EST-46115: Modelación Bayesiana

Profesor: Alfredo Garbuno Iñigo — Primavera, 2022 — Stan.
Objetivo. Un primer acercamiento a `Stan`, sintaxis, manipulación de objetos y visualizaciones sencillas. Además un caso de estudio donde empezaremos a ver ciertos errores en el ajuste de los modelos.
Lectura recomendada: Sección 6.2.1 de [3].

1. INTRODUCCIÓN

Veremos ejemplos sencillos sobre la sintaxis de `Stan` ([1]) con el cual simularemos realizaciones de parámetros `no observables` para los cuales tenemos un `estado de conocimiento` reflejado en la distribución posterior.

Habíamos visto que los *bayesics* son los componentes:

1. Un conjunto de datos.
2. Supuestos del proceso generador de datos.
3. Conocimiento previo sobre los componentes que rigen el modelo.

Para `Stan` tenemos que seguir algo muy similar. Es decir,

1. Definir el modelo.
2. Ingestar los datos.
3. Darle *play* al botón de inferencia.

Un modelo de `Stan` se escribe en un archivo de texto y es una secuencia de bloques con nombre. En general el esqueleto es como sigue:

```
1 print_file("modelos/tutorial/esqueleto.stan")
```

Listing 1. *Función* personalizada *para imprimir un archivo de texto.*

```
functions {
    // ... function declarations and definitions ...
}
data {
    // ... declarations ...
}
transformed data {
    // ... declarations ... statements ...
}
parameters {
    // ... declarations ...
}
transformed parameters {
    // ... declarations ... statements ...
}
model {
    // ... declarations ... statements ...
}
generated quantities {
    // ... declarations ... statements ...
}
```

LISTING 2. *Estructura de un modelo de* ***Stan****.*

En general `todos` los bloques `son opcionales`, y **no es necesario** tener todos para compilar un modelo. Para mas información puedes consultar la guía de Stan.

1.1. Estructura del código

- Un bloque `data`. Por ejemplo, Y el número observado de éxitos en 10 pruebas.
- Un bloque `parameters`. Por ejemplo, θ la tasa de éxito en la realización de las pruebas.
- Un bloque `model`. Por ejemplo, $Y \sim \mathsf{Binomial}(10, \theta)$ y $\theta \sim \mathsf{Beta}(2, 2)$.

El código es:

```
data {
  int<lower = 0, upper = 10> Y;
}
parameters {
  real<lower = 0, upper = 1> theta;
}
model {
  Y ~ binomial(10, theta);
  theta ~ beta(2, 2);
}
```

```
## Modelo Beta-Binomial --------------------------------
modelos_files <- "modelos/compilados/tutorial"
ruta <- file.path("modelos/tutorial/beta-binomial.stan")
modelo <- cmdstan_model(ruta, dir = modelos_files)
```

LISTING 3. *Código necesario para interactuar con* ***Stan*** *desde* `R`*.*

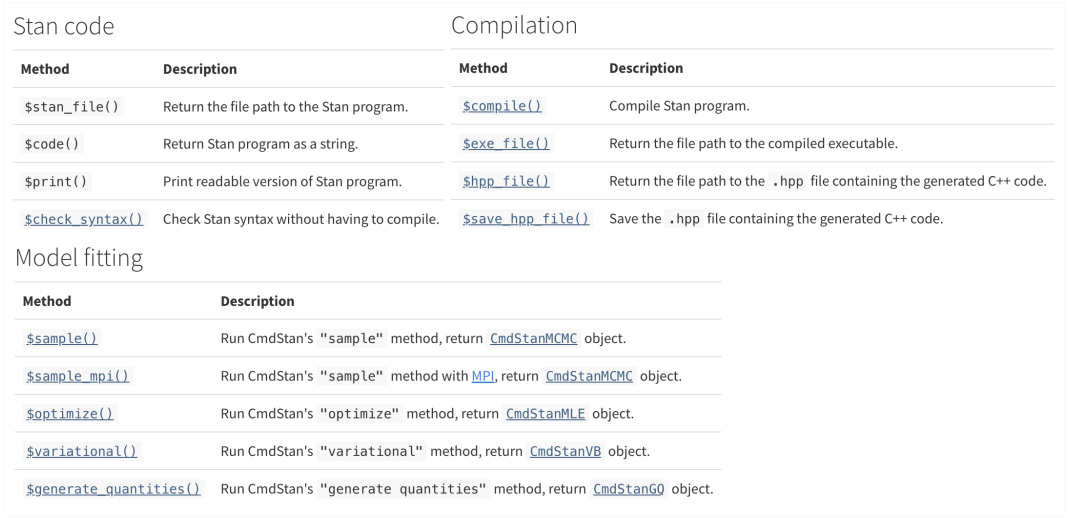
Para leer mas sobre la herramienta y sus interacción desde línea de comandos puedes consultar la documentación de `Stan`.

```
class(modelo)
```

```
[1] "CmdStanModel" "R6"
```

LISTING 4. *Tipo de objeto que regresa la compilación del modelo.*

Con esto tenemos un `objeto` (OOP) con distintos `métodos` que podemos utilizar. Puedes consultar aquí los métodos disponibles de dichos objetos.



Stan code

Method	Description
`$stan_file()`	Return the file path to the Stan program.
`$code()`	Return Stan program as a string.
`$print()`	Print readable version of Stan program.
`$check_syntax()`	Check Stan syntax without having to compile.

Compilation

Method	Description
`$compile()`	Compile Stan program.
`$exe_file()`	Return the file path to the compiled executable.
`$hpp_file()`	Return the file path to the `.hpp` file containing the generated C++ code.
`$save_hpp_file()`	Save the `.hpp` file containing the generated C++ code.

Model fitting

Method	Description
`$sample()`	Run CmdStan's `"sample"` method, return `CmdStanMCMC` object.
`$sample_mpi()`	Run CmdStan's `"sample"` method with MPI, return `CmdStanMCMC` object.
`$optimize()`	Run CmdStan's `"optimize"` method, return `CmdStanMLE` object.
`$variational()`	Run CmdStan's `"variational"` method, return `CmdStanVB` object.
`$generate_quantities()`	Run CmdStan's `"generate quantities"` method, return `CmdStanGQ` object.

FIGURA 1. *Métodos de objetos de la clase* `CmdStanModel`.

Por ejemplo, tenemos un método que puede generar muestras del `modelo probabilístico` que se definió en el bloque de modelo.

Necesitamos los datos en un formato muy especial (una lista):

```
data.list ← list(Y = 7)
```

> La interacción desde `R` con `Stan` necesita los datos ordenados en listas con nombres. En `Python` éstos son diccionarios. Ambos, generalizan a archivos en formato `JSON`.

