Lectura recomendada: Puedes consultar [2] el cual es una extensión de [1].

1. INTRODUCCIÓN

Decimos que un intervalo de probabilidad está bien **calibrado** si tiene la misma cobertura nominal con la que está construido. Digamos, si el intervalo es un intervalo de probabilidad del 80 % , entonces 4 de 5 veces éste contendrá al verdadero valor del parámetro.

Por definición, la distribución posterior está calibrada –si los datos son generados por el modelo–. Utilizando **calibración basada en simulación**~ explotaremos esta propiedad.

La idea es: generar parámetros de la previa para generar datos condicionados en los parámetros simulados con el objetivo de estudiar la calibración de la posterior bajo escenarios independientes.

2. CALIBRACIÓN DE LA POSTERIOR

Supongamos que tenemos un modelo bayesiano donde tenemos una distribución previa $\pi(\theta)$ para los parámetros del mecanismo generador de datos (verosimilitud) $\pi(y|\theta)$.

Ahora, consideremos que generamos una simulación

$$\theta_{\text{sim}} \sim \pi(\theta), \quad (1)$$

con la cual generamos

$$y_{\text{sim}} \sim \pi(y|\theta_{\text{sim}}). \quad (2)$$

Por construcción lo que hemos hecho es

$$(y_{\text{sim}}, \theta_{\text{sim}}) \sim \pi(y, \theta). \quad (3)$$

Ahora, la regla de Bayes nos dice que la distribución posterior es proporcional a la conjunta

$$\pi(\theta|y) \propto \pi(y, \theta). \quad (4)$$

Por lo que también podríamos pensar que

$$\theta_{\text{sim}} \sim \pi(\theta|y_{\text{sim}}). \quad (5)$$

Si utilizamos nuestro muestreador favorito para generar

$$\theta_1, \dots, \theta_M \sim \pi(\theta|y_{\text{sim}}). \quad (6)$$

Entonces podríamos comparar las muestras simuladas con el parámetro que generó los datos. Esto es, por que θ_{sim} es una realización aleatoria de la posterior, y por lo tanto los estadísticos de orden de θ_{sim} deberían de ser uniformes con respecto a los de $\theta_1, \dots, \theta_M$.

O dicho de otra manera,

$$\pi(\theta) = \int \pi(\theta|y_{\text{sim}})\pi(y_{\text{sim}}|\theta_{\text{sim}})\pi(\theta_{\text{sim}}) dy_{\text{sim}} d\theta_{\text{sim}}. \quad (7)$$

Lo cual nos da un mecanismo para evaluar qué tan bien estamos utilizando nuestro algoritmo para generar muestras de la posterior [2].

3. CALIBRACIÓN BASADA EN CALIBRACIÓN (CBS)

Utilizaremos la propiedad discutida arriba para generar

$$y_{\text{sim}}^{(n)}, \theta_{\text{sim}}^{(n)} \sim \pi(y, \theta), \quad n = 1, \dots, N. \quad (8)$$

Para cada uno de los datos simulados $y_{\text{sim}}^{(n)}$ utilizamos nuestro algoritmo para generar

$$\theta_1^{(n)}, \dots, \theta_M^{(n)} \sim \pi(\theta|y_{\text{sim}}^{(n)}). \quad (9)$$

Si consideramos los estadísticos de orden—el número de simulaciones de la posterior menores al parámetro simulado—en cada componente de nuestro vector de parámetros, tendremos que

$$r_n = \text{orden} \left(\theta_{\text{sim}}^{(n)}, \left\{ \theta_1^{(n)}, \dots, \theta_M^{(n)} \right\} \right) \quad (10)$$

$$= \sum_{m=1}^M 1[\theta_m^{(n)} < \theta_{\text{sim}}^{(n)}], \quad (11)$$

será un entero distribuido $\text{Uniforme}\{0, \dots, M\}$. Esto es, el estadístico de orden r_n tiene probabilidad $1/(M+1)$ de tomar valores entre $0, \dots, M$.

Talts et al. [2] histogramas. Cook et al. [1] prueba de hipótesis que pone a prueba la uniformidad.

3.1. Ejemplo: Modelo conjugado

Consideremos un modelo de una observación Gaussiana $y \sim \text{N}(\mu, \sigma^2)$, en donde utilizamos el siguiente conocimiento previo para los parámetros

$$\mu \sim \text{N}(0, 1), \quad (12)$$

y asumimos una σ conocida.

Este modelo es un modelo conjugado **Normal-Normal** el cual tiene una distribución posterior

$$\mu|y \sim \text{N} \left(\frac{y}{\sigma^2 + 1}, 1 + \frac{1}{\sigma^2} \right). \quad (13)$$

Consideremos $\sigma^2 = 2$. Si realizamos una simulación de la previa tenemos el siguiente parámetro:

```
1 set.seed(108791)
2 sim <- list(mu = rnorm(1))
3 sim > as.data.frame()
```

```
1      mu
2 1 0.61
```

Con los cuales podemos simular un conjunto de datos:

```
1 data <- list(y = rnorm(1, sim$mu, sd = sqrt(2)))
2 data
```

```
1 $y
2 [1] 1.4
```

Y con estos datos simulamos de la posterior $M = 4$ iteraciones:

```
1 params <- tibble(mu = rnorm(4, data$y/3, sd = sqrt(2/3)))
2 params > as.data.frame()
```

```
1      mu
2 1 0.947
3 2 0.037
4 3 1.268
5 4 0.954
```

Hacemos las comparaciones contra $\mu_{\text{sim}} = 0.61$:

```
1 params >
2   mutate(indicadora = ifelse(mu < sim$mu, 1, 0)) >
3   as.data.frame()
```

```
1      mu indicadora
2 1 0.947           0
3 2 0.037           1
4 3 1.268           0
5 4 0.954           0
```

Si calculamos el estadístico de rango, obtenemos una $r_{1,\mu} = 1$. El cual debería de estar uniformemente distribuido entre los enteros del 0 al 4. ¿lo ponemos a prueba?

```
1 experimento <- function(id){
2   sim <- list(mu = rnorm(1))
3   data <- list(y = rnorm(1, sim$mu, sd = sqrt(2)))
4   mu <- rnorm(4, data$y/3, sd = sqrt(2/3))
5   sum(mu < sim$mu)
6 }
7
8 resultados <- tibble(id = 1:100) >
9   mutate(rank = map_dbl(id, experimento))
```

La idea es replicar el procedimiento de generación de parámetros y muestras sintéticas con la intención de observar un comportamiento uniforme en los histogramas (Fig. 1).

