Tarea 4 - MLG

Blanca Garcia - 118886 Yuneri Perez - 199813
Thomas Rudolf - 169293 Mariano Villafuerte - 156057

Spiegelhalter et al. (1995) analiza la mortalidad del escarabajo del trigo en la siguiente tabla, usando BUGS.

Dosis	# muertos	# expuestos
1.6907	6	59
1.7242	13	60
1.7552	18	62
1.7842	28	56
1.8113	52	63
1.8369	53	59
1.8610	61	62
1.8839	60	60

Estos autores usaron una parametrización usual en dos parámetros de la forma $p_i \equiv P(\text{muerte}|w_i)$, pero comparan tres funciones ligas diferentes:

$$\text{logit} : p_i = \frac{\exp(\alpha + \beta z_i)}{1 + \exp(\alpha + \beta z_i)}$$

$$\text{probit} : p_i = \Phi(\alpha + \beta z_i)$$

$$\text{complementario log-log} : p_i = 1 - \exp[-\exp(\alpha + \beta z_i)]$$

en donde se usa la covariada centrada $z_i = w_i - \bar{w}$ para reducir la correlación entre la ordenada α y la pendiente β. En OpenBUGS el código para implementar este modelo es el que sigue:

```
model {
  for (i in 1:k) {
    y[i] ~ dbin(p[i], n[i])
    logit(p[i]) <- alpha + beta * (w[i] - mean(w))
    # probit(p[i]) <- alpha + beta * (w[i] - mean(w))
    # cloglog(p[i]) <- alpha + beta * (w[i] - mean(w))
  } # End of loop over i
  alpha ~ dnorm(0.0, 1.0E-3)
  beta ~ dnorm(0.0, 1.0E-3)
} # End of model
```

Lo que sigue al símbolo # es un comentario, así que esta versión corresponde al modelo logit. También dbin denota la distribución binomial y dnorm denota la distribución normal, donde el segundo argumento denota la precisión, no la varianza (entonces las iniciales normales para α y β tienen precisión 0.001, que son aproximadamente iniciales planas (no informativas)). Se hace el análisis en STAN, se incluye el código en el anexo.

Logit model

variable	mean	sd	q5	q95
lp___	−187.27	1.02	−189.32	−186.29
alpha	0.75	0.14	0.52	0.98
beta	34.64	2.98	29.89	39.72
p[1]	0.06	0.02	0.04	0.09
p[2]	0.16	0.03	0.12	0.21
p[3]	0.36	0.03	0.30	0.42
p[4]	0.61	0.03	0.55	0.66
p[5]	0.80	0.03	0.75	0.84
p[6]	0.90	0.02	0.87	0.93
p[7]	0.96	0.01	0.93	0.97
p[8]	0.98	0.01	0.97	0.99

5. Supongan que se tiene una matriz de 4×4 de variables aleatorias Bernoulli, y la denotamos por $[X_{ij}]$, y sea $N(X)$ el número total de éxitos en X (la suma de X) y $D(X)$ es el total de vecinos de dichos éxitos (horizontales o verticales) que difieren. Por ejemplo:

X	$N(X)$	$D(X)$
$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$	1	2
$\begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix}$	3	5

Noten que si se escriben los elementos de X en un vector V , entonces existe una matriz M de 24×16 tal que $D(X)$ es la suma de los valores absolutos de MV . El 24 surge porque hay 24 pares de vecinos a revisar. Supongan que $\pi(X)$, la distribución de X, es proporcional a

$$\pi(X) \propto p^{N(X)}(1-p)^{16-N(X)} exp(-\lambda D(X))$$

Si $\lambda = 0$, las variables son iid Bernoulli (p).
Usar el método de Metropolis-Hastings usando los siguientes kerneles de transición.

Hay 216 posibles estados, uno por cada posible valor de X.

