Tarea 3 - Modelos Lineales Generalizados

Fecha de entrega: 6 de marzo de 2024.

- Blanca E. García Manjarrez – 118886
- Mariano Villafuerte Gonzalez – 156057
- Thomas M. Rudolf - 169293
- Yuneri Pérez Arellano - 199813

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

Dosis	No Muertos	No Expuestos
w_i	y_i	n_i
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 $p_i \equiv P(muerte \mid w_i)$, pero comparan tres funciones ligas diferentes:

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

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

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

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:

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)). Hacer el análisis en OpenBUGS.

```
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[])
} #fin del loop i
alpha ~ dnorm(0.0,1.0e-3)
dbeta ~ dnorm(0.0,1.0e-3)
} #fin del código
```

```
dosis <- c(1.6907, 1.7242, 1.7552, 1.7842, 1.8113, 1.8369, 1.8610, 1.8839)
muertos <- c(6, 13, 18, 28, 52, 53, 61, 60)
expuestos <- c(59, 60, 62, 56, 63, 59, 62, 60)

data_input <- list(
  k = length(muertos),
  n = expuestos,
  y = muertos,
  w = dosis - mean(dosis)
)
```

```
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] = 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]);
  }
}
```

```
# Realiza el muestreo
model_fit <- rstan::sampling(model_p1_cloglog, data = data_input)
```

```
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 1).
## Chain 1:
```

```
## Chain 1: Gradient evaluation took 1.6e-05 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0.16 seconds.
## Chain 1: Adjust your expectations accordingly!
## Chain 1:
## Chain 1:
## Chain 1: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 1: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 1: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 1: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 1: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 1: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 1: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 1: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 1: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 1: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 1: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 1: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 0.009 seconds (Warm-up)
## Chain 1:                0.007 seconds (Sampling)
## Chain 1:                0.016 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 2).
## Chain 2:
## Chain 2: Gradient evaluation took 1e-06 seconds
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0.01 seconds.
## Chain 2: Adjust your expectations accordingly!
## Chain 2:
## Chain 2:
## Chain 2: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 2: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 2: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 2: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 2: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 2: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 2: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 2: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 2: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 2: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 2: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 2: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 2:
## Chain 2:  Elapsed Time: 0.009 seconds (Warm-up)
## Chain 2:                0.007 seconds (Sampling)
## Chain 2:                0.016 seconds (Total)
## Chain 2:
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 3).
## Chain 3:
## Chain 3: Gradient evaluation took 2e-06 seconds
## Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0.02 seconds.
## Chain 3: Adjust your expectations accordingly!
## Chain 3:
```

```
## Chain 3:
## Chain 3: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 3: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 3: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 3: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 3: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 3: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 3: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 3: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 3: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 3: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 3: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 3: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 3:
## Chain 3:  Elapsed Time: 0.008 seconds (Warm-up)
## Chain 3:                0.007 seconds (Sampling)
## Chain 3:                0.015 seconds (Total)
## Chain 3:
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 4).
## Chain 4:
## Chain 4: Gradient evaluation took 1e-06 seconds
## Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 0.01 seconds.
## Chain 4: Adjust your expectations accordingly!
## Chain 4:
## Chain 4:
## Chain 4: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 4: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 4: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 4: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 4: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 4: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 4: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 4: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 4: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 4: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 4: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 4: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 4:
## Chain 4:  Elapsed Time: 0.008 seconds (Warm-up)
## Chain 4:                0.007 seconds (Sampling)
## Chain 4:                0.015 seconds (Total)
## Chain 4:
```

```
# Interpretaciones
print(model_fit)
```