Para cada réplica $n = 1, \dots, N$, podemos generar un número fijo de simulaciones de la posterior (M). Talts et al. [2] recomiendan simular tantas iteraciones de la posterior como se

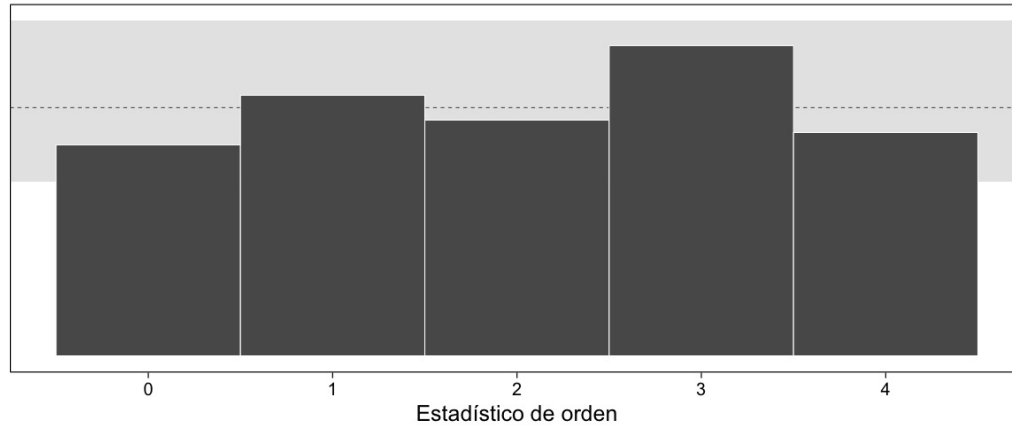


FIGURA 1. Histogramas de estadísticas de orden con 4 simulaciones de la posterior. Construimos una línea de referencia (y bandas de confianza) bajo los supuestos de la distribución uniforme de los estadísticos de orden.

requiera y *resumir* (agrupar) los resultados en 20 cubetas. De tal forma que podamos criticar un histograma de 20 barras. En la Fig. 2 observamos un histograma con 20 cubetas y la línea de referencia de un modelo uniforme con $M = 20$. Adicional, se muestran los intervalos de un experimento binomial con N réplicas con probabilidad $1/M$ de caer en cada cubeta.

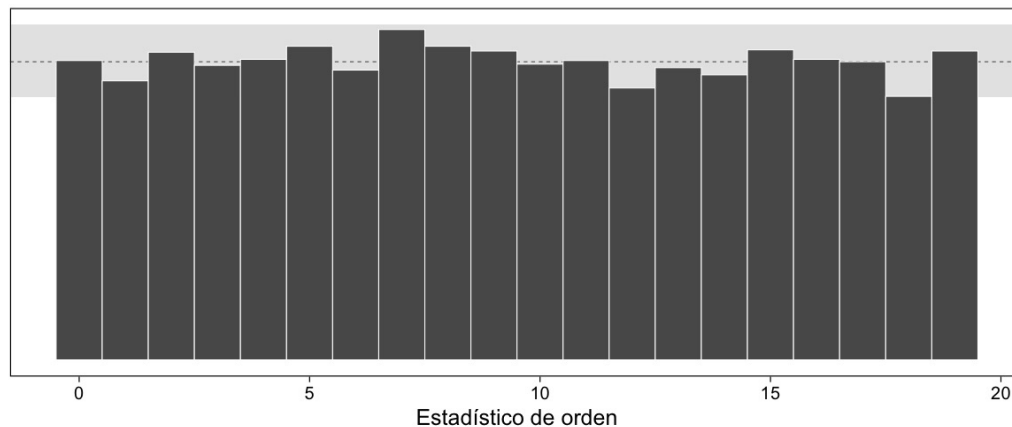


FIGURA 2. Histogramas de estadísticas de orden con 19 simulaciones de la posterior. Construimos una línea de referencia (y bandas de confianza) bajo los supuestos de la distribución uniforme de los estadísticos de orden.

El procedimiento descrito arriba nos permite evaluar de manera *visual* los histogramas. Alternativas a esta estrategia es poder evaluar la función de acumulación empírica (ECDF) contra el modelo uniforme. Esto también puede compararse de manera visual como se muestra en la Fig. 3, en donde estamos comparando contra la función de acumulación (CDF) de experimentos uniformes (panel izquierdo). Por otro lado, la comparación gráfica entre la ECDF y CDF se vuelve compleja en realizarse si el número de cubetas (M) es muy elevado. Por eso tendemos a comparar la diferencia, asumiendo una aproximación Gaussiana (panel derecho).

3.2. Cuando el modelo está mal especificado

Consideremos los errores típicos de una implementación de un modelo. Por ejemplo, tenemos un modelo que tiene una dispersión mas pequeña que la que debería. En estas situaciones tenemos un comportamiento de los histogramas en forma de \cup como se muestra en la Fig. 4.