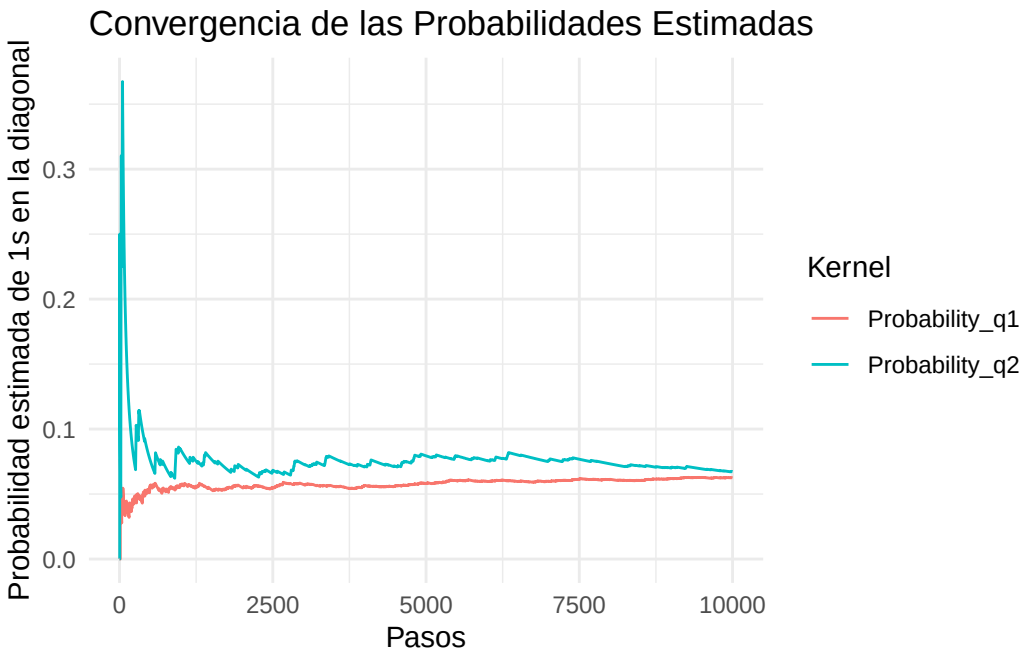
a. Sea q_1 tal que cada transición es igualmente plausible con probabilidad 1/216. Esto es, el siguiente estado candidato para X es simplemente un vector de 16 iid Bernoulli (p).

b. Sea q_2 tal que se elige una de las 16 entradas en X con probabilidad 1/16, y luego se determina el valor de la celda a ser 0 o 1 con probabilidad 0.5 en cada caso. Entonces sólo un elemento de X puede cambiar en cada transición a lo más.