Para darle *play* :

```
muestras ← modelo$sample(data = data.list,
                         chains = 1,
                         iter=1500,
                         iter_warmup=500,
                         seed=483892929,
                         refresh=700)
```

```
Running MCMC with 1 chain...

Chain 1 Iteration:    1 / 2000 [  0%]  (Warmup)
Chain 1 Iteration:  501 / 2000 [ 25%]  (Sampling)
Chain 1 Iteration: 1200 / 2000 [ 60%]  (Sampling)
```

```
Chain 1 Iteration: 1900 / 2000 [ 95%]  (Sampling)
Chain 1 Iteration: 2000 / 2000 [100%]  (Sampling)
Chain 1 finished in 0.0 seconds.
```

Listing 5. *Resultados de muestreo.*

El resultado es:

```
class(muestras)
```

```
[1] "CmdStanMCMC" "CmdStanFit"  "R6"
```

Listing 6. *Tipo de objeto que regresa la compilación del modelo.*

Donde se pueden explorar los métodos de estos objetos en la documentación.

1.2. Visualizaciones

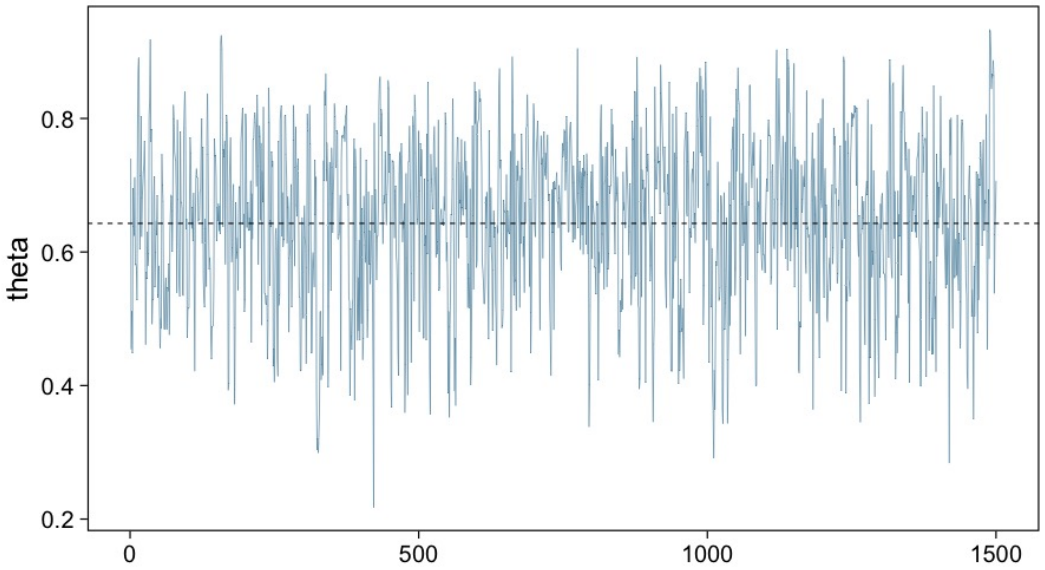


Figura 2. *Trazas (trayectorias) del componente θ en el modelo Beta-Binomial.*

Nota: tuvimos que definir qué parámetros queremos en la visualización. Por *default* incluye un misterioso `_lp` que hace referencia a la evaluación de la log-posterior para cada elemento de la simulación. Adicional, nota (en el código fuente) que la sintaxis para el gráfico utiliza la gramática y las funciones de `ggplot2`.

1.3. Ejercicio (0)

¿Cómo utilizarías `Stan` para generar números aleatorios de:

1. la distribución previa;
2. la distribución predictiva posterior?

Utiliza el ejemplo Beta-Binomial de arriba para ponerlo en práctica. *Hint*: revisa la documentación del bloque `generated quantities`.

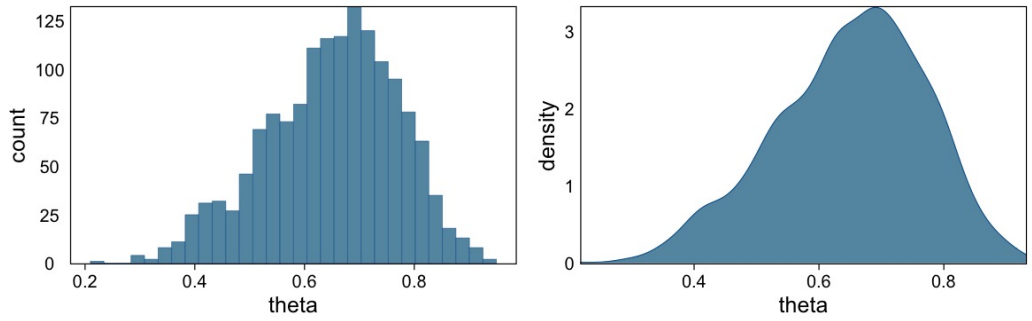


FIGURA 3. *Histogramas del componente θ en el modelo Beta-Binomial.*

1.4. Ejercicio (1)

Repite lo anterior para un modelo Poisson-Gamma. Es decir, para una colección de observaciones $(Y_1, Y_2) = (2, 9)$ donde suponemos que $Y_j \overset{\text{iid}}{\sim} \text{Poisson}(\lambda)$ y $\lambda \sim \text{Exponencial}(3)$.

Hints: Revisa la documentación para definir vectores (en este caso de longitud 2) en el bloque de datos.

1.5. Ejercicio (2)

Utiliza el ambiente de `Stan` para encontrar el estimador de Máxima verosimillitud de los dos problemas que hemos trabajado. Es decir, el caso Beta-Binomial y Poisson-Gamma.

2. CASO: ESCUELAS

Utilizaremos los datos de un estudio de desempeño de 8 escuelas ([2, 4]). Los datos consisten en el puntaje promedio de cada escuela `y` y los errores estándar reportados `sigma` la dispersión de los resultados de dicha prueba.

```
## Caso: escuelas ---------------------------------------
data ← tibble( id = factor(seq(1, 8)),
                y = c(28, 8, -3, 7, -1, 1, 18, 12),
                sigma = c(15, 10, 16, 11, 9, 11, 10, 18))
```

En este caso se utiliza un modelo normal para los resultados de cada escuela

$$y_j \sim \text{N}(\theta_j, \sigma_j), \qquad j = 1, \ldots, J\,,$$

donde $J = 8$, y θ_j representa el promedio de los alumnos de escuela que no observamos pero del cual tenemos un estimador y_j.

Nota que tenemos J valores distintos para θ_j y σ_j. Dado que esperamos que las escuelas provengan de la misma población de escuelas asumimos que

$$\theta_j \sim \text{N}(\mu, \tau), \qquad j = 1, \ldots, J\,,$$

donde μ representa la media poblacional (el promedio en el sistema escolar) y τ la desviación estándar alrededor de este valor.

Representamos nuestra incertidumbre en estos dos valores por medio de

$$\mu \sim \text{N}(0, 5), \qquad \tau \sim \text{Half-Cauchy}(0, 5)\,,$$

lo cual representa información poco precisa de estos valores poblacionales.