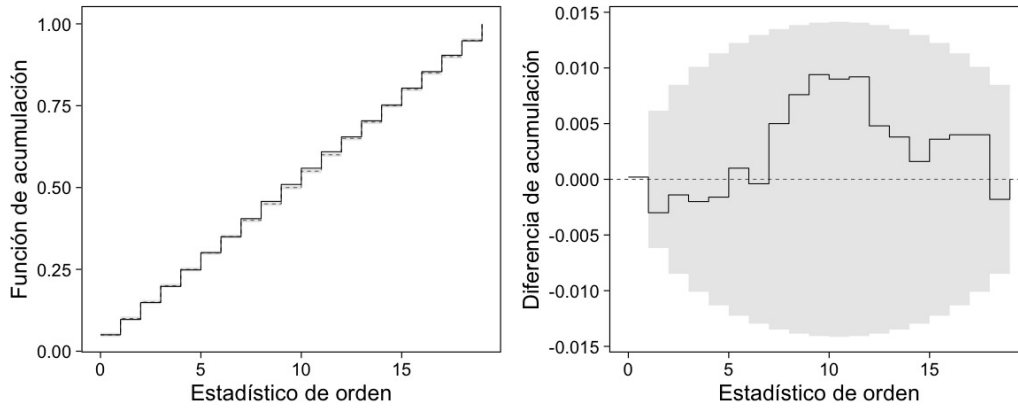


FIGURA 3. Gráficos alternativos para evaluar la prueba uniforme.

Esto corresponde a un modelo con una **incertidumbre** baja contra la que debería tener.

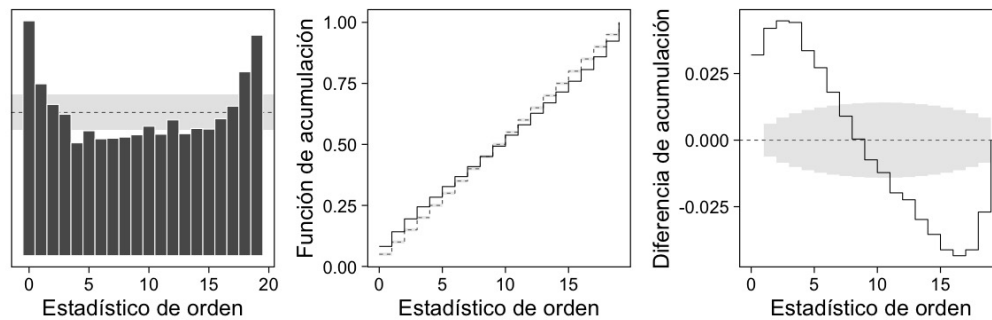


FIGURA 4. Gráficos de comparación uniforme cuando la implementación está sub-dispersa.

Cuando la implementación es de un modelo sobre-disperso tenemos un comportamiento en forma de \cap como se muestra en la Fig. 5. Esto corresponde a un modelo con una **incertidumbre** mayor a la que debería corresponder.

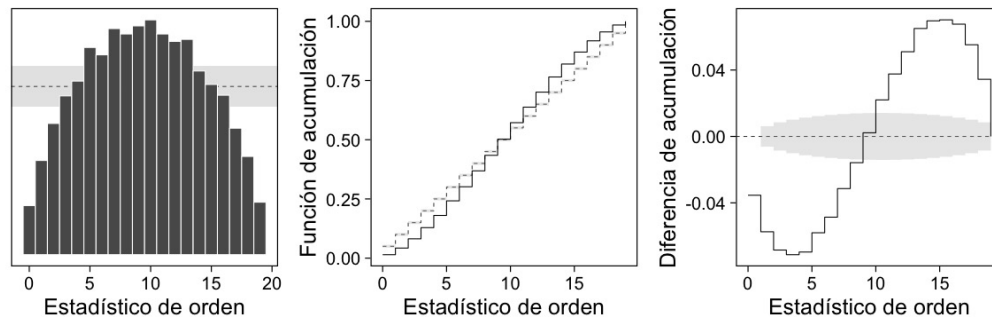


FIGURA 5. Gráficos de comparación uniforme cuando la implementación está sobre-dispersa.

Cuando la implementación es de un modelo con sesgo a la derecha tenemos un comportamiento como se muestra en la Fig. 6. Esto corresponde a un modelo que está **sobre-estimando** los resultados que debería tener.

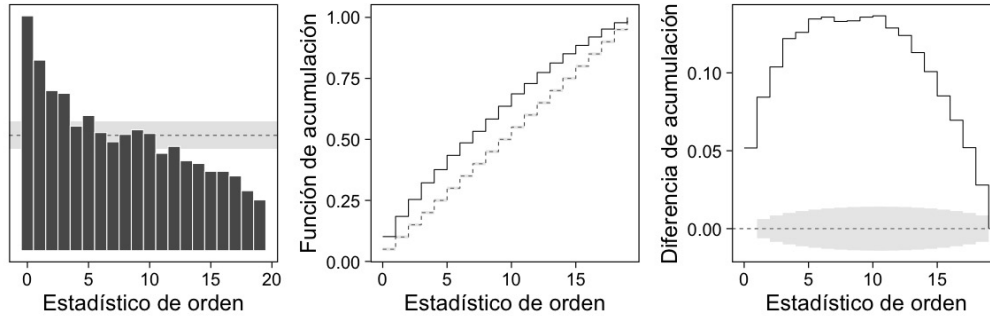


FIGURA 6. Gráficos de comparación uniforme cuando la implementación tiene un sesgo a la derecha.

El caso contrario (sesgo a la izquierda) representa un modelo que está **sub-estimando** las probabilidades.

3.3. Pruebas de uniformidad

Una manera de poder efectuar una prueba es considerar una χ^2 y verificar que los conteos en las cubetas corresponden, en promedio, a lo que esperaríamos con ordenes uniformes.