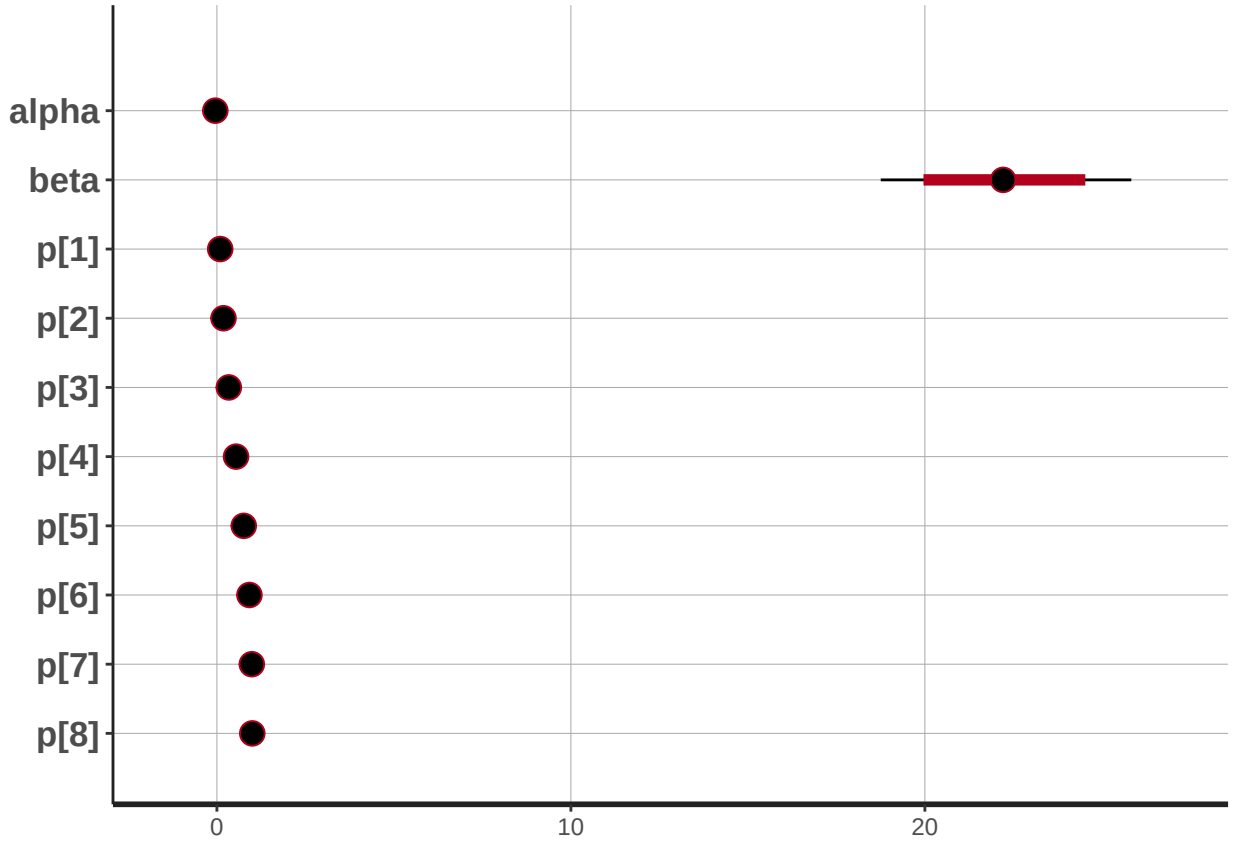
```
## Inference for Stan model: anon_model.
## 4 chains, each with iter=2000; warmup=1000; thin=1;
## post-warmup draws per chain=1000, total post-warmup draws=4000.
##
##          mean se_mean   sd    2.5%     25%     50%     75%   97.5% n_eff Rhat
## alpha   -0.05    0.00 0.08   -0.20   -0.10   -0.04    0.01    0.11  2715    1
## beta    22.22    0.04 1.80   18.75   20.95   22.21   23.45   25.83  2610    1
## p[1]     0.09    0.00 0.02    0.06    0.08    0.09    0.11    0.14  2364    1
```

```
## p[2]     0.19    0.00 0.03    0.14    0.17    0.19    0.20    0.24  2324    1
## p[3]     0.34    0.00 0.03    0.28    0.32    0.34    0.36    0.40  2318    1
## p[4]     0.54    0.00 0.03    0.48    0.52    0.54    0.56    0.60  2529    1
## p[5]     0.76    0.00 0.03    0.70    0.74    0.76    0.78    0.81  3214    1
## p[6]     0.92    0.00 0.02    0.87    0.90    0.92    0.93    0.95  3506    1
## p[7]     0.98    0.00 0.01    0.96    0.98    0.99    0.99    1.00  3356    1
## p[8]     1.00    0.00 0.00    0.99    1.00    1.00    1.00    1.00  2859    1
## lp__  -183.34    0.02 0.97 -186.03 -183.75 -183.04 -182.64 -182.36  1686    1
##
## Samples were drawn using NUTS(diag_e) at Tue Mar  5 14:56:34 2024.
## 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).
```

```
plot(model_fit)
```

```
## ci_level: 0.8 (80% intervals)
```

```
## outer_level: 0.95 (95% intervals)
```




```
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]);
  }
}
```

```
# Realiza el muestreo
model_fit2 <- rstan::sampling(modelo_p1_logit, data = data_input)
```

```
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 1.1e-05 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0.11 seconds.
## Chain 1: Adjust your expectations accordingly!
## Chain 1:
## Chain 1:
## Chain 1: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 1: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 1: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 1: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 1: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 1: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 1: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 1: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 1: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 1: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 1: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 1: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 0.007 seconds (Warm-up)
## Chain 1:                0.006 seconds (Sampling)
## Chain 1:                0.013 seconds (Total)
## Chain 1:
```

```
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 2).
## Chain 2:
## Chain 2: Gradient evaluation took 1e-06 seconds
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0.01 seconds.
## Chain 2: Adjust your expectations accordingly!
## Chain 2:
## Chain 2:
## Chain 2: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 2: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 2: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 2: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 2: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 2: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 2: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 2: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 2: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 2: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 2: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 2: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 2:
## Chain 2:  Elapsed Time: 0.007 seconds (Warm-up)
## Chain 2:                0.007 seconds (Sampling)
## Chain 2:                0.014 seconds (Total)
## Chain 2:
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 3).
## Chain 3:
## Chain 3: Gradient evaluation took 1e-06 seconds
## Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0.01 seconds.
## Chain 3: Adjust your expectations accordingly!
## Chain 3:
## Chain 3:
## Chain 3: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 3: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 3: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 3: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 3: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 3: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 3: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 3: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 3: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 3: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 3: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 3: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 3:
## Chain 3:  Elapsed Time: 0.007 seconds (Warm-up)
## Chain 3:                0.006 seconds (Sampling)
## Chain 3:                0.013 seconds (Total)
## Chain 3:
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 4).
## Chain 4:
## Chain 4: Gradient evaluation took 1e-06 seconds
```

```
## Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 0.01 seconds.
## Chain 4: Adjust your expectations accordingly!
## Chain 4:
## Chain 4:
## Chain 4: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 4: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 4: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 4: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 4: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 4: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 4: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 4: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 4: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 4: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 4: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 4: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 4:
## Chain 4:  Elapsed Time: 0.007 seconds (Warm-up)
## Chain 4:                0.006 seconds (Sampling)
## Chain 4:                0.013 seconds (Total)
## Chain 4:
```