Ambas q son simétricas, irreducibles y tienen diagonales positivas. La primera se mueve más rápido que la segunda.
Estamos interesados en la probabilidad de que todos los elementos de la diagonal sean 1. Usen las dos q para estimar la probabilidad de 1's en la diagonal para $\lambda = 0, 1, 3$ y $p = 0.5, 0.8$. Esto se puede hacer calculando la fracción acumulada de veces que la cadena tiene 1's sobre la diagonal conforme se muestrea de la cadena. Comparar los resultados de las 2 cadenas. Tienen el mismo valor asintótico, pero ¿una cadena llega más rápido que laotra? ¿Alguna cadena muestra más autocorrelación que la otra (por ejemplo, estimados de la probabilidad en 100 pasos sucesivos muestran más correlación para una cadena que para la otra?).
En este problema, también determinen cuántas simulaciones hay que hacer, para desechar el periodo de calentamiento.

```r
# funciones
pi <- function(X, p, lambda) {
  N_X <- sum(X)
  D_X <- D(X)
  return((p^N_X) * ((1 - p)^(16 - N_X)) * exp(-lambda * D_X))
}

D <- function(X) {
  differences <- 0
  for (i in 1:4) {
    for (j in 1:4) {
      if (j < 4) {
        differences <- differences + as.numeric(X[i, j] != X[i, j+1])
      }
      if (i < 4) {
        differences <- differences + as.numeric(X[i, j] != X[i+1, j])
      }
    }
  }
  return(differences)
}

q1_generate <- function(p) {
  return(matrix(rbinom(16, 1, p), nrow = 4, ncol = 4))
}

q2_generate <- function(X) {
  X_prime <- X
  i <- sample(1:4, 1)
  j <- sample(1:4, 1)
  X_prime[i, j] <- 1 - X_prime[i, j]
  return(X_prime)
}

metropolis_hastings_step <- function(X, p, lambda, kernel) {
  if (kernel == 'q1') {
    X_prime <- q1_generate(p)
  } else if (kernel == 'q2') {
    X_prime <- q2_generate(X)
  }
  acceptance_prob <- min(1, pi(X_prime, p, lambda) / pi(X, p, lambda))
```

```
  if (runif(1) < acceptance_prob) {
    return(X_prime)
  } else {
    return(X)
  }
}

simulate_metropolis_hastings <- function(p, lambda, kernel, n_steps = 10000) {
  X <- matrix(rbinom(16, 1, p), nrow = 4, ncol = 4)
  diagonal_ones_history <- numeric(n_steps)
  for (step in 1:n_steps) {
    X <- metropolis_hastings_step(X, p, lambda, kernel)
    diagonal_ones_history[step] <- all(diag(X) == 1)
  }
  return(cumsum(diagonal_ones_history) / (1:n_steps))
}

# Parámetros
p <- 0.5
lambda <- 0
n_steps <- 10000

# Simulaciones para ambos kerneles
probability_history_q1 <- simulate_metropolis_hastings(p, lambda, 'q1', n_steps)
probability_history_q2 <- simulate_metropolis_hastings(p, lambda, 'q2', n_steps)

# Graficar la convergencia de las probabilidades
df <- data.frame(Step = 1:n_steps, Probability_q1 = probability_history_q1, Probability_q2
df_melt <- reshape2::melt(df, id.vars = "Step", variable.name = "Kernel", value.name = "Pr

ggplot(df_melt, aes(x = Step, y = Probability, colour = Kernel)) +
  geom_line() +
  labs(title = "Convergencia de las Probabilidades Estimadas",
       x = "Pasos",
       y = "Probabilidad estimada de 1s en la diagonal") +
  theme_minimal()
```


Convergencia de las Probabilidades Estimadas
Probabilidad estimada de 1s en la diagonal
0.0
0.1
0.2
0.3
0
2500
5000
7500
10000
Pasos
Kernel
Probability_q1
Probability_q2

6. Considera los siguientes números:

$$0.4,\ 0.01,\ 0.2,\ 0.1,\ 2.1,\ 0.1,\ 0.9,\ 2.4,\ 0.1,\ 0.2$$

Usen la distribución exponencial $Y_i \sim exp(\theta)$ para modelar estos datos y asignen una inicial sobre $log(\theta)$.

- Definan los datos en WinBUGS (u OpenBUGS). Usen $\theta = 1$ como valor inicial.

```
# Datos observados
datos_obs <- c(0.4, 0.01, 0.2, 0.1, 2.1, 0.1, 0.9, 2.4, 0.1, 0.2)

# Valor inicial para theta
theta_inicial <- 1
```

- Escriban el código para el modelo.

```
// Modelo exponencial
data {
  int<lower=0> N;
  vector[N] y;
  real<lower=0> theta0;
}

parameters {
  real<lower=0> theta;
}

model {
  theta ~ exponential(1 / theta0);
  y ~ exponential(theta);
}
```

- Compilen el modelo y obtengan una muestra de 1000 iteraciones después de descartar las 500 iteraciones iniciales como burnin.