El estadístico de prueba sería

$$\hat{\chi}^2 = \sum_{m=1}^M \frac{(b_m - e_m)^2}{e_m}, \quad (14)$$

donde b_m denota el número de réplicas en la cubeta m ésima y e_m denota el número de réplicas que esperaríamos caigan en dicha cubeta.

La prueba radica en que los términos de la suma son potencias cuadradas de una normal estándar y por lo tanto

$$\hat{\chi}^2 \sim \chi_{M-1}^2, \quad (15)$$

de la cual podemos evaluar una prueba de hipótesis.

Nota la prueba de hipótesis definida anteriormente no tiene una potencia alta.

4. CBS EN STAN

La idea, como hemos mencionado antes, es poner a prueba si nuestra implementación de un modelo es la adecuada. Estas pruebas no están diseñadas para verificar que nuestro modelo es el adecuado.

Usaremos **Stan** para:

1. Simular datos.
2. Ajustar la distribución posterior.
3. Calcular los estadísticos de orden.

Esto implicará que tenemos que correr nuestro simulador varias veces para poder producir un histograma de estadísticos de orden que esperamos tenga una distribución de muestreo uniforme dentro de los rangos.

4.1. Implementación en Stan

Podemos utilizar un bloque **transformed data** para simular parámetros y datos para el modelo. Regresando a nuestro modelo Normal-Normal, tenemos un bloque que genera parámetros simulados.

```

1 transformed data {
2   real mu_sim = normal_rng(0, 1);
3   real y_sim = normal_rng(mu_sim, sqrt(2));
4 }

```

Adicional, podemos utilizar un bloque `generated quantities` para calcular las indicadores y los estadísticos de orden

```

1 generated quantities {
2   int<lower=0, upper=1> lt_sim = { mu < mu_sim };
3 }

```

4.2. Consideración para métodos de MCMC

Utilizar técnicas de MCMC nos permite simular de la distribución objetivo. Esperaríamos que las muestras sean lo más cercanas a ser independientes. El diagnóstico N_{eff} nos puede dar una indicación de con cuántas muestras nos podemos quedar para realizar los histogramas.

4.3. Ejemplo

Regresaremos a nuestro ejemplo de las escuelas. Sabemos que el modelo puede tener problemas si no está bien parametrizado. Realizaremos un estudio numérico con $N = 500$ réplicas del proceso. En cada una simulamos de tal forma que adelgazamos la cadena de Markov cada 10 iteraciones. El número total de simulaciones se fija para recuperar $M = 100$ ordenes posibles. Los gráficos muestran histogramas con 20 cubetas.

Nota que [2] proponen un algoritmo para poder aplicar SBC a muestras de un cadena de Markov. Dicha propuesta esta basada en estar revisando, por réplica, el número efectivo de simulaciones para poder generar una muestra que pueda ser adelgazada después. Sin embargo, el problema de las escuelas está tan bien identificado y sabemos que nuestra implementación del modelo será deficiente, que no será necesario pedir cadenas tan estables.

```

1 transformed data {
2   real mu_sim = normal_rng(0, 5);
3   real tau_sim = fabs(normal_rng(0, 5));
4   int<lower=0> J = 8;
5   array[J] real theta_sim = normal_rng(rep_vector(mu_sim, J), tau_sim);
6   array[J] real<lower=0> sigma = fabs(normal_rng(rep_vector(0, J), 5));
7   array[J] real y = normal_rng(theta_sim, sigma);
8 }
9 parameters {
10  real mu;
11  real<lower=0> tau;
12  array[J] real theta;
13 }
14 model {
15  mu ~ normal(0, 5);
16  tau ~ normal(0, 5);
17  theta ~ normal(mu, tau);
18  y ~ normal(theta, sigma);
19 }
20 generated quantities {
21  int<lower=0, upper=1> mu_lt_sim = mu < mu_sim;
22  int<lower=0, upper=1> tau_lt_sim = tau < tau_sim;

```

```

23   int<lower=0, upper=1> theta1_lt_sim = theta[1] < theta_sim[1];
24 }

```

Nota que el bloque de `transformed data` escribe el proceso generador de los datos. Primero, simulamos los parámetros poblacionales (μ, τ) ; después, los datos (y_j, σ_j) .

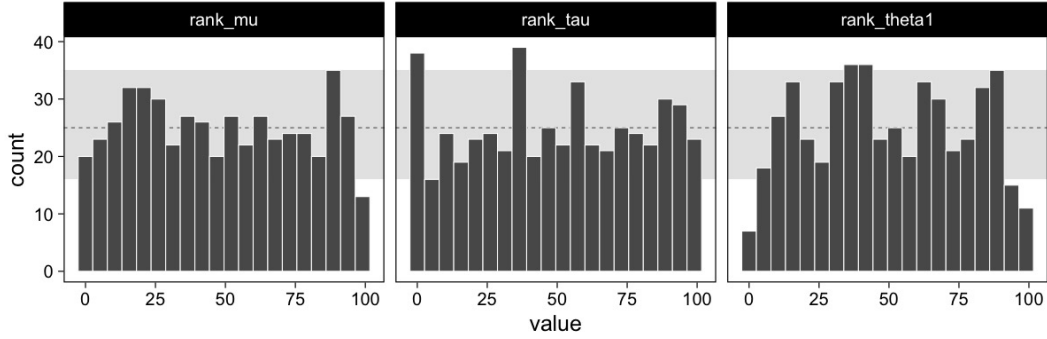


FIGURA 7. Contraste de histogramas contra la distribución uniforme.

