Meta Análisis con inferencia bayesiana

Paula Vargas Pellicer

26/04/2022

Meta análisis

Queremos saber si las aves migratorias llegan a sus zonas de reproducción más o menos en la misma época del año que en décadas pasadas, o si ahora llegan antes o después como resultado del cambio climático. Hemos recopilado datos de muchos estudios publicados diferentes que informan la cantidad de días antes o después de que llegan las aves en días/año y días/grado Celsius. Este conjunto de metadatos global incluye datos de muchas especies, países, latitudes, etc.

Usando un meta análisis, podemos calcular la diferencia promedio de días en todas las poblaciones para las que tenemos datos utilizando el intercepto como nuestro único efecto fijo. Pero asumiríamos que habrá una gran cantidad de variación en torno a esta respuesta promedio, y queremos averiguar si podemos encontrar algún patrón en ella, tal vez entre especies, ubicaciones o rasgos de historia de vida.

Incluir especies como un efecto aleatorio nos dirá cuánta variación hay entre especies. También estimará una respuesta promedio para cada una; sin embargo, generalmente es más informativo reportar la variación entre especies, en lugar del efecto de cada especie por separado.

```
library("MCMCglmm")

library("dplyr")

migracion <- read.csv("~/Downloads/migration_metadata.csv", header = T)


# Échale un ojo a la base de datos y a su estructura
```

Primero vamos a ver la variable de tiempo, que es la que nos interesa, si quieres luego puedes revisar la de temperatura.

```
migracion %>%
filter(Predictor == "year") -> migraciontiempo
```

Antes de comenzar, grafiquemos los datos. Un gráfico de embudo se usa típicamente para visualizar datos para metaanálisis. Esto se hace trazando la variable predictora frente a 1/error estándar para cada punto de datos. Esto pondera cada estudio en la gráfica por su precisión, dando finalmente menos peso a los estudios con alto error estándar. En este caso, Slope es el cambio de fecha de llegada en días/año.

```
plot(migraciontiempo$Slope, I(1/migraciontiempo$SE))
```

Podemos hacer un "zoom"

```
plot(migraciontiempo$Slope, I(1/migraciontiempo$SE), xlim = c(-2,2), ylim = c
(0, 60))
```

Ahora, podemos ver con más detalle que el valor real parece canalizarse justo a la izquierda de cero, y hay bastante variación en torno a esto.

Ahora ejecutaremos un modelo de efectos aleatorios, con solo la intersección como efecto fijo. El intercepto va a estimar el cambio promedio en la fecha de llegada en todos los puntos de datos. Hay muchas especies, ubicaciones y estudios diferentes en este análisis, por lo que los efectos aleatorios estimarán si existe una verdadera variación alrededor de la intercepción y cuánto de ella puede explicarse por un efecto de especie, ubicación o estudio.

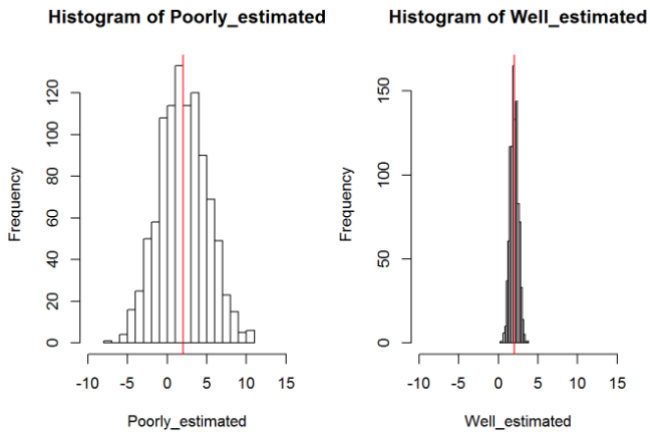
```
randomtest <- MCMCglmm(Slope ~ 1, random = ~Species + Location + Study, data
= migraciontiempo)

summary(randomtest)
```

Ahora tenemos una distribución de parámetros estimados porque MCMCglmm ha ejecutado 13 000 iteraciones del modelo y ha muestreado 1000 de ellas para proporcionar una distribución posterior.

El resumen nos muestra una media posterior para cada efecto, Intervalos Creíbles del 95% superior e inferior (no son Intervalos de Confianza) de la distribución, tamaño de muestra efectivo y para los efectos fijos, un valor de pMCMC.

Revisando la significancia

Podemos aceptar que un efecto fijo es significativo cuando los intervalos creíbles no se extienden al cero, esto se debe a que, si la distribución posterior se extiende en el cero, no podemos estar seguros de que no sea cero. Idealmente, la distribución posterior también tiene que ser estrecha, lo que indica que el valor de ese parámetro se conoce con precisión.



