EST-46115: Modelación Bayesiana

Profesor: Alfredo Garbuno Iñigo — Primavera, 2023 — Flujo de trabajo bayesiano.
Objetivo: Ya hemos estudiado la herramienta por excelencia de cómputo bayesiano. Ahora, ilustraremos con ejemplos casos de aplicación con un enfoque bayesiano. Estableceremos las bases para continuar nuestro proceso de aprendizaje a temas que se requerirá estudiar con mayor cuidado para navegar el camino de un modelado iterativo.
Lectura recomendada: El Capítulo 9 de [5] muestra un caso del proceso iterativo de modelado bayesiano. El artículo [4] busca explorar los diversos componentes del proceso de construcción de modelos en los cuales profundizaremos en esta segunda parte del curso.

1. INTRODUCCIÓN

El desarrollo de algoritmos de muestreo nos permiten `explorar computacionalmente` una distribución de probabilidad de interés $\pi(\cdot)$. En el contexto bayesiano $\pi(\cdot)$ denota la distribución (previa o posterior) para un problema de inferencia, donde queremos reportar resúmenes

$$\pi(f) = \mathbb{E}[f(\theta)] = \int_{\Theta} f(\theta)\, \pi(\theta|y)\, \mathrm{d}\theta\,. \tag{1}$$

Para resolver un problema de modelado en el contexto bayesiano debemos de tener en mente los distintos componentes del `modelo` y las `herramientas computacionales`.

- Los `datos` con los que contamos (o podemos contar), y.
- Nuestra `abstracción del modelo generativo`, $\pi(y|\theta)$.
- Nuestra matematización de nuestro `conocimiento sobre el problema`, $\pi(\theta)$.
- Los `resúmenes` que reportaremos, $f(\theta)$ ó $f(\hat{y})$.
- El `mecanismo computacional` para resolver integrales, $\int_{\Theta} \cdot\, \pi(\theta|y)\, \mathrm{d}\theta$.

Exploraremos la idea general de un marco de trabajo bayesiano para después enfocarnos en cada uno de los componentes del proceso. Esto con el objetivo de entender los pasos de este proceso iterativo.

En general cuenta con tres pasos:

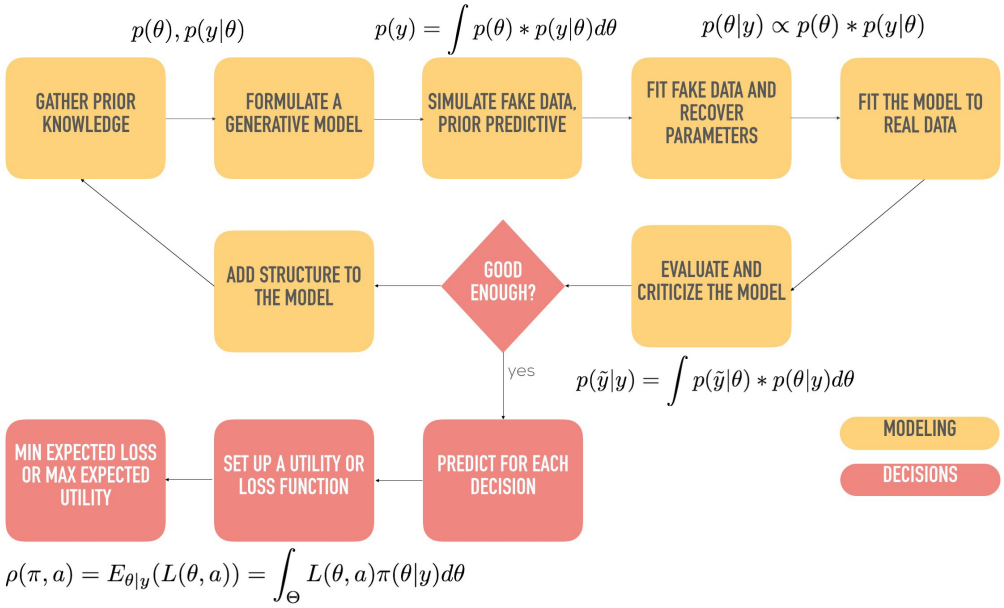
1. Inferencia.
2. Exploración y mejora de modelos.
3. Comparación de modelos.