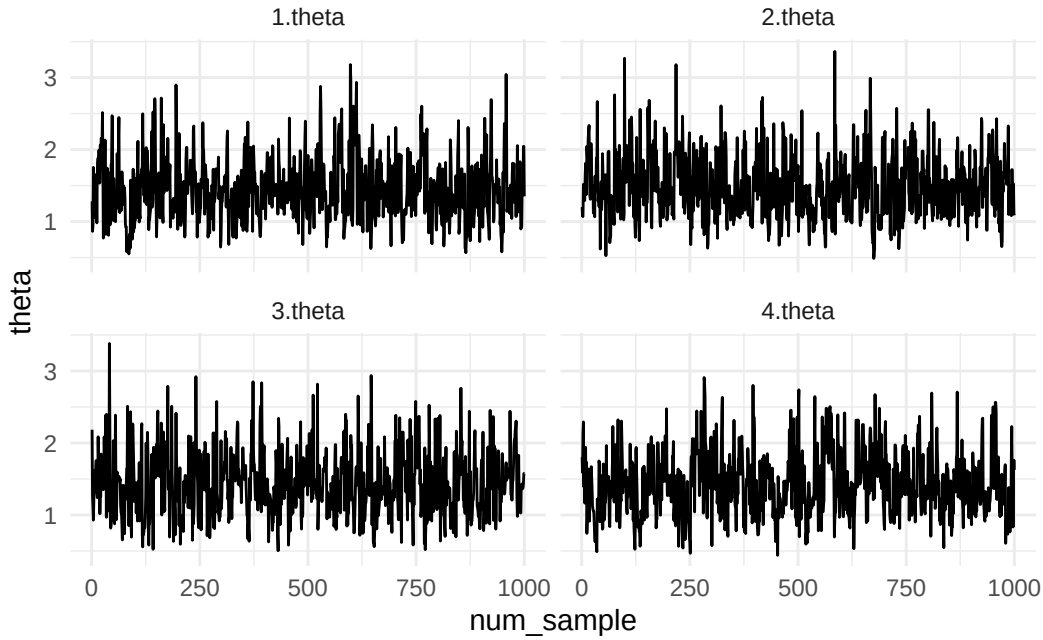
```
mod <- cmdstan_model("modelo_p6_exp.stan")

stan_data <- list(N = length(datos_obs),
                  y = datos_obs,
```

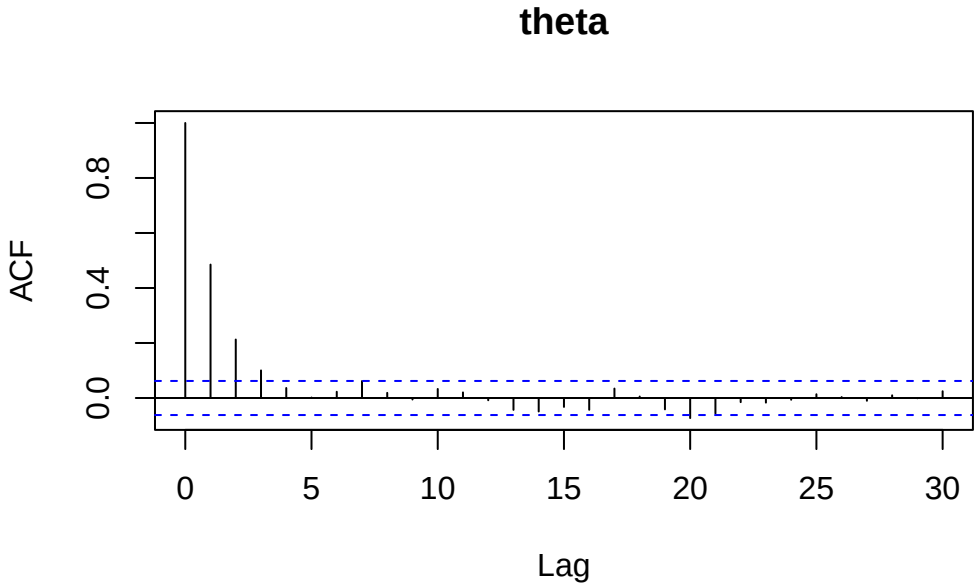
```
                    theta0 = theta_inicial)

fit <- mod$sample(
  data = stan_data,
  seed = 156057,
  chains = 4,
  parallel_chains = 4,
  iter_warmup = 500,
  iter_sampling = 1000,
  refresh = 0,
  show_messages=F
)
```