3. PRIMER MODELO EN STAN

La forma en que escribimos el modelo en Stan es de manera generativa (*bottom up*):

$$\mu \sim \mathsf{N}(0, 5)\,, \tag{1a}$$
$$\tau \sim \mathsf{Half\text{-}Cauchy}(0, 5)\,, \tag{1b}$$
$$\theta_j \sim \mathsf{N}(\mu, \tau), \qquad j = 1, \ldots, J\,, \tag{1c}$$
$$y_j \sim \mathsf{N}(\theta_j, \sigma_j), \qquad j = 1, \ldots, J\,. \tag{1d}$$

```
print_file("modelos/caso-escuelas/modelo-escuelas.stan")
```

```
data {
  int<lower=0> J;
  real y[J];
  real<lower=0> sigma[J];
}
parameters {
  real mu;
  real<lower=0> tau;
  real theta[J];
}
model {
  mu ~ normal(0, 5);
  tau ~ cauchy(0, 5);
  theta ~ normal(mu, tau);
  y ~ normal(theta, sigma);
}
```

Listing 7. *Código del modelo para el desempeño de las escuelas.*

Nota que `sigma` está definida como *parte del conjunto de datos* que el usuario debe de proveer. Aunque es un parámetro en nuestro modelo (verosimilitud) no está sujeto al proceso de inferencia. Por otro lado, nota que la declaración no se hace de manera componente por componente, sino de forma `vectorizada`.

Una vez escrito nuestro modelo, lo podemos compilar utilizando la librería de `cmdstanr`, que es la interface con `Stan` desde `R`.

```
modelos_files ← "modelos/compilados/caso-escuelas"
ruta ← file.path("modelos/caso-escuelas/modelo-escuelas.stan")
modelo ← cmdstan_model(ruta, dir = modelos_files)
```

Los datos que necesita el bloque `data` se pasan como una *lista con nombres*.

```
data_list ← c(data, J = 8)
```

3.1. Simulación

Contra todas las recomendaciones usuales, corramos sólo una cadena corta:

```
muestras ← modelo$sample(data = data_list,
                         chains = 1,
                         iter=700,
                         iter_warmup=500,
                         seed=483892929,
                         refresh=1200)
```

```
Running MCMC with 1 chain...

Chain 1 Iteration:    1 / 1200 [  0%]  (Warmup)
Chain 1 Iteration:  501 / 1200 [ 41%]  (Sampling)
Chain 1 Iteration: 1200 / 1200 [100%]  (Sampling)
Chain 1 finished in 0.1 seconds.

Warning: 53 of 700 (8.0%) transitions ended with a divergence.
This may indicate insufficient exploration of the posterior distribution.
Possible remedies include:
  * Increasing adapt_delta closer to 1 (default is 0.8)
  * Reparameterizing the model (e.g. using a non-centered parameterization)
  * Using informative or weakly informative prior distributions
```

Listing 8. *Resultados del muestreador en el modelo.*

El muestreador en automático nos regresa ciertas alertas las cuales podemos inspeccionar más a fondo con el siguiente comando:

```
muestras$cmdstan_diagnose()
```

```
Processing csv files: /var/folders/lk/4hdvzkhx269df8zc5xmkqgwr0000gn/T/
    RtmpQgQ2MW/modelo-escuelas-202202241738-1-1c22d2.csv

Checking sampler transitions treedepth.
Treedepth satisfactory for all transitions.

Checking sampler transitions for divergences.
53 of 700 (7.6%) transitions ended with a divergence.
These divergent transitions indicate that HMC is not fully able to explore the
     posterior distribution.
Try increasing adapt delta closer to 1.
If this doesn't␣remove␣all␣divergences,␣try␣to␣reparameterize␣the␣model.

Checking␣E-BFMI␣-␣sampler␣transitions␣HMC␣potential␣energy.
The␣E-BFMI,␣0.16,␣is␣below␣the␣nominal␣threshold␣of␣0.3␣which␣suggests␣that␣
    HMC␣may␣have␣trouble␣exploring␣the␣target␣distribution.
If␣possible,␣try␣to␣reparameterize␣the␣model.

Effective␣sample␣size␣satisfactory.

The␣following␣parameters␣had␣split␣R-hat␣greater␣than␣1.1:
␣␣tau,␣theta[1],␣theta[7]
Such␣high␣values␣indicate␣incomplete␣mixing␣and␣biased␣estimation.
You␣should␣consider␣regularizating␣your␣model␣with␣additional␣prior␣
    information␣or␣a␣more␣effective␣parameterization.

Processing␣complete.
```

Listing 9. *Diagnósticos y resumen.*

Notamos que parece ser que tenemos varias transiciones divergentes, algunos parámetros tienen una $\hat{R}$ tienen un valor que excede la referencia de 1.1 (lo veremos más adelante), y parece ser que los estadisticos de energía también presentan problemas.

Podemos inspeccionar el resultado de las simulaciones utilizando:

```
muestras$cmdstan_summary()
```

```
Inference for Stan model: modelo_escuelas_model
1 chains: each with iter=(700); warmup=(0); thin=(1); 700 iterations saved.

Warmup took 0.029 seconds
Sampling took 0.042 seconds

                  Mean     MCSE   StdDev        5%    50%    95%    N_Eff  N_Eff
                     /s    R_hat

lp__               -12      2.0      8.0       -25    -12   0.36       16
    391      1.1
accept_stat__     0.76  1.1e-01  3.7e-01  4.6e-16   0.98   1.00  1.1e+01  2.5e
    +02  1.1e+00
stepsize__       0.086      nan  2.8e-17  8.6e-02  0.086  0.086      nan
    nan      nan
treedepth__        3.9  4.1e-01  1.5e+00  1.0e+00    4.0    6.0  1.3e+01  3.1e
    +02  1.1e+00
n_leapfrog__        28  4.2e+00  2.3e+01  3.0e+00     19     63  3.0e+01  7.1e
    +02  1.1e+00
divergent__      0.076  6.0e-02  2.6e-01  0.0e+00   0.00    1.0  1.9e+01  4.6e
    +02  1.1e+00
energy__            17  2.0e+00  8.5e+00  4.0e+00     17     30  1.7e+01  4.2e
    +02  1.1e+00

mu                 4.0     0.47      3.5      -1.7    3.4    9.7       55
    1313      1.0
tau                2.9     0.55      3.0      0.32    1.7    8.9       30
    704      1.1
theta[1]           5.4     0.60      5.1      -1.6    4.0     15       74
    1759      1.1
theta[2]           4.4     0.56      4.8      -2.6    3.4     12       72
    1713      1.0
theta[3]           3.4     0.47      5.4      -5.1    3.3     11      130
    3100      1.0
theta[4]           4.1     0.54      4.9      -3.6    3.4     12       82
    1960      1.0
theta[5]           3.5     0.46      4.4      -4.1    3.2     11       92
    2194      1.0
theta[6]           3.7     0.49      4.8      -4.7    3.6     11       99
    2351     1.00
theta[7]           5.4     0.59      4.9      -1.2    4.2     14       68
    1624      1.1
theta[8]           4.5     0.53      4.9      -3.0    3.6     12       85
    2023      1.0

Samples were drawn using hmc with nuts.
For each parameter, N_Eff is a crude measure of effective sample size,
and R_hat is the potential scale reduction factor on split chains (at
convergence, R_hat=1).
```

Listing 10. *Resumen utilizando los métodos de* `CmdStanModel`.

Donde además de los resúmenes usuales para nuestros parámetros de interés encontramos resúmenes internos del simulador (los veremos mas adelante).