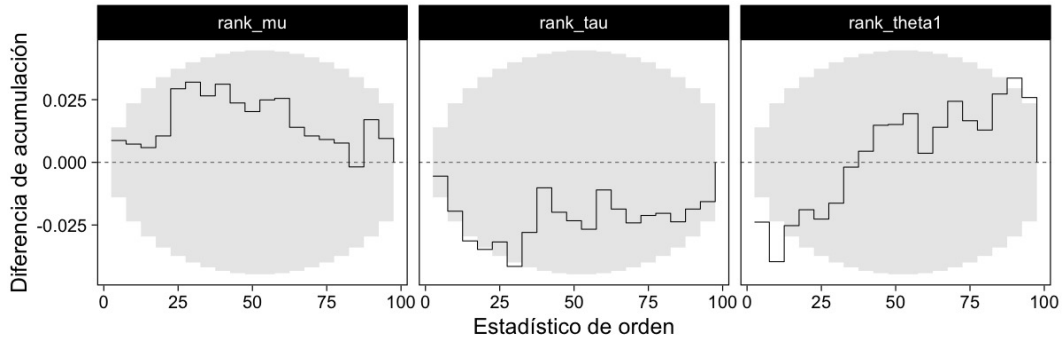


FIGURA 8. Diferencia entre la ECDF y la CDF bajo un modelo uniforme de los estadísticos de orden.

5. CASO PRÁCTICO

Consideraremos un modelo de mezclas

$$\pi(y|\theta, w) = \sum_{k=1}^K w_k \pi_k(y|\theta_k), \quad (16)$$

donde $\sum_k w_k = 1$, θ es un vector de parámetros por bloques, y las densidades π_k pueden pertenecer a la misma familia.

En este caso consideraremos dos componentes $K = 2$, $\theta = (\mu_1, \mu_2)^\top$ y π_k la función de masa de probabilidad de una Poisson con media μ_k .

El modelo que escrito en `Stan` como sigue. Nota que dejaremos en un ciclo externo la simulación de datos sintéticos, por lo tanto, no utilizaremos el bloque de `generated quantities`. Todo el procesamiento lo haremos fuera de `Stan`.

```

1 data {
2   int<lower=0> N;
3   int y[N];

```



```

4 }
5
6 parameters {
7   real mu1;
8   real mu2;
9   real<lower=0, upper=1> omega;
10 }
11
12 model {
13   target += log_mix(omega, poisson_log_lpmf(y | mu1), poisson_log_lpmf(y | mu2
14   ));
15   target += normal_lpdf(mu1 | 3, 1);
16   target += normal_lpdf(mu2 | 3, 1);
17 }

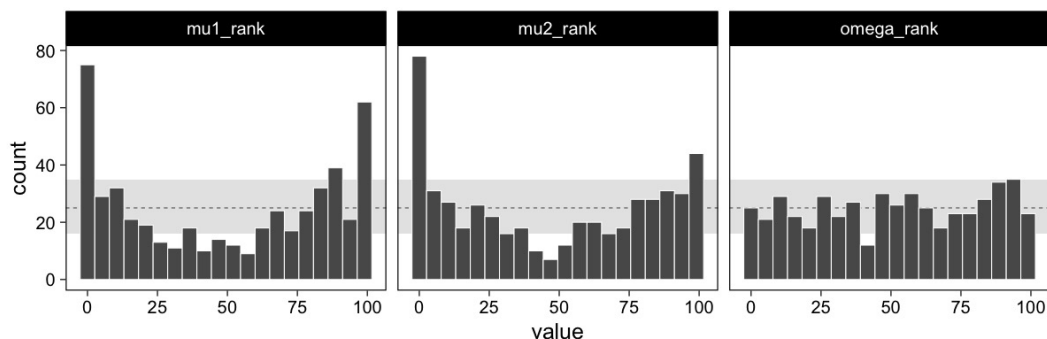
```

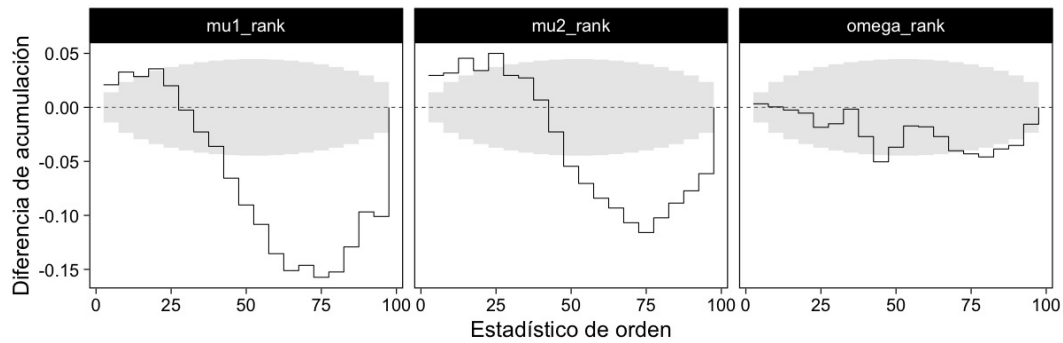
```

1 generate_poisson_mix ← function(N){
2   ## Generamos parametros simulados
3   mu1 ← rnorm(1, 3, 1)
4   mu2 ← rnorm(1, 3, 1)
5   omega ← runif(1)
6   ## Generamos datos sinteticos
7   y ← numeric(N)
8   for(n in 1:N){
9     if(runif(1) < omega){
10      y[n] ← rpois(1, exp(mu1))
11    } else {
12      y[n] ← rpois(1, exp(mu2))
13    }
14  }
15  ## Regresamos en lista
16  sim ← within(list(), {
17    mu ← c(mu1, mu2)
18    omega ← omega
19  })
20  obs ← list(N = N, y = y)
21  list(sim = sim, obs = obs)
22 }