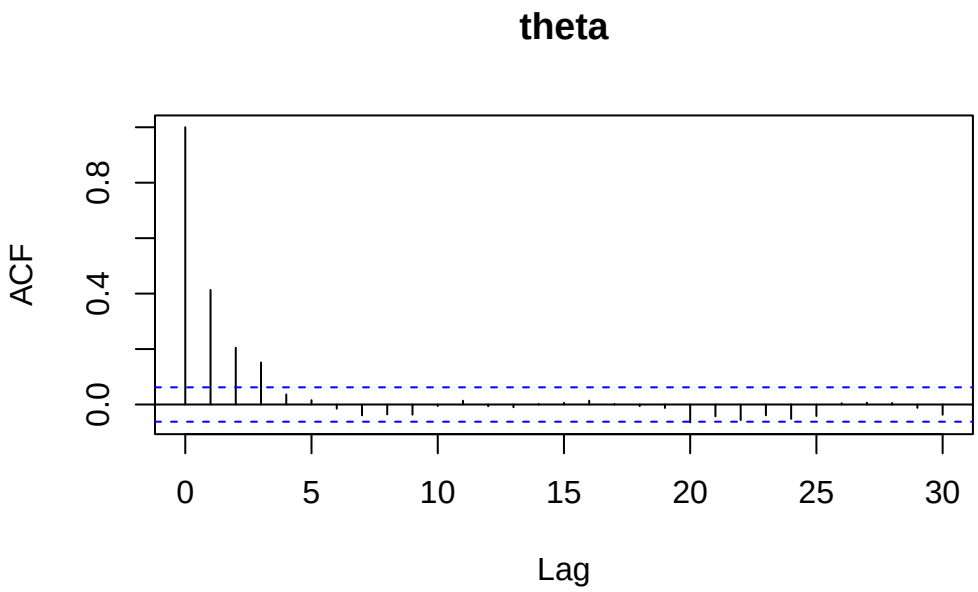
variable	mean	sd	q5	q95
lp___	−7.31	0.69	−8.73	−6.80
theta	1.45	0.44	0.81	2.25

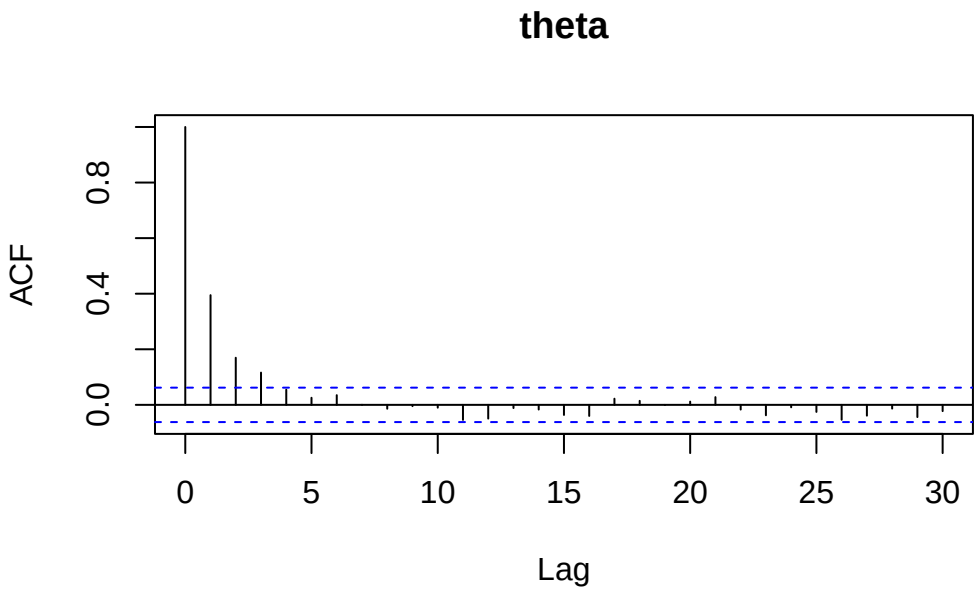
- Monitoreen la convergencia gráficamente usando gráficas 'trace' y de autocorrelación.