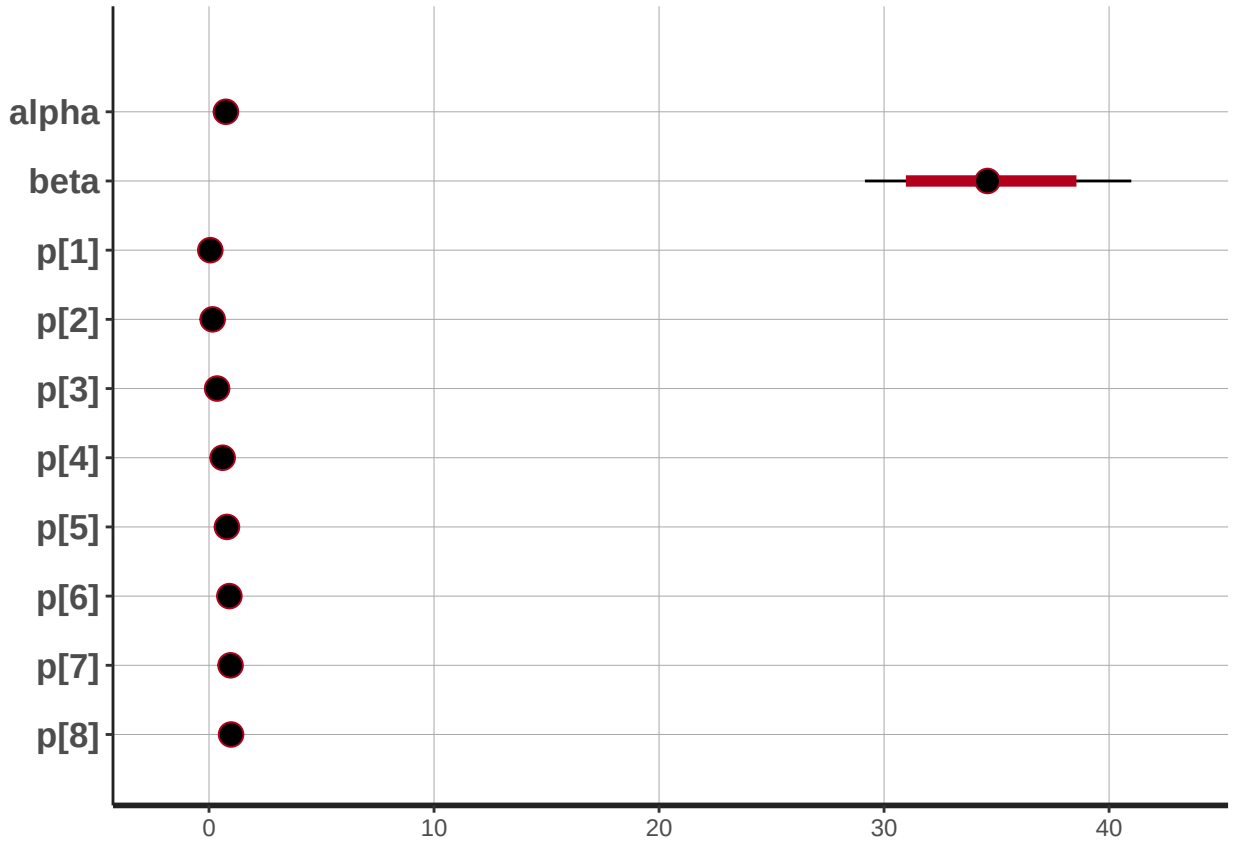
```
# Interpretaciones
print(model_fit2)
```

```
## Inference for Stan model: anon_model.
## 4 chains, each with iter=2000; warmup=1000; thin=1;
## post-warmup draws per chain=1000, total post-warmup draws=4000.
##
##          mean se_mean   sd    2.5%     25%     50%     75%   97.5% n_eff Rhat
## alpha    0.75    0.00 0.14    0.47    0.66    0.75    0.84    1.03  2564    1
## beta    34.70    0.06 2.98   29.15   32.68   34.60   36.62   40.99  2404    1
## p[1]     0.06    0.00 0.02    0.03    0.05    0.06    0.07    0.09  3340    1
## p[2]     0.16    0.00 0.03    0.11    0.14    0.16    0.18    0.22  3784    1
## p[3]     0.36    0.00 0.03    0.29    0.34    0.36    0.38    0.43  3901    1
## p[4]     0.61    0.00 0.03    0.54    0.58    0.61    0.63    0.67  2892    1
## p[5]     0.80    0.00 0.03    0.74    0.78    0.80    0.81    0.85  2261    1
## p[6]     0.90    0.00 0.02    0.86    0.89    0.90    0.92    0.94  2157    1
## p[7]     0.96    0.00 0.01    0.93    0.95    0.96    0.96    0.98  2175    1
## p[8]     0.98    0.00 0.01    0.96    0.97    0.98    0.98    0.99  2215    1
## lp__  -187.26    0.03 1.06 -190.16 -187.65 -186.94 -186.52 -186.26  1647    1
##
## Samples were drawn using NUTS(diag_e) at Tue Mar  5 14:56:53 2024.
## 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).
```

```
plot(model_fit2)
```

```
## ci_level: 0.8 (80% intervals)
```

```
## outer_level: 0.95 (95% intervals)
```




```
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]);
  }
}
```

```
# Realiza el muestreo
model_fit3 <- rstan::sampling(modelo_p1_probit, data = data_input)
```

```
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 2e-06 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0.02 seconds.
## Chain 1: Adjust your expectations accordingly!
## Chain 1:
## Chain 1:
## Chain 1: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 1: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 1: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 1: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 1: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 1: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 1: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 1: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 1: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 1: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 1: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 1: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 0.007 seconds (Warm-up)
## Chain 1:                0.005 seconds (Sampling)
## Chain 1:                0.012 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 2).
## Chain 2:
## Chain 2: Gradient evaluation took 1e-06 seconds
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0.01 seconds.
## Chain 2: Adjust your expectations accordingly!
## Chain 2:
## Chain 2:
## Chain 2: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 2: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 2: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 2: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 2: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 2: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 2: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 2: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 2: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 2: Iteration: 1600 / 2000 [ 80%]  (Sampling)
```

```
## Chain 2: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 2: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 2:
## Chain 2:  Elapsed Time: 0.007 seconds (Warm-up)
## Chain 2:                0.005 seconds (Sampling)
## Chain 2:                0.012 seconds (Total)
## Chain 2:
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 3).
## Chain 3:
## Chain 3: Gradient evaluation took 0 seconds
## Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0 seconds.
## Chain 3: Adjust your expectations accordingly!
## Chain 3:
## Chain 3:
## Chain 3: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 3: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 3: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 3: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 3: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 3: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 3: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 3: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 3: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 3: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 3: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 3: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 3:
## Chain 3:  Elapsed Time: 0.008 seconds (Warm-up)
## Chain 3:                0.006 seconds (Sampling)
## Chain 3:                0.014 seconds (Total)
## Chain 3:
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 4).
## Chain 4:
## Chain 4: Gradient evaluation took 1e-06 seconds
## Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 0.01 seconds.
## Chain 4: Adjust your expectations accordingly!
## Chain 4:
## Chain 4:
## Chain 4: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 4: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 4: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 4: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 4: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 4: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 4: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 4: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 4: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 4: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 4: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 4: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 4:
## Chain 4:  Elapsed Time: 0.007 seconds (Warm-up)
```

```
## Chain 4:                0.006 seconds (Sampling)
## Chain 4:                0.013 seconds (Total)
## Chain 4:
```