```

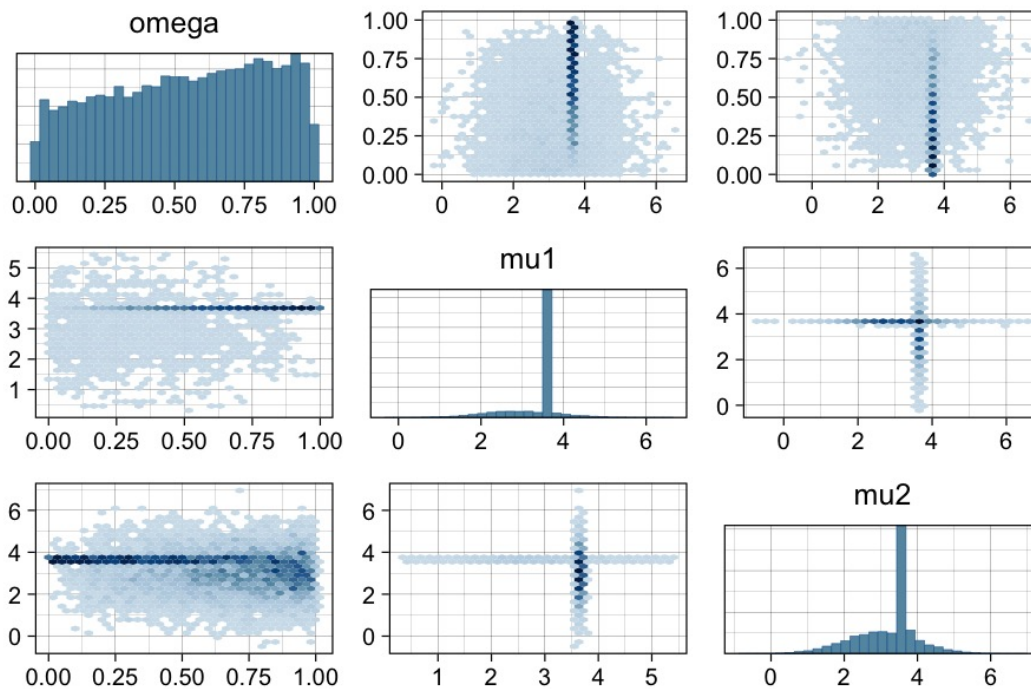
El modelo tiene un poco de problemas en correr. Por ejemplo, algunas simulaciones tienen un número efectivo de simulaciones mucho menores de las que corremos (alrededor del 10%). Así que hace sentido adelgazar la cadena para mitigar los efectos de correlación en los gráficos de diagnóstico.





Los resultados nos muestran histogramas que corresponden a un modelo sobre-disperso. Lo cual es consecuencia de un modelo posterior con mucho mayor incertidumbre de la que esperaríamos.

Por supuesto, esto lo pudimos haber diagnosticado observando una simulación de la posterior. Sin embargo, bajo este enfoque (estudiar una sola simulación) siempre puede quedar la duda si lo que observamos es artefacto de una simulación (por ejemplo de fijar una semilla) o es un comportamiento generalizado.



5.1. Re-implementando

```

1 data {
2   int<lower=0> N;
3   int y[N];
4 }
5
6 parameters {
7   real mu1;
8   real mu2;
9   real<lower=0, upper=1> omega;
10 }
11

```

```

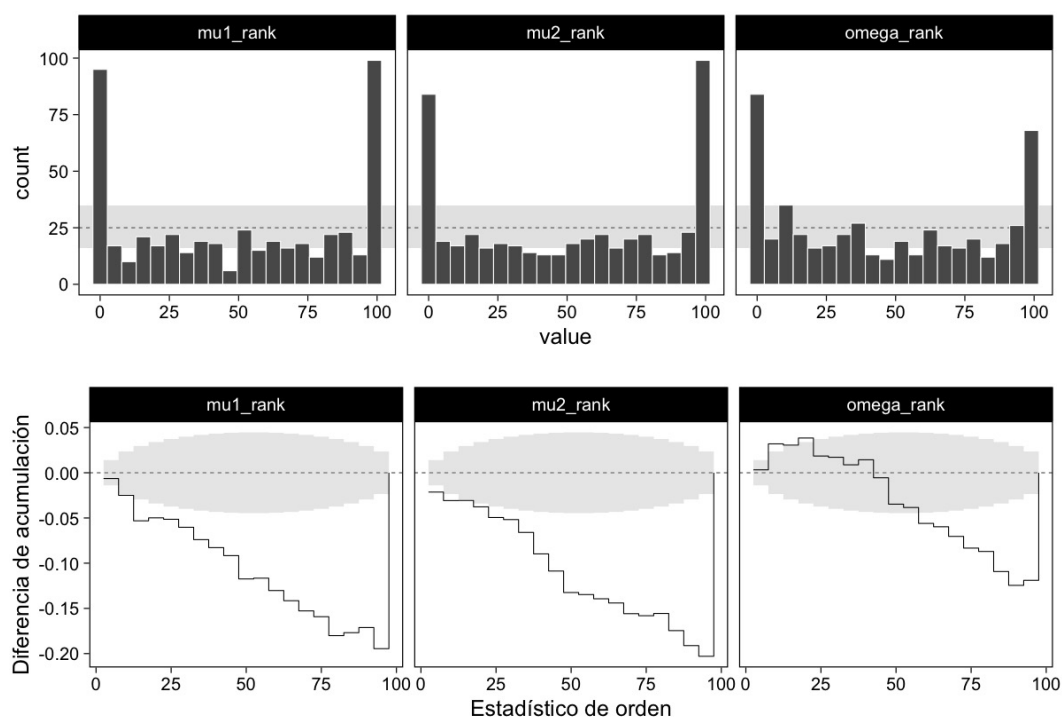
12 model {
13   for(n in 1:N) {
14     target += log_mix(omega,
15                       poisson_log_lpmf(y[n] | mu1),
16                       poisson_log_lpmf(y[n] | mu2));
17   }
18   target += normal_lpdf(mu1 | 3, 1);
19   target += normal_lpdf(mu2 | 3, 1);
20 }