3.2. Alternativas: Rstan

Podemos utilizar las funciones de RStan (otra interfase con Stan desde R) para visualizar los resúmenes de manera alternativa.

```
stanfit ← rstan::read_stan_csv(muestras$output_files())
stanfit
```

```
Inference for Stan model: modelo-escuelas-202202241738-1-1c22d2.
1 chains, each with iter=1200; warmup=500; thin=1;
post-warmup draws per chain=700, total post-warmup draws=700.

          mean se_mean  sd   2.5%    25%   50%  75% 97.5% n_eff Rhat
mu         4.0     0.47 3.5  -2.42   1.66   3.4  6.6  11.1    55  1.0
tau        2.9     0.55 3.0   0.32   0.59   1.6  4.3  11.1    29  1.1
theta[1]   5.4     0.60 5.1  -3.50   2.50   4.0  8.4  17.2    73  1.1
theta[2]   4.4     0.57 4.8  -3.99   1.62   3.4  7.5  14.3    71  1.0
theta[3]   3.4     0.48 5.4  -8.36   0.83   3.3  6.7  14.5   129  1.0
theta[4]   4.1     0.54 4.9  -5.79   1.39   3.4  7.3  13.6    82  1.0
theta[5]   3.5     0.46 4.4  -6.08   1.16   3.2  6.6  11.8    91  1.0
theta[6]   3.7     0.49 4.8  -6.97   1.04   3.6  7.0  12.7    98  1.0
theta[7]   5.4     0.59 4.9  -2.64   2.65   4.1  8.1  16.7    67  1.1
theta[8]   4.5     0.53 4.9  -4.63   1.84   3.6  7.6  14.5    84  1.0
lp__     -11.6     2.01 8.0 -25.98 -18.30 -11.9 -3.8   1.4    16  1.1

Samples were drawn using NUTS(diag_e) at Thu Feb 24 17:38:35 2022.
For each parameter, n_eff is a crude measure of effective sample size,
and Rhat is the potential scale reduction factor on split chains (at
convergence, Rhat=1).
```

Listing 11. *Resumen obtenido con librería de Rstan.*

En caso de necesitarlo podemos extraer las muestras en una tabla para poder procesarlas y generar visualizaciones. Por ejemplo, un gráfico de traza con τ que es el parámetro donde más problemas parecemos tener.

Claramente no podemos afirmar que el muestreador está explorando bien la posterior. Hay correlaciones muy altas. Si usáramos la media acumulada no seríamos capaces de diagnosticar estos problemas.

Utilizar gráficos de dispersión bivariados nos ayuda a identificar mejor el problema. En color salmón apuntamos las muestras con transiciones *divergentes* (mas adelante lo explicaremos).

Otra visualización muy conocida es la de coordenadas paralelas. En este tipo de gráficos podemos observar de manera simultánea ciertos patrones en todos los componentes.

Y por último, también podemos explorar la autocorrelación de la cadena.

3.3. Generando mas simulaciones

Hasta ahora los resultados parecen no ser buenos. Tenemos muestras con transiciones *divergentes* y una *correlación muy alta* entre las muestras. Podríamos aumentar el número de simulaciones con la esperanza que esto permita una mejor exploracion de la posterior:

```
muestras ← modelo$sample(data        = data_list,
                         chains      = 1,
                         iter        = 5000,
                         iter_warmup = 5000,
                         seed        = 483892929,
                         refresh     = 10000)
```

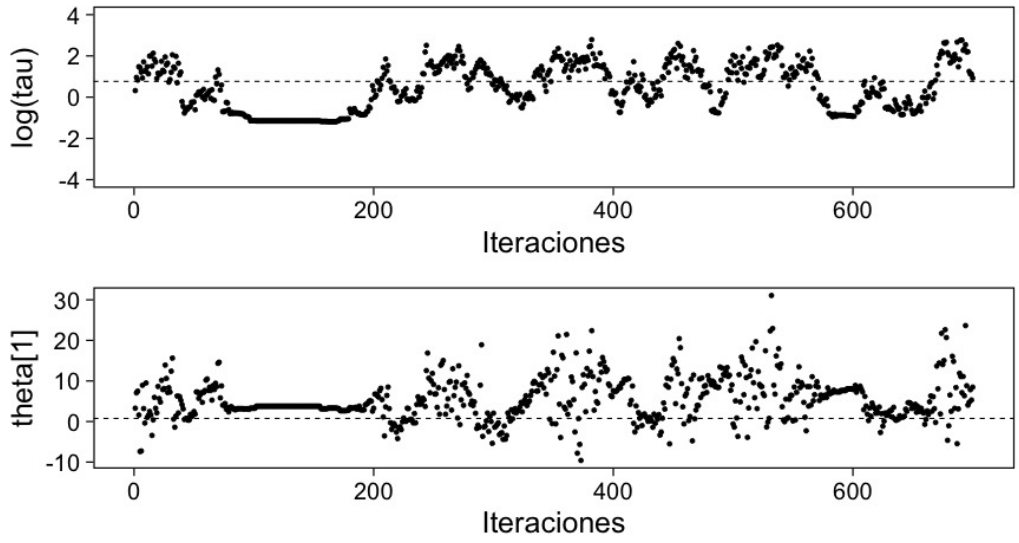


FIGURA 4. *Trayectorías de las muestras del modelo para los componentes* $\log \tau$ *y* θ_1.

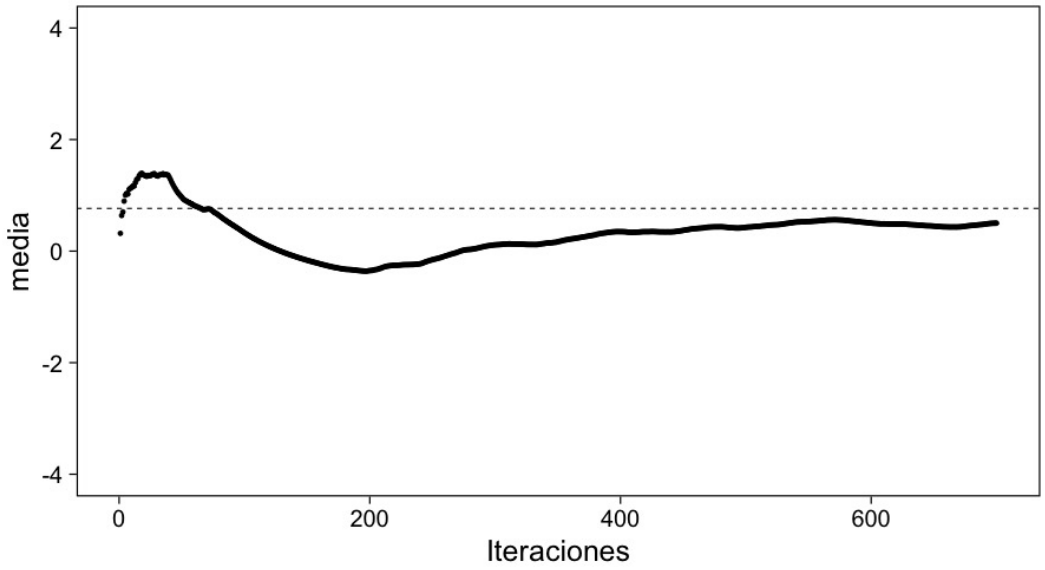


FIGURA 5. *Media acumulada de* $\log \tau$.