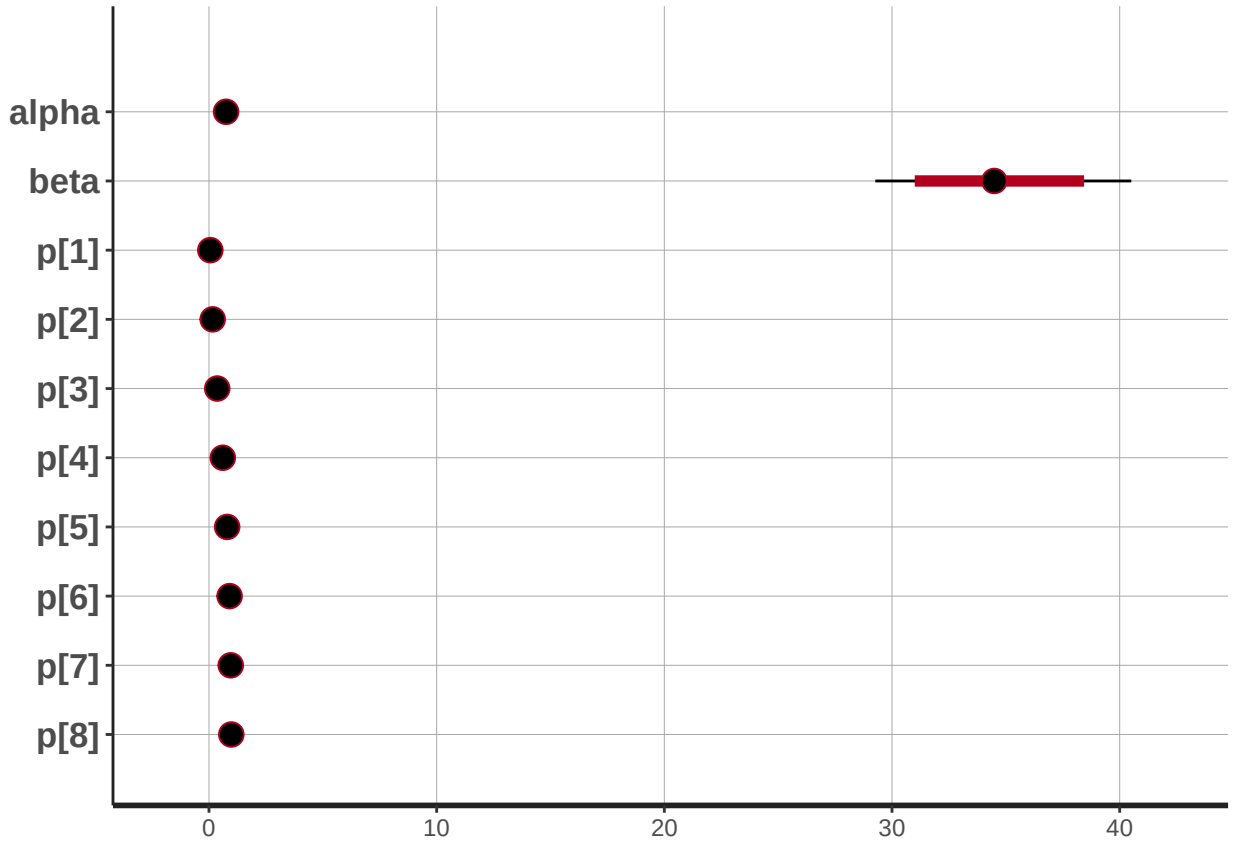
```
# Interpretaciones
print(model_fit3)
```

```
## Inference for Stan model: anon_model.
## 4 chains, each with iter=2000; warmup=1000; thin=1;
## post-warmup draws per chain=1000, total post-warmup draws=4000.
##
##          mean se_mean   sd    2.5%     25%     50%     75%   97.5% n_eff Rhat
## alpha    0.75    0.00 0.14    0.48    0.66    0.75    0.85    1.02  2698    1
## beta    34.61    0.06 2.91   29.27   32.57   34.49   36.50   40.51  2528    1
## p[1]     0.06    0.00 0.02    0.03    0.05    0.06    0.07    0.09  3126    1
## p[2]     0.16    0.00 0.03    0.11    0.14    0.16    0.18    0.22  3552    1
## p[3]     0.36    0.00 0.03    0.29    0.34    0.36    0.38    0.43  3786    1
## p[4]     0.61    0.00 0.03    0.54    0.58    0.61    0.63    0.67  2960    1
## p[5]     0.80    0.00 0.03    0.74    0.78    0.80    0.81    0.84  2426    1
## p[6]     0.90    0.00 0.02    0.86    0.89    0.90    0.92    0.94  2328    1
## p[7]     0.96    0.00 0.01    0.93    0.95    0.96    0.96    0.97  2338    1
## p[8]     0.98    0.00 0.01    0.96    0.97    0.98    0.98    0.99  2366    1
## lp__  -187.25    0.02 1.00 -189.94 -187.65 -186.97 -186.53 -186.26  1870    1
##
## Samples were drawn using NUTS(diag_e) at Tue Mar  5 14:56:53 2024.
## 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).
```

```
plot(model_fit3)
```

```
## ci_level: 0.8 (80% intervals)
```

```
## outer_level: 0.95 (95% intervals)
```