```

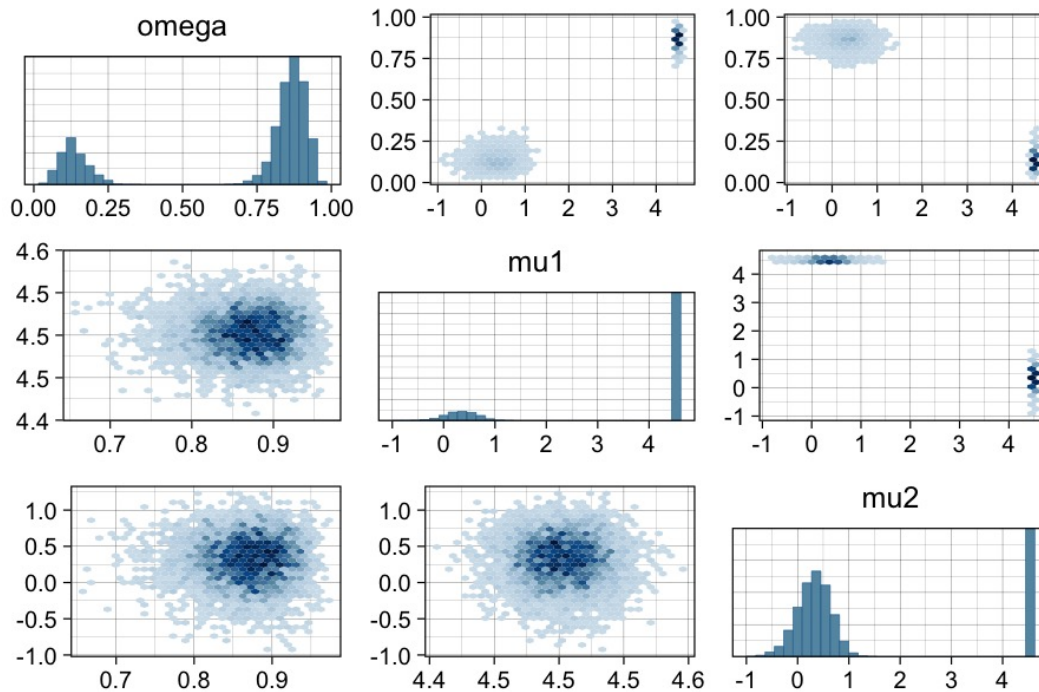
```

1 simulaciones <- tibble(id = 1:500) >
2   mutate(results = map(id, replicate_experiment, modelo))

```



Aun tenemos un modelo con muy poca incertidumbre. Al parecer hay todavía algo que no está bien en la implementación.



5.2. Arreglando problemas de identificabilidad

```

1 data {
2   int<lower=0> N;
3   int y[N];
4 }
5
6 parameters {
7   ordered[2] mu;
8   real<lower=0, upper=1> omega;
9 }
10
11 model {
12   for(n in 1:N) {
13     target += log_mix(omega,
14                       poisson_log_lpmf(y[n] | mu[1]),
15                       poisson_log_lpmf(y[n] | mu[2]));
16   }
17   target += normal_lpdf(mu | 3, 1);
18 }

```

```

1 generate_poisson_mix_ordered <- function(N){
2   ## Generamos parametros simulados
3   mu <- sort(rnorm(2, 3, 1))
4   omega <- runif(1)
5   ## Generamos datos sinteticos
6   y <- numeric(N)
7   for(n in 1:N){
8     if(runif(1) < omega){
9       y[n] <- rpois(1, exp(mu[1]))
10    } else {
11      y[n] <- rpois(1, exp(mu[2]))
12    }
13  }
14 }

```

```

13 }
14 ## Regresamos en lista
15 sim <- within(list(), {
16     mu <- mu
17     omega <- omega
18 })
19 obs <- list(N = N, y = y)
20 list(sim = sim, obs = obs)
21 }