Con los efectos aleatorios, estimamos la varianza. Como la varianza no puede ser cero ni negativa, aceptamos que un efecto aleatorio es significativo cuando la distribución de la varianza no se eleva contra cero. Para comprobar esto, podemos trazar el histograma de cada distribución posterior.

```
hist(mcmc(randomtest$VCV)[,"Study"])

hist(mcmc(randomtest$VCV)[,"Location"])

hist(mcmc(randomtest$VCV)[,"Species"])
```

Aquí podemos ver que la distribución de la varianza para la Ubicación y la Especie se empalma en el cero. Para que un efecto aleatorio sea significativo, queremos que las colas estén bien alejadas de cero.

Convergencia del modelo

```
plot(randomtest$Sol)
```

Aquí podemos ver la estimación de la traza y la densidad para la intercepción. La traza es como una serie de tiempo de lo que hizo el modelo mientras se ejecutaba y se puede usar para evaluar la mezcla (o la convergencia), mientras que la densidad es como un histograma suavizado de las estimaciones de la distribución posterior que produjo el modelo para cada iteración del modelo

Para asegurarnos de que el modelo haya convergido, el gráfico de seguimiento debe verse como un gusano azotador. Parece que la intercepción se ha mezclado bien.

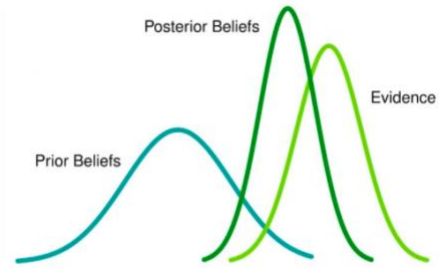
Hagamos lo mismo ahora, pero para las varianzas de los efectos aleatorios. Dependiendo de tu computadora y pantalla, es posible que recibas un mensaje de error que indique que los gráficos son demasiado grandes para mostrarlos; puedes hacer que tu panel de gráficos sea más grande arrastrándolo hacia arriba y hacia la izquierda.

```
plot(randomtest$VCV)
```

Parece que algunas de las variaciones de los efectos aleatorios no se han mezclado muy bien. El tamaño efectivo de la muestra también es muy pequeño. Tal vez podríamos mejorar esto aumentando el número de iteraciones, pero debido a que la cadena parece estar atascada alrededor de cero, parece que tendremos que usar un priori más fuerte que el predeterminado.

Priorizaciones

La parte más difícil de entender de un análisis bayesiano es cómo ajustar las priorizaciones correctas.



Estas son cuantificaciones matemáticas de nuestro conocimiento previo de lo que pensamos que podría ser la media y/o la varianza de un parámetro. Ajustamos un prior separado para cada efecto fijo y aleatorio, y para el residual.

Por lo tanto, podemos usar los datos previos para informar al modelo qué forma creemos que tomará la distribución posterior.

Ejecutemos el modelo nuevamente, pero esta vez usaremos priorizaciones expandidas para los efectos aleatorios al incluir prior = prior1. Cada efecto aleatorio está representado por una G, y el residual está representado por R. La expansión del parámetro se refiere al hecho de que hemos incluido una media previa (alfa.mu) y una matriz de (co)varianza (alfa.V), así como V y nu. Por ahora, alpha.V va a ser 1000,

```
a <- 1000
prior1 <- list(R = list(V = diag(1), nu = 0.002),
               G = list(G1 = list(V = diag(1), nu = 1, alpha.mu = 0, alpha.V
= diag(1)*a),
                        G1 = list(V = diag(1), nu = 1, alpha.mu = 0, alpha.V
= diag(1)*a),
                        G1 = list(V = diag(1), nu = 1, alpha.mu = 0, alpha.V
= diag(1)*a)))

randomprior <- MCMCglmm(Slope ~ 1, random = ~Species + Location + Study,
                        data = migraciontiempo, prior = prior1, nitt = 60000)

summary(randomprior)
```

¡Los tamaños de muestra efectivos son mucho más grandes ahora! Esta es una buena señal.

```
plot(randomprior$VCV)
```

Los modelos parecen haberse mezclado mucho mejor también.

Antes de hacer las comprobaciones de nuestro modelo, quiero controlar el error de muestreo en el modelo. Esta es una de las razones clave por las que usaríamos MCMCglmm para el metaanálisis, en lugar de otro programa o paquete.