2. Consideren las siguientes dos distribuciones condicionales completas, analizadas en el artículo de Casella y George (1992) que les incluí como lectura:

$$f(x|y) \propto ye^{-yx} \quad 0 < x < B < \infty$$

$$f(y|x) \propto xe^{-xy} \quad 0 < y < B < \infty$$

3. Consideren las siguientes dos distribuciones condicionales completas, analizadas en el artículo de Casella y George (1992) que les incluí como lectura:

$$f(x|y) \propto ye^{-yx} \quad 0 < x < B < \infty$$

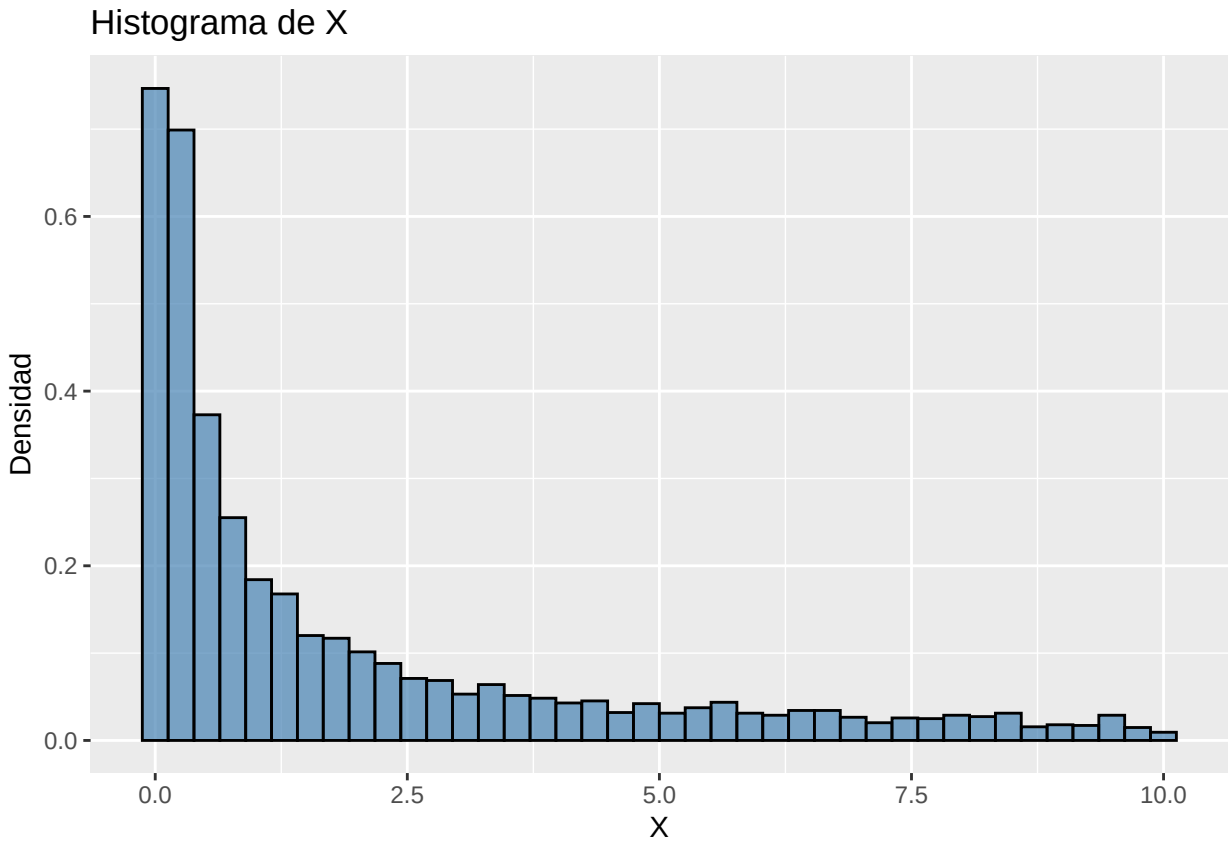
$$f(y|x) \propto xe^{-xy} \quad 0 < y < B < \infty$$