```

```

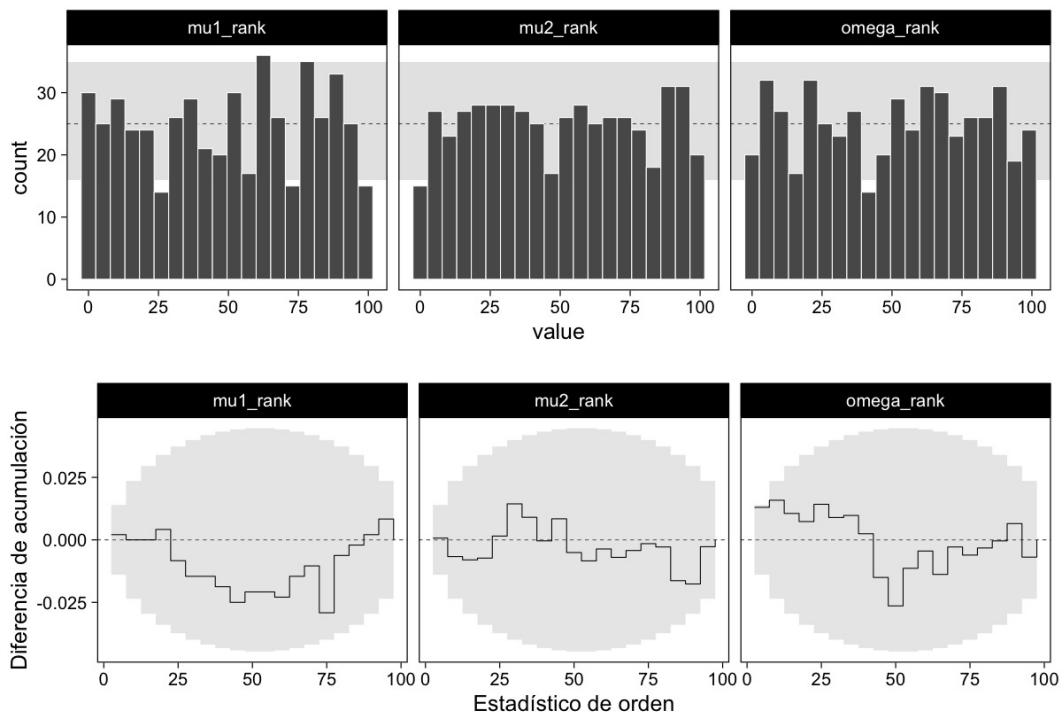
1 replicate_experiment_ordered <- function(id, modelo){
2   data <- generate_poisson_mix_ordered(50)
3   posterior <- modelo$sample(data$obs, chains = 1, refresh = 1000,
4     iter_sampling = 990, thin = 10)
5
6   posterior$draws(format = "df") >
7   as_tibble() >
8   mutate(
9     mu1_bool = 'mu[1]' < data$sim$mu[1],
10    mu2_bool = 'mu[2]' < data$sim$mu[2],
11    omega_bool = omega < data$sim$omega) >
12   summarise(
13     mu1_rank = sum(mu1_bool),
14     mu2_rank = sum(mu2_bool),
15     omega_rank = sum(omega_bool),
16   )
17 }

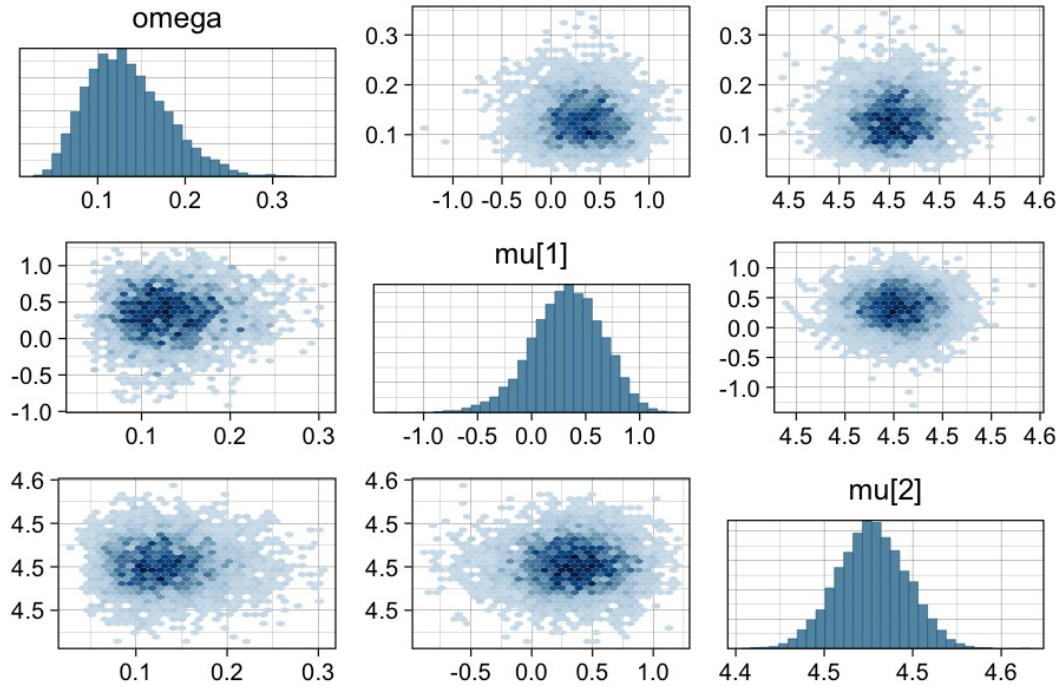
```

```

1 simulaciones <- tibble(id = 1:500) >
2   mutate(results = map(id, replicate_experiment_ordered, modelo))

```





6. CONCLUSIONES

En esta sección mostramos un mecanismo para identificar distribuciones bien calibradas. El mecanismo aprovecha que por definición hacer inferencia Bayesiana es un procedimiento bien calibrado. Es decir, siempre y cuando los datos sean generados por el modelo probabilístico nuestra cobertura de intervalos será igual a la nominal.

Existen alternativas para evaluar métodos de muestreo. Sin embargo, estos mecanismos son utilizados cuando hacemos alguna inferencia aproximada. Es decir, cuando estamos dispuestos a hacer una aproximación de la verosimilitud (usualmente el componente mas costoso) o de la posterior misma (que veremos rumbo al final del curso).

En el mismo espíritu de diagnósticos de MCMC, SBC es un mecanismo para evaluar y criticar la implementación de un modelo. No nos dice qué modelo tiene sentido bajo un conjunto de datos. Esto es, justo lo que estudiaremos en la sección siguiente.

REFERENCIAS

- [1] S. R. Cook, A. Gelman, and D. B. Rubin. Validation of Software for Bayesian Models Using Posterior Quantiles. *Journal of Computational and Graphical Statistics*, 15(3):675–692, sep 2006. ISSN 1061-8600, 1537-2715. . [1](#), [2](#)
- [2] S. Talts, M. Betancourt, D. Simpson, A. Vehtari, and A. Gelman. Validating Bayesian Inference Algorithms with Simulation-Based Calibration. *arXiv:1804.06788 [stat]*, oct 2020. [1](#), [2](#), [3](#), [7](#)