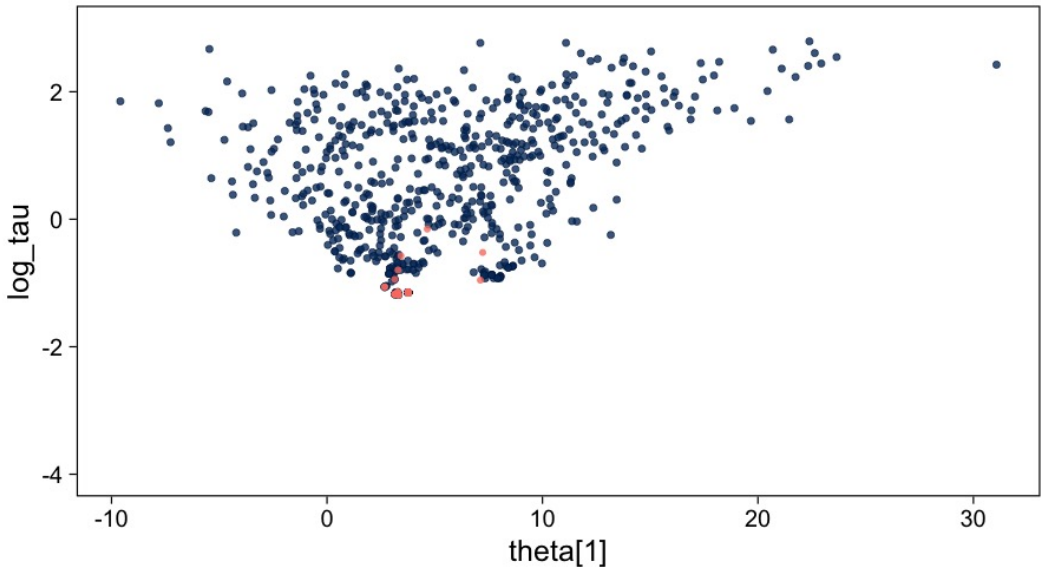


FIGURA 6. *Gráfico de dispersión. Muestras en color salmón representan simulaciones* problemáticas.

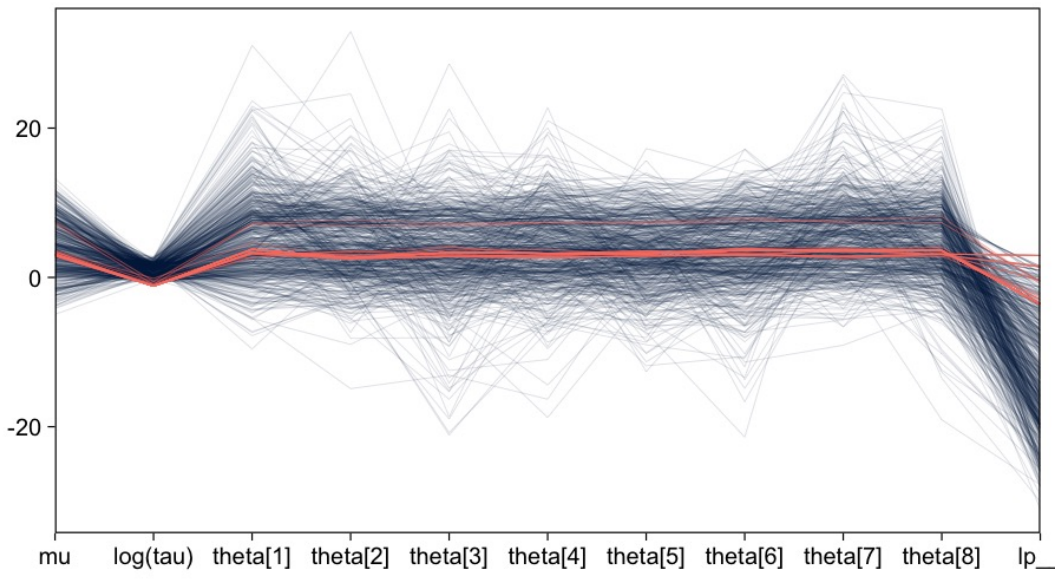


FIGURA 7. *Gráfico de coordenadas paralelas. Permiten* conectar *los distintos componentes de un vector. Color salmón representa simulaciones* problemáticas.

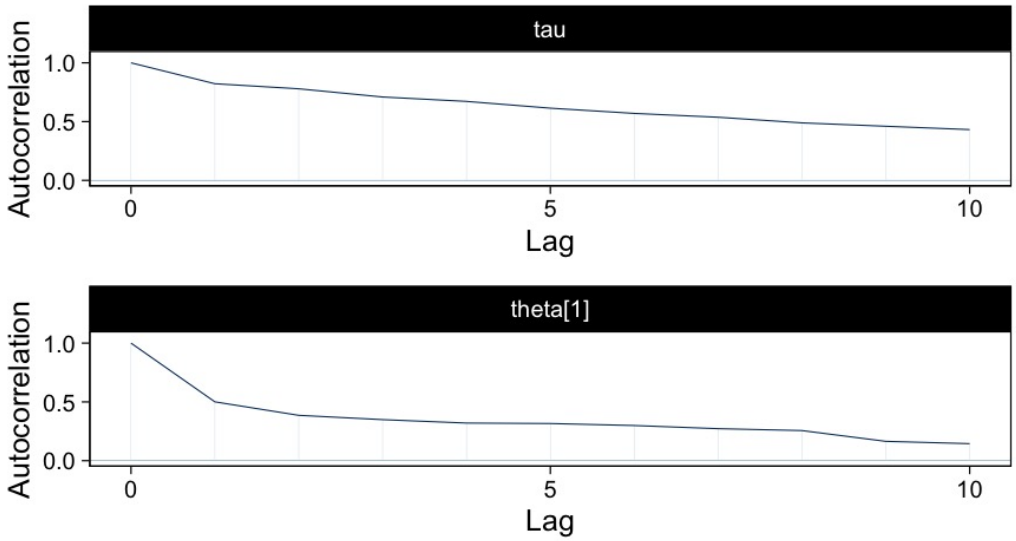


FIGURA 8. *Autocorrelaciones en las simulaciones.*

```
stanfit ← rstan::read_stan_csv(muestras$output_files())
stanfit
```

```
Inference for Stan model: modelo-escuelas-202202241738-1-87573b.
1 chains, each with iter=10000; warmup=5000; thin=1;
post-warmup draws per chain=5000, total post-warmup draws=5000.

          mean se_mean  sd  2.5%    25%   50%   75% 97.5% n_eff Rhat
mu         4.0    0.16 3.3  -2.4   1.71   3.9   6.1  10.7   438    1
tau        4.2    0.22 3.3   0.6   1.91   3.4   5.5  12.7   224    1
theta[1]   6.2    0.23 5.9  -3.5   2.25   5.4   9.0  21.0   637    1
theta[2]   4.7    0.19 5.0  -5.2   1.37   4.3   7.7  15.5   736    1
theta[3]   3.5    0.15 5.4  -8.4   0.78   3.3   6.7  13.9  1265    1
theta[4]   4.5    0.15 5.0  -5.3   1.54   4.3   7.4  14.9  1063    1
theta[5]   3.1    0.15 4.8  -7.3   0.41   3.2   6.1  12.2   962    1
theta[6]   3.6    0.15 5.0  -6.8   0.96   3.4   6.6  13.7  1154    1
theta[7]   6.2    0.30 5.4  -2.3   2.47   5.8   9.3  18.5   327    1
theta[8]   4.5    0.17 5.5  -5.9   1.42   4.3   7.7  16.5  1052    1
lp__     -16.1    0.62 5.7 -27.1 -20.25 -16.2 -12.0  -5.3    85    1

Samples were drawn using NUTS(diag_e) at Thu Feb 24 17:38:38 2022.
For each parameter, n_eff is a crude measure of effective sample size,
and Rhat is the potential scale reduction factor on split chains (at
convergence, Rhat=1).
```

Como vemos, seguimos teniendo problemas con la exploración del espacio parametral (donde está definida nuestra distribución de θ) y tenemos dificultades en explorar esa zona con τ pequeña. Esto lo confirmamos en la siguiente gráfica.