En este contexto `no necesariamente queremos escoger el mejor modelo`. Lo que buscamos es generar un `mejor entendimiento del modelo` y esto lo podemos lograr al evaluar las bondades de uno sobre otro. El cómputo asociado con modelación bayesiana es muy complejo y puede tomar varias iteraciones en lo que estamos seguros de lo que esta realizando. Es decir, nos puede tomar varios pasos estar seguros que podemos confiar en nuestro modelo en relación a los datos que estamos analizando.

Lo importante es considerar la `pregunta objetivo`. Es decir, encontrar la pregunta que esperamos nuestro análisis pueda resolver. Por supuesto en el camino encontraremos preguntas sobre los datos, los modelos, la inferencia. Tener en mente la respuesta que queremos proveer nos permitirá definir muchos de los componentes del flujo que seguirá nuestro trabajo.

Es decir, nos permitirá acotar la colección de modelos apropiados, cómo escoger previas, qué esperar de la distribución posterior, cómo seleccionar entre modelos, qué reportar, cómo resumir la información de nuestro modelo, qué conclusiones comunicar.

> Do not be the Data Scientist and statistician that immediately reaches for Bayesian Methods, Neural Networks, Distributed Computing Clusters, or other complex tools before truly *understanding the need*. —Martin et al. [5].



2. EJEMPLO: DATOS DE CONTEO

Con este ejemplo veremos la aplicación de modelos bayesianos en el contexto de *customer service*. Los datos disponibles son *tweets* sobre reclamos a compañías y además contamos con los *tweets* de respuesta para dichas reclamaciones. Nos concentraremos en compañías áreas pero se puede extender el modelo a otras. El objetivo de este análisis es poder definir qué compañía responde a mayor número de *tweets* por reclamaciones de manera diaria.

La variable de interés es el número de mensajes atendidos diario (`atendidos`) y nos concentraremos en el periodo de Octubre y Noviembre del 2017.

```
1 reclamos ▷ print(n = 3)
```

Donde vemos que el número promedio de reclamos es el siguiente.

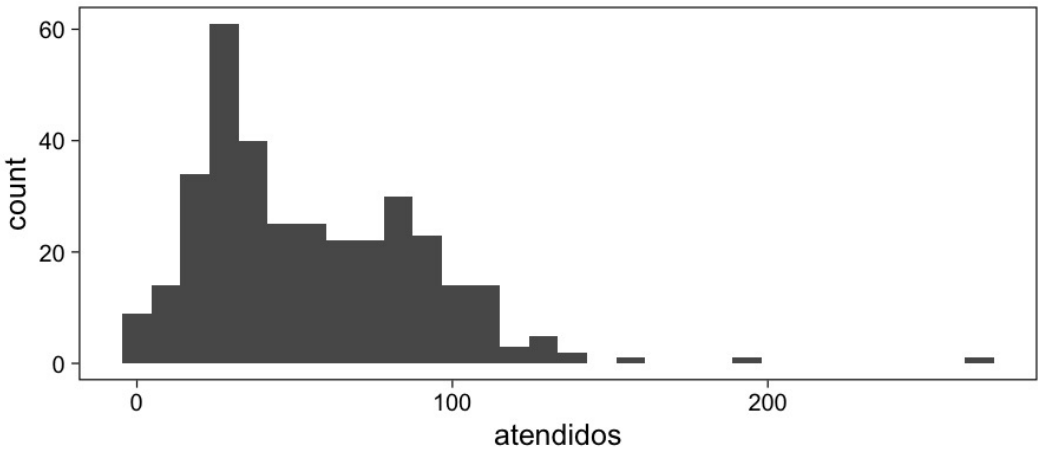


FIGURA 1. *Conteos de reclamos diarios atendidos por las aerolíneas.*

```
# A tibble: 1 × 1
  promedio
     <dbl>
1     54.9
```

2.1. Primer modelo

Dado que nuestro objetivo es modelo el `número de reclamos atendidos` lo natural es pensar en una variable *aleatoria de conteo*. Dentro de las opciones naturales tenemos como candidatos:

- `Binomial`: el número de éxitos que suceden con una tasa θ dentro de una colección de n observaciones.
- `Poisson`: el número de eventos que suceden con una tasa de ocurrencia λ.