- Obtener un estimado de la distribución marginal de X cuando $B = 10$ usando el Gibbs sampler.

```
X = rep(0, 5000)
Y = rep(0, 5000)

k = 30

for (i in 1:5000) {
  x = rep(1, k)
  y = rep(1, k)

  for (j in 2:k) {
    temp_x = 11
    while(temp_x > 10) {
      x[j] = rexp(1, y[j-1])
```

```
      temp_x = x[j]
    }

    temp_y = 11
    while(temp_y > 10) {
      y[j] = rexp(1, x[j])
      temp_y = y[j]
    }
  }

  X[i] = x[k]
  Y[i] = y[k]
}

ggplot(data = data.frame(X), aes(x = X)) +
  geom_histogram(bins = 40, aes(y = ..density..), color = "black", fill = "steelblue", alpha = 0.7) +
  ggtitle("Histograma de X") +
  xlab("X") +
  ylab("Densidad")
```

```
## Warning: The dot-dot notation (`..density..`) was deprecated in ggplot2 3.4.0.
## i Please use `after_stat(density)` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```



- Ahora supongan que $B = \infty$ así que las distribuciones condicionales completas son ahora las ordinarias

distribuciones exponenciales no truncadas. Mostrar analíticamente que $f_x(t) = 1/t$ es una solución a la ecuación integral en este caso:

$$f_x(x) = \int \left[\int f_{x|y}(x|y) f_{y|t}(y|t) dy \right] f_x(t) dt$$

¿El Gibbs sampler convergerá a esta solución?

Usando la función dada:

$$\begin{aligned} f_x(x) &= \int \left[\int f_{x|y}(x|y) f_{y|t}(y|t) dy \right] f_x(t) dt \\ &= \int \left[\int y e^{-yx} t e^{-ty} dy \right] f_x(t) dt \\ &= \int t \left[\int y e^{-yx} t e^{-ty} dy \right] f_x(t) dt \end{aligned}$$

Substituyendo $f_x(t) = \frac{1}{t}$ tenemos:

$$\begin{aligned} &= \int \left[\frac{t}{(x+t)^2} \right] \frac{1}{t} dt \\ &= \frac{1}{x} \end{aligned}$$

Aunque esta es una solución, 1/x no es una función de densidad. Cuando se aplica el muestreador de Gibbs a las densidades condicionales, la convergencia se rompe. No da una aproximación a 1/x, de hecho, no obtenemos una muestra de variables aleatorias de una distribución marginal.

3. Supongan que una variable aleatoria y se distribuye de acuerdo a la densidad poli-Cauchy:

$$g(y) = \prod_{i=1}^{n} \frac{1}{\pi(a + (y - a_1)^2)}$$

donde $a = (a_1, ..., a_n)$ es un vector de parámetros. Supongan que $n = 6$ y $a = (1, 2, 3, 6, 7, 8)$.

- Escriban una función que calcule la log-densidad de y.
- Escriban una función que tome una muestra de tamaño 10,000 de la densidad de y, usando Metropolis-Hastings con función propuesta una caminata aleatoria con desviación estandar C. Investiguen el efecto de la elección de C en la tasa de aceptación, y la mezcla de la cadena en la densidad.
- Usando la muestra simulada de una "buena" elección de C, aproximar la probabilidad $P(6 < Y < 8)$.

4. Supongan que el vector (X, Y) tiene función de distribución conjunta:

$$f(x, y) = \frac{x^{a+y-1} e^{-(1+b)x} b^a}{y! \Gamma(a)}, \quad x > 0, \quad y = 0, 1, 2, ...$$

y deseamos simular de la densidad conjunta.

- Mostrar que la densidad condicional $f(x|y)$ es una Gamma e identificar los parámetros.