3.4. Haciendo tweaks en el simulador

Podríamos correr una cadena con algunas opciones que permitan la exploracion mas segura de la distribución.

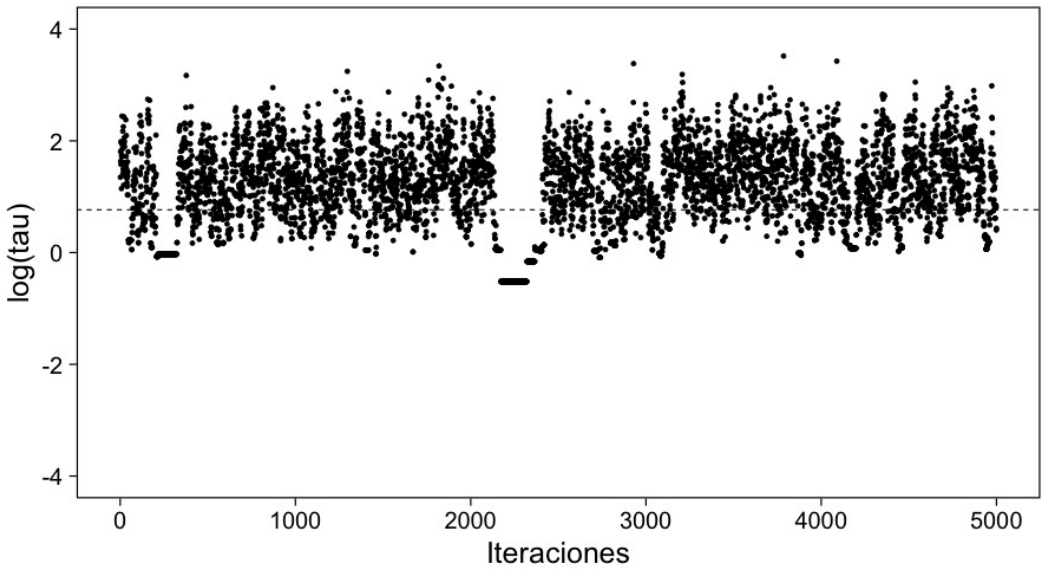


FIGURA 9. *Trayectorías de simulaciones.*

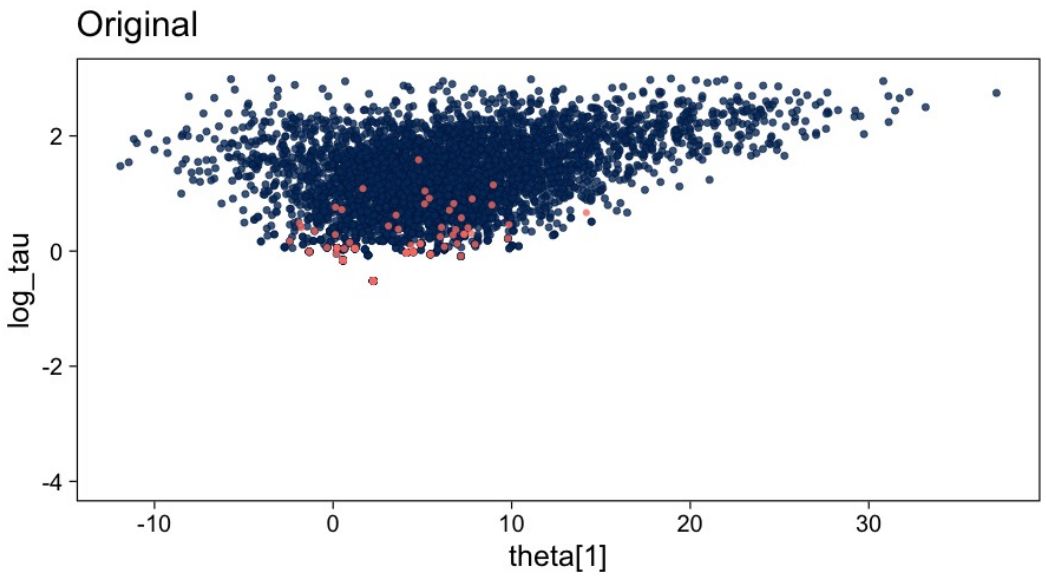


FIGURA 10. *Gráficos de dispersión.*

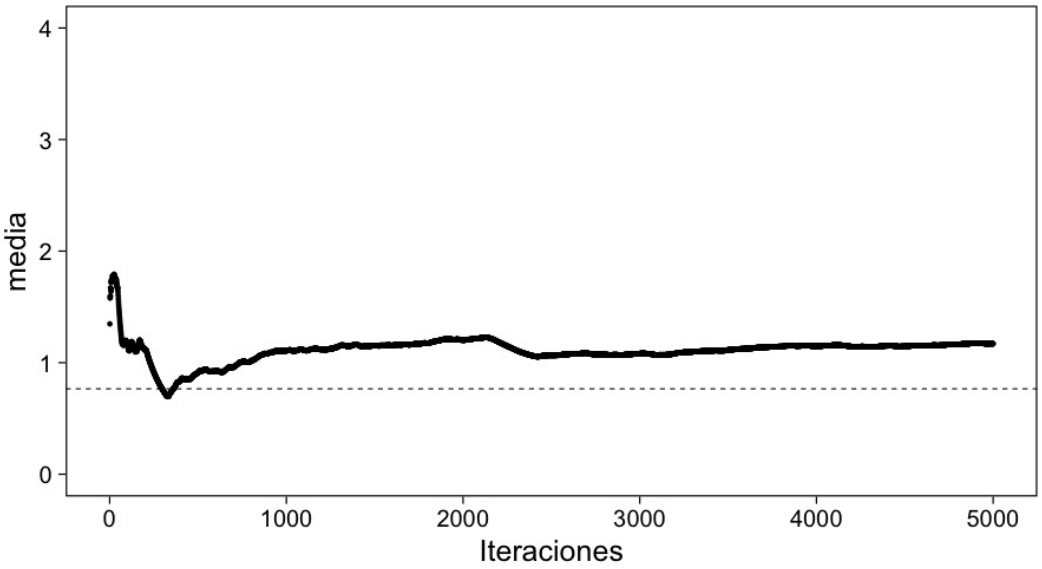


FIGURA 11. *Media acumulada de* $\log \tau$.

```
muestras ← modelo$sample(data         = data_list,
                         chains       = 1,
                         iter         = 5000,
                         iter_warmup  = 5000,
                         seed         = 483892929,
                         refresh      = 10000,
                         adapt_delta  = .90)
```