La suposición clave de un metaanálisis es que la varianza entre observaciones debida al error de muestreo se puede aproximar como el error estándar al cuadrado. Podemos usar un truco computacional para permitir esto en MCMCglmm ajustando idh(SE):units como un efecto aleatorio y fijando la varianza asociada en 1. Ahora tenemos cuatro priors aleatorios, el último de los cuales está fijo en 1.

```
prior2 <- list(R = list(V = diag(1), nu = 0.002),
               G = list(G1 = list(V = diag(1), nu = 1, alpha.mu = 0, alpha.V
= diag(1)*a),
                        G1 = list(V = diag(1), nu = 1, alpha.mu = 0, alpha.V
= diag(1)*a),
                        G1 = list(V = diag(1), nu = 1, alpha.mu = 0, alpha.V
= diag(1)*a),
                        G1 = list(V = diag(1), fix = 1)))

randomerror2 <- MCMCglmm(Slope ~ 1, random = ~Species + Location + Study + id
h(SE):units,
                         data = migraciontiempo, prior = prior2, nitt = 60000
)

plot(randomerror2$VCV)
```

Si revisas el resumen de salida, puedes ver que ahora que hemos incluido el error de medición, nuestras estimaciones se conservan mucho más. A los estudios con un error estándar más alto se les ha dado un peso estadístico más bajo.

Ahora, realizaremos nuestras comprobaciones de modelo. Para hacer esto, simularemos nuevos datos dados los mismos valores de parámetro (estructuras de varianza/covarianza (priors)), y luego los representaremos con nuestros datos reales para asegurarnos de que se superpongan. Recuerde que, en los análisis bayesianos, los datos permanecen fijos y son los parámetros los que cambian. Entonces, si hemos usado los parámetros correctos, deberíamos poder usarlos para simular nuevos datos que se parecen al original.

```
xsim <- simulate(randomerror2) # corre 100 modelos nuevos, alrededor de la mi
sma estructura de varianza/covarianza pero con datos simulados

plot(migraciontiempo$Slope, I(1/migraciontiempo$SE))
points(xsim, I(1/migraciontiempo$SE), col = "red")
```

Esto parece encajar razonablemente bien, aunque los datos simulados están ligeramente sesgados hacia la izquierda.

El problema es que la varianza de muestreo para las estimaciones de baja precisión es en realidad mayor que el SE^2 (es decir, no se cumple la suposición de un metaanálisis). Esto significa que

a) todavía se le da demasiado peso a los estudios de baja precisión, y

b) parte de la variación biológica (la varianza de las unidades) está sobreestimada y

c) si el sesgo de publicación es más probable entre los estudios de baja precisión, entonces el tamaño medio del efecto puede estar sesgado.

Una solución rápida es ver si la varianza entre observaciones aumenta más con el error estándar informado más rápido de lo que debería. Para hacer esto, podemos estimar la varianza, en lugar de suponer que es 1, y ver si la estimación es mayor que 1.

Volvamos a ejecutar el modelo, pero esta vez cambiando el error de medición anterior para que ya no esté fijo en 1.

```
prior3 <- list(R = list(V = diag(1), nu = 0.002),
               G = list(G1 = list(V = diag(1), nu = 1, alpha.mu = 0, alpha.V
= diag(1)*a),
                        G1 = list(V = diag(1), nu = 1, alpha.mu = 0, alpha.V
= diag(1)*a),
                        G1 = list(V = diag(1), nu = 1, alpha.mu = 0, alpha.V
= diag(1)*a),
                        G1 = list(V = diag(1), nu = 1, alpha.mu = 0, alpha.V
= diag(1)*a)))

randomerror3 <- MCMCglmm(Slope ~ 1, random = ~Species + Location + Study + id
h(SE):units,
                         data = migraciontiempo, prior = prior3, nitt = 60000
)
```

Ahora podemos simular nuevos datos otra vez y trazarlos contra nuestros datos recopilados.

```
xsim <- simulate(randomerror3)

plot(migraciontiempo$Slope, I(1/migraciontiempo$SE))
points(xsim, I(1/migraciontiempo$SE), col = "red")
```

¡Ahora es tu turno!

```
Filtra los datos por filas que tienen la temperatura como predictor
Traza los datos usando un gráfico de embudo
Ejecuta un modelo básico de efectos aleatorios.
Guarda el modo posterior.
Traza VCV (aleatorio) y Sol (fijo) y verificar la auto correlación
Aumenta el número de iteraciones y guarda, verifica tus priors
Haz las comprobaciones de modelo
¡Interpreta tu modelo!
```