Para encontrar la densidad condicional $f(x|y)$, primero necesitamos encontrar la densidad marginal de Y,$f_Y(y)$ ya que la densidad condicional se define como:

$$f(x|y) = \frac{f(x, y)}{f_Y(y)}$$

Dada la función de distribución conjunta, podemos encontrar la densidad marginal de Y integrando sobre todos los posibles varores de x:

$$f_Y(y) = \int_0^\infty f(x, y) dx$$

$$= \int_0^\infty \frac{x^{a+y-1}e^{-(1+b)x}b^a}{y!\Gamma(a)}dx$$

Sacando los terminos que no dependen de x de la integral, tenemos:

$$= \frac{b^a}{y!\Gamma(a)} \int_0^\infty x^{a+y-1}e^{-(1+b)x}dx$$

Recordemos que la función Gamma se define como:

$$f_X(x) = \frac{\lambda}{\Gamma(\alpha)}(\lambda x)^{\alpha-1}e^{-\lambda}$$

donde

$$\Gamma(\alpha) = \int_0^\infty t^{\alpha-1}e^{-t}dt$$

Y la de densidad acumulada es:

$$F_X(x) = \int_0^x \frac{\lambda}{\Gamma(\alpha)}(\lambda y)^{\alpha-1}e^{-\lambda y}\ dy$$

La integral que tenemos esta incompleta por lo que agregaremos un 1 para completar una Gamma con parámetros $(a+y)$ y $(1+b)$:

$$= \frac{b^a}{y!\Gamma(a)}\Gamma(a+y) \int_0^\infty \frac{x^{a+y-1}e^{-(1+b)x}}{\Gamma(a+y)}dx$$

$$f_Y(y) = \frac{b^a\Gamma(a+y)}{y!\Gamma(a)}$$

Ahora podemos encontrar la densidad condicional $f(x|y)$:

$$f(x|y) = \frac{f(x,y)}{f_Y(y)} = \frac{\frac{x^{a+y-1}e^{-(1+b)x}b^a}{y!\Gamma(a)}}{\frac{b^a\Gamma(a+y)}{y!\Gamma(a)}}$$

Simplificamos y obtenemos:

$$f(x|y) = \frac{x^{a+y-1}e^{-(1+b)x}}{\Gamma(a+y)}$$

Identificamos que la distribución condicional $f(x|y)$ es una distribución Gamma con parámetros $\alpha = (a+y)$ y $\beta = (1+b)$

- Mostrar que la densidad condicional $f(y|x)$ es Poisson.

  Para encontrar la densidad condicional $f(y|x)$, primero necesitamos encontrar la densidad marginal de X,$f_X(x)$ ya que la densidad condicional se define como:

$$f(y|x) = \frac{f(x,y)}{f_X(x)}$$

  Dada la función de distribución conjunta, podemos encontrar la densidad marginal de X sumando para todos los posibles varores de y:

$$f_X(x) = \sum_{y=0}^{\infty} f(x,y)$$

$$= \sum_{y=0}^{\infty} \frac{x^{a+y-1}e^{-(1+b)x}b^a}{y!\Gamma(a)}$$

Se sacan de la sumatoria los terminos que no depende de y, quedando:

$$= \frac{e^{-bx}b^a}{\Gamma(a)} \sum_{y=0}^{\infty} \frac{e^{-x}x^{a+y-1}}{y!}$$

$$= \frac{e^{-bx}b^a}{\Gamma(a)} \sum_{y=0}^{\infty} e^{-x} \left(\frac{x^{a+y-1}}{y!} \right)$$

$$= \frac{e^{-bx}b^a}{\Gamma(a)} e^{-x} \sum_{y=0}^{\infty} x^{a-1} \frac{x^y}{y!}$$

$$= \frac{e^{-bx}b^a}{\Gamma(a)} e^{-x} e^{x} x^{a-1}$$

$$f_X(x) = \frac{e^{-bx}b^a}{\Gamma(a)} x^{a-1}$$

Entonces la densidad condicional de $f(y|x)$ es:

$$f(y|x) = \frac{f(x,y)}{f_X(x)} = \frac{\frac{x^{a+y-1}e^{-(1+b)x}b^a}{y!\Gamma(a)}}{\frac{e^{-bx}b^a}{\Gamma(a)}x^{a-1}}$$

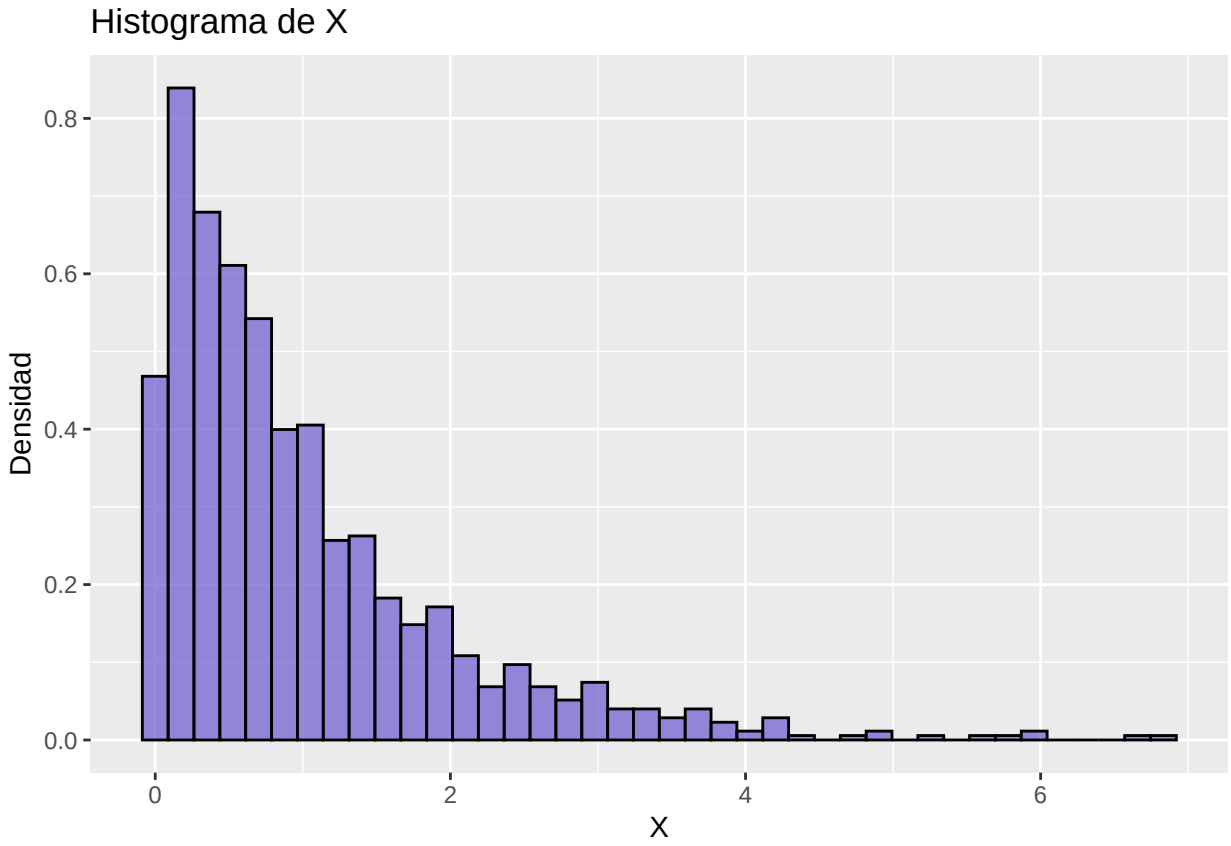
$$f(y|x) = \frac{e^{-x}x^y}{y!} \quad \sim Possion(x)$$

- Escriban una función para implementar el Gibbs sampler cuando las constantes son dadas con valores $a = 1$ y $b = 1$.