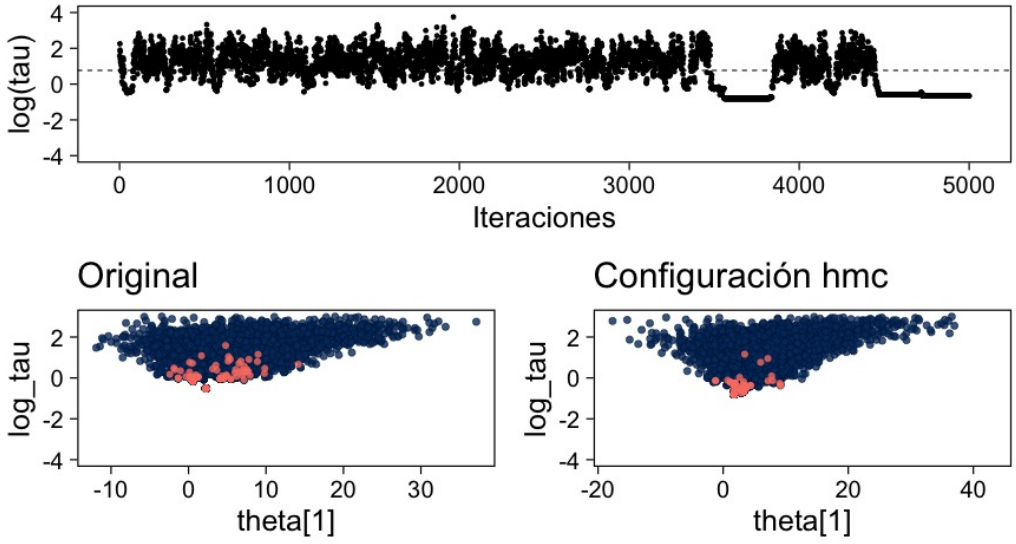


FIGURA 12. *Gráficos de comparación.*

4. CAMBIANDO LIGERAMENTE EL MODELO

Tener cuidado en la simulación del sistema Hamiltoniano nos ayuda hasta cierto punto. Seguimos teniendo problemas y no hay garantías que nuestra simulación y nuestros estimadores Monte Carlo no estén sesgados.

Esta situación es muy común en *modelos jerárquicos*. El cual hemos definido como

$$\mu \sim \mathsf{N}(0, 5)\,, \tag{2a}$$
$$\tau \sim \mathsf{Half\text{-}Cauchy}(0, 5)\,, \tag{2b}$$
$$\theta_j \sim \mathsf{N}(\mu, \tau), \qquad j = 1, \ldots, J\,, \tag{2c}$$
$$y_j \sim \mathsf{N}(\theta_j, \sigma_j), \qquad j = 1, \ldots, J\,. \tag{2d}$$

El problema es la geometría de la distribución posterior. La ventaja es que existe una solución sencilla para hacer el problema de muestreo mas sencillo. Esto es al escribir el modelo en términos de una variable auxiliar:

$$\mu \sim \mathsf{N}(0, 5)\,, \tag{3a}$$
$$\tau \sim \mathsf{Half\text{-}Cauchy}(0, 5)\,, \tag{3b}$$
$$\tilde{\theta}_j \sim \mathsf{N}(0, 1), \qquad j = 1, \ldots, J\,, \tag{3c}$$
$$\theta_j = \mu + \tau \cdot \tilde{\theta}_j, \qquad j = 1, \ldots, J\,, \tag{3d}$$
$$y_j \sim \mathsf{N}(\theta_j, \sigma_j), \qquad j = 1, \ldots, J\,. \tag{3e}$$

El modelo en `Stan` es muy parecido. La nomenclatura que se utiliza es: **modelo centrado** para el primero, y para la reparametrización presentada en la ecuación de arriba nos referimos a un **modelo no centrado**.

```
## Cambiando de modelo -------------------------------------
print_file("modelos/caso-escuelas/modelo-escuelas-ncp.stan")
```

> Nota que la definición de nuevos parametros se hace desde el bloque `transformed parameters` en donde la asignación se ejecuta componente por componente mientras que la definición del modelo de probabilidad conjunto se puede hacer de manera vectorizada.

Igual que antes lo necesitamos compilar para hacerlo un objeto ejecutable desde `R`.

```
ruta_ncp ← file.path("modelos/caso-escuelas/modelo-escuelas-ncp.stan")
modelo_ncp ← cmdstan_model(ruta_ncp, dir = modelos_files)
```

Muestreamos de la posterior

```
muestras_ncp ← modelo_ncp$sample(data = data_list,
                          chains = 1,
                          iter=5000,
                          iter_warmup=5000,
                          seed=483892929,
                          refresh=10000)
```

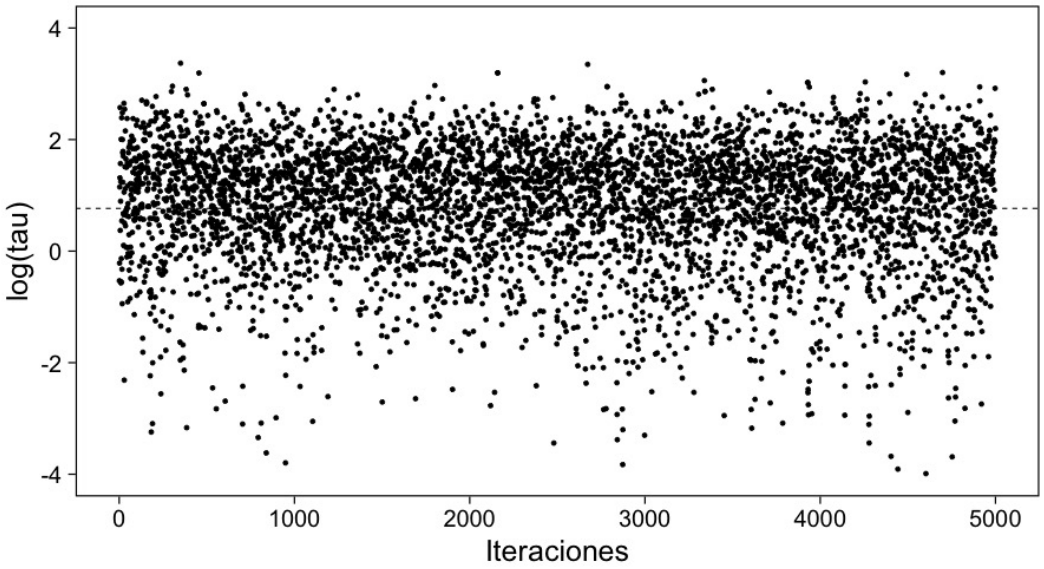
```
Running MCMC with 1 chain...

Chain 1 Iteration:    1 / 10000 [  0%]  (Warmup)
```

```
Chain 1 Iteration: 5001 / 10000 [ 50%]  (Sampling)
Chain 1 Iteration: 10000 / 10000 [100%]  (Sampling)
Chain 1 finished in 0.3 seconds.
```