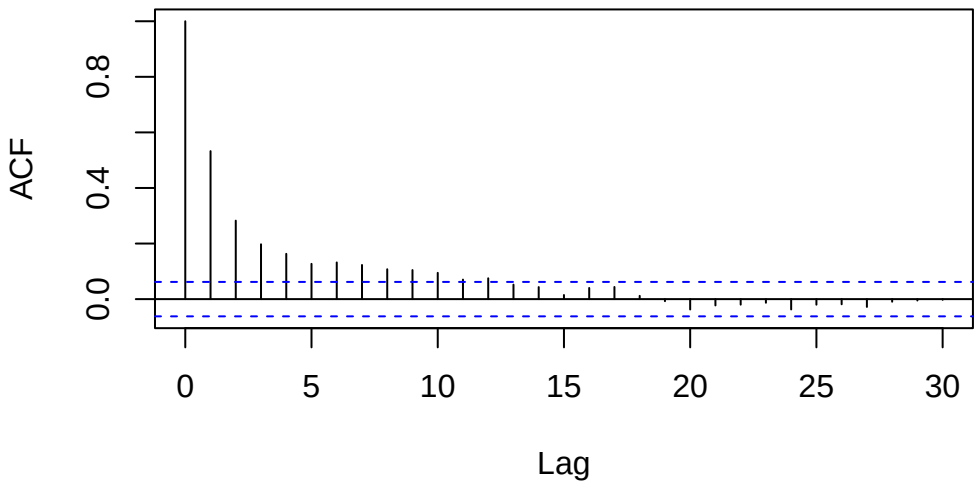
```
# Extraer las muestras
samples <- as.data.frame(fit$draws(variables='theta')) %>%
  mutate(num_sample=seq(1,1000, by=1))%>%
  pivot_longer(cols=-c(num_sample),
               names_to='corrida',
               values_to='theta')
```


1.theta
2.theta
3.theta
4.theta
theta
num_sample
0
250
500
750
1000
1
2
3


theta
ACF
Lag
0.0
0.4
0.8
0
5
10
15
20
25
30


theta
ACF
0.0
0.4
0.8
0
5
10
15
20
25
30
Lag


theta
ACF
0.0
0.4
0.8
0
5
10
15
20
25
30
Lag



- Obtengan estadísticas sumarias posteriores y densidades para θ, $1/\theta$ y $log(\theta)$

```
samples %>%
  mutate(inverso = 1 / theta,
         logaritmo = log(theta)) %>%
  summarize(across(c(theta, inverso, logaritmo),
                   list(
                     mean = ~ mean(.x, na.rm = TRUE),
                     median = ~ median(.x, na.rm = TRUE),
                     p5 = ~ quantile(.x, probs = 0.05, na.rm = TRUE),
                     p95 = ~ quantile(.x, probs = 0.95, na.rm = TRUE)
                   ),
                   .names = "{.col}_{.fn}")) %>%
  pivot_longer(everything(), names_to = c("statistic", ".value"), names_sep = "_") %>%
  gt() %>%
  fmt_number()
```

statistic	mean	median	p5	p95
theta	1.45	1.41	0.81	2.25
inverso	0.76	0.71	0.44	1.23
logaritmo	0.33	0.34	−0.21	0.81

- Con los datos el primer inciso, ahora usen las distribuciones (i) gamma (ii) log-normal para modelar los datos, así como (iii) normal para los logaritmos de los valores originales. En todos los modelos hagan un ejercicio similar al de los numerales previos, y comparen los resultados obtenidos bajo todos los modelos. Hagan todos los supuestos que tengan que hacer.

Anexo

STAN Model

```
data {
  int<lower=0> k; // observaciones
  array[k] int<lower=0> n; // expuestos
  array[k] int<lower=0> y; // muertos
  array[k] real w; // dosis
}

transformed data {
  real w_mean = mean(w); // promedio de w para centrar
}

parameters {
  real alpha;
  real beta;
}

transformed parameters {
  array[k] real p; // probability of success

  for (i in 1:k) {
    p[i] = inv_logit(alpha + beta * (w[i] - w_mean)); // logit
    // p[i] = Phi_approx(alpha + beta * (w[i] - w_mean)); // probit
    // p[i] = 1 - exp(-exp(alpha + beta * (w[i] - w_mean))); // cloglog
  }
}

model {
  alpha ~ normal(0, 1000); // inicial de alfa
  beta ~ normal(0, 1000); // inicial de beta

  for (i in 1:k) {
```

```
    y[i] ~ binomial(n[i], p[i]);
  }
}
```