```
fx_y <- function(x, y, a, b){ return(dgamma(x, shape = (a + y), rate = (1 + b))) }

fy_x <- function(y, x){ return(dpois(y, lambda = x)) }

gibbs <- function(n, a, b){
  x = rep(0, n)
  y = rep(0, n)
  x[1] = 1
  y[1] = 1
  for(i in 2:n){
    x[i] = rgamma(1, shape = (a + y[i-1]), rate = (1 + b))
    y[i] = rpois(1, lambda = x[i])
  }
  return(data.frame(x, y))
}
```

- Con su función, escriban 1,000 ciclos del Gibbs sampler y de la salida, hacer los histogramas y estimar $E(Y)$.

```
res4 <- gibbs(1000, 1, 1)

ggplot(data = res4, aes(x = x)) +
  geom_histogram(bins = 40, aes(y = ..density..), color = "black", fill = "slateblue3", alpha = 0.7) +
  ggtitle("Histograma de X") +
  xlab("X") +
  ylab("Densidad")
```




```
E_Y <- data.frame(E_Y = mean(res4$y))
E_Y |> gt()
```

E_Y
0.986

5. Supongan que se tiene una matriz de 4Ö4 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	$N(X)$	$D(X)$
$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$	1	2
$\begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$	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 2^{16} posibles estados, uno por cada posible valor de X. a) Sea q_1 tal que cada transición es igualmente plausible con probabilidad $1/2^{16}$. 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's 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's 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 la otra? ¿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.

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 $log(\theta)$.

a) Definan los datos enWinBUGS (u OpenBUGS). Usen $\theta = 1$ como valor inicial.
b) Escriban el código para el modelo.
c) Compilen el modelo y obtengan una muestra de 1000 iteraciones después de descartar las 500 iteraciones iniciales como burnin.
d) Monitoreen la convergencia gráficamente usando gráficas 'trace' y de autocorrelación.
e) Obtengan estadísticas sumarias posteriores y densidades para θ, $1/\theta$ y $log(\theta)$
f) 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.