El modelo `poisson` es el candidato natural para este modelo. El cual tiene una función de masa de probabilidad igual a

$$\mathbb{P}(X = k|\lambda) = \frac{\lambda^k e^{-\lambda}}{k!}, \tag{2}$$

cuando $X|\lambda \sim \mathsf{Poisson}(\lambda)$ con $\lambda > 0$.

Dada nuestra ignorancia sobre el problema escogemos una distribución previa para λ como una $\mathsf{Gamma}(100, 2)$ que es una distribución inicial `poco informativa`. En promedio con esta distribución esperamos 50 reclamos atendidos por twitter al día con una dispersión de 25 reclamos.

El modelo lo escribimos como sigue:

```
data {
  int N;
  int y[N];
}

parameters {
  real<lower=0> lambda;
}
```

```

model {
  lambda ~ gamma(100, 2);
  y ~ poisson(lambda);
}
```

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

Lo utilizamos para muestrear de la previa y la posterior

```
data_list ← list(N = nrow(reclamos), y = reclamos$atendidos)
previa ← modelo$sample(data = list(N = 0, y = c()), refresh = 0)
posterior ← modelo$sample(data = data_list, refresh = 0)
```

> Nota que para muestrear de la previa utilizamos un `bloque de datos vacío`.

Podemos extraer resúmenes como sigue

```
# A tibble: 2 × 10
  variable    mean   median    sd   mad       q5      q95  rhat ess_bulk ess_tail
  <chr>      <dbl>    <dbl> <dbl> <dbl>    <dbl>    <dbl> <dbl>    <dbl>    <dbl>
1 lp__     57356.   57356.   0.680 0.297 57354.   57356     1.00     1646.     2311.
2 lambda      54.9     54.9  0.396 0.399    54.2     55.5   1.00     1451.     1855.
```

Y finalmente podemos crear histogramas para comparar la distribución previa contra la posterior sobre el parámetro de interés, ver Fig. 2.

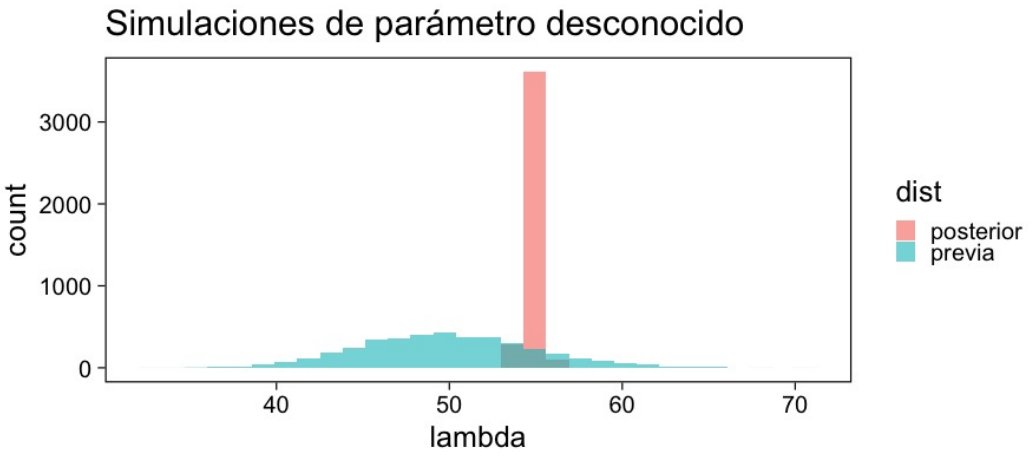


FIGURA 2. *Histograma de λ bajo la distribución previa (azul) y posterior (salmón).*

También podemos obtener histogramas de la distribución predictiva (previa y posterior) para comparar las inferencias sobre `observables` bajo nuestro modelo, ver Fig. 3.

Finalmente, hacemos una comparación con el histograma de los datos.