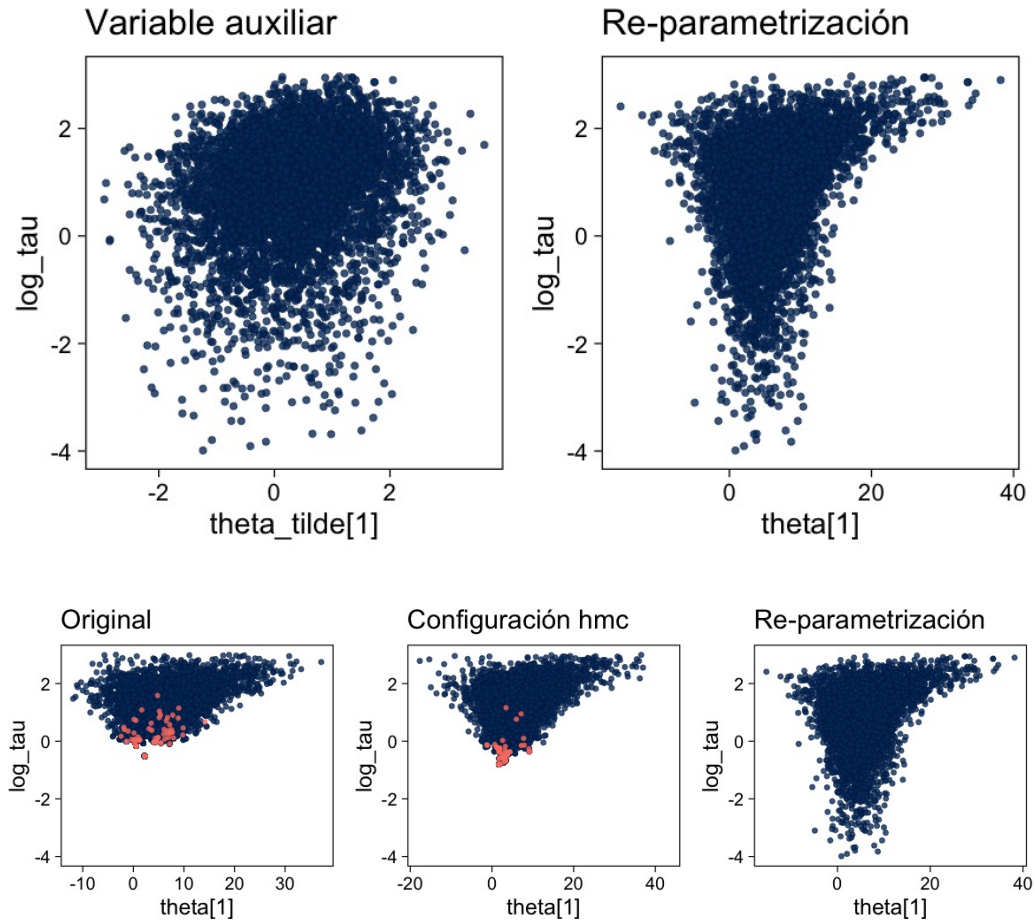
```
stanfit_ncp ← rstan::read_stan_csv(muestras_ncp$output_files())
stanfit_ncp
```

```
Inference for Stan model: modelo-escuelas-ncp-202202241738-1-02582b.
1 chains, each with iter=10000; warmup=5000; thin=1;
post-warmup draws per chain=5000, total post-warmup draws=5000.

                 mean se_mean   sd   2.5%   25%   50%   75% 97.5% n_eff Rhat
mu               4.33    0.05 3.38  -2.32  2.11  4.30  6.54  10.9  4653    1
tau              3.60    0.05 3.20   0.15  1.27  2.78  4.94  12.0  4006    1
theta_tilde[1]   0.31    0.01 0.99  -1.65 -0.38  0.32  1.00   2.2  5272    1
theta_tilde[2]   0.10    0.01 0.95  -1.82 -0.52  0.11  0.73   2.0  5086    1
theta_tilde[3]  -0.08    0.01 0.97  -1.99 -0.73 -0.10  0.58   1.8  4702    1
theta_tilde[4]   0.07    0.01 0.93  -1.77 -0.57  0.06  0.71   1.9  5974    1
theta_tilde[5]  -0.16    0.01 0.93  -1.97 -0.79 -0.17  0.48   1.7  5767    1
theta_tilde[6]  -0.08    0.01 0.94  -1.88 -0.73 -0.08  0.54   1.8  5841    1
theta_tilde[7]   0.37    0.01 0.97  -1.60 -0.27  0.39  1.03   2.2  4837    1
theta_tilde[8]   0.09    0.01 0.99  -1.81 -0.59  0.10  0.78   2.0  5059    1
theta[1]         6.10    0.08 5.60  -3.23  2.51  5.52  8.98  19.2  4663    1
theta[2]         4.89    0.07 4.68  -4.04  1.89  4.69  7.62  14.8  4869    1
theta[3]         3.88    0.08 5.35  -7.77  1.04  4.01  7.07  13.9  4454    1
theta[4]         4.74    0.06 4.81  -4.63  1.68  4.63  7.63  14.8  5533    1
theta[5]         3.55    0.07 4.80  -6.99  0.80  3.71  6.57  12.4  4890    1
theta[6]         3.88    0.07 4.97  -6.89  1.06  4.04  6.96  13.3  5390    1
theta[7]         6.29    0.07 5.16  -2.45  2.93  5.79  9.01  18.6  4983    1
theta[8]         4.87    0.08 5.35  -5.83  1.79  4.70  7.91  15.7  4705    1
lp__            -6.99    0.05 2.30 -12.16 -8.36 -6.70 -5.33  -3.4  2153    1

Samples were drawn using NUTS(diag_e) at Thu Feb 24 17:38:40 2022.
For each parameter, n_eff is a crude measure of effective sample size,
and Rhat is the potential scale reduction factor on split chains (at
convergence, Rhat=1).
```

Si graficamos la dispersión de τ ($\log \tau$), vemos un mejor comportamiento (del cual ya teníamos indicios por los diagnósticos del modelo).



Si regresamos a los gráficos de dispersión para verificar que se hayan resuelto los problemas observamos lo siguiente:



REFERENCIAS

[1] B. Carpenter, A. Gelman, M. D. Hoffman, D. Lee, B. Goodrich, M. Betancourt, M. Brubaker, J. Guo, P. Li, and A. Riddell. Stan: a probabilistic programming language. *Journal of Statistical Software*, 76(1): nil, 2017. . URL https://doi.org/10.18637/jss.v076.i01. 1

[2] A. Gelman, J. B. Carlin, H. S. Stern, D. B. Dunson, A. Vehtari, and D. B. Rubin. *Bayesian Data Analysis*, volume 2. CRC press Boca Raton, FL, 2014. 5

[3] A. Johnson, M. Ott, and M. Dogucu. *Bayes Rules! An Introduction to Applied Bayesian Modeling*. 2021. 1

[4] D. B. Rubin. Estimation in Parallel Randomized Experiments. *Journal of Educational Statistics*, 6(4): 377–401, 1981. ISSN 0362-9791. . 5