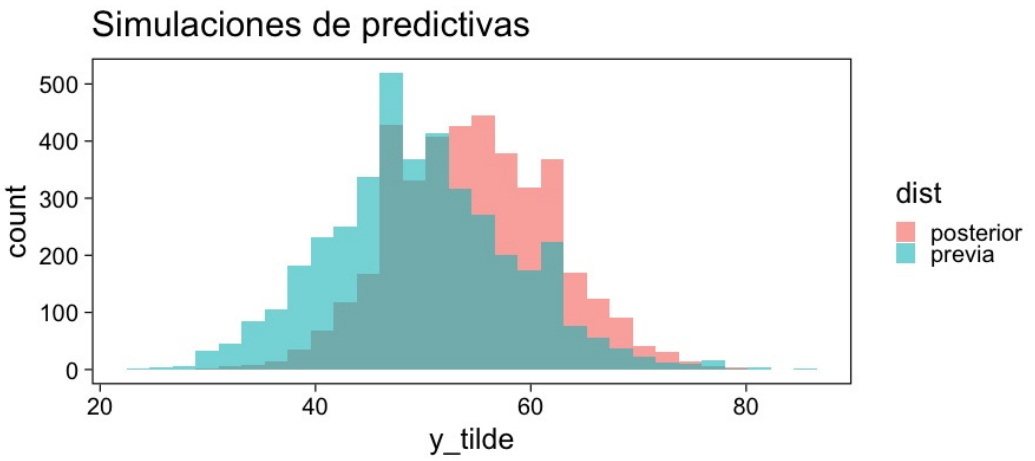


FIGURA 3. *Histogramas de observaciones hipotéticas del modelo bajo la distribución previa (azul) y posterior (salmón).*

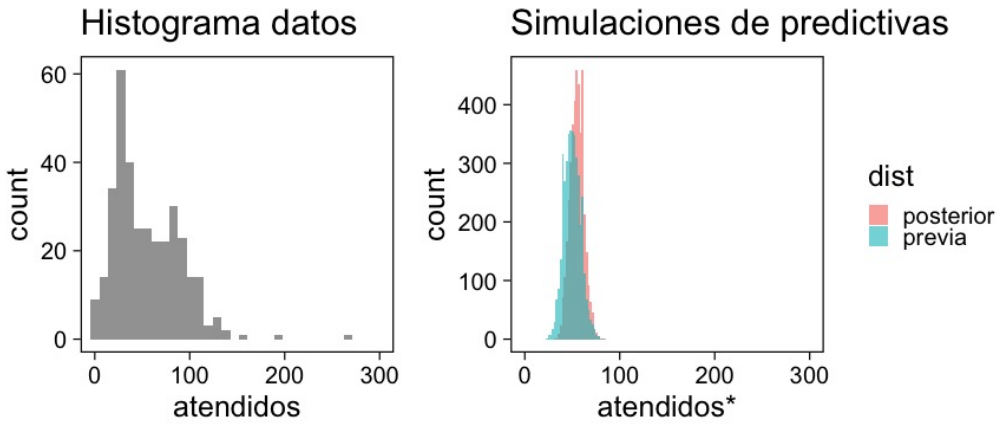


FIGURA 4. *Histogramas de observaciones hipotéticas del modelo bajo la distribución previa (azul) y posterior (salmón).*

2.1.1. Conclusiones: Lo que observamos en Fig. 4, recurrir a `sobre-dispersión` es un `fenómeno muy común` con modelos de conteo.

En promedio nuestras estimaciones funcionan bien. La media posterior es cercana al estimador de máxima verosimilitud.

Sin embargo, el modelo no es capaz de controlar la variabilidad de las observaciones. Esto es por que bajo un modelo Poisson la media y varianza están controladas con el mismo parámetro. Cuando esto sucede –los datos tienen mayor variabilidad que la sugerida por un modelo Poisson (o Binomial)– hablamos de datos de conteo con `sobre-dispersión`.

2.2. Sobre-dispersión

Calculemos la media y varianza de nuestro datos:

```
# A tibble: 1 × 2
  promedio varianza
     <dbl>    <dbl>
1     54.9    1230.
```

Claramente no podremos modelar la variabilidad de nuestros datos con un modelo Poisson. Así que necesitamos buscar qué distribución es la adecuada para nuestro problema (ver Sección 17.2 de [3]). Una variable aleatoria `Binomial Negativa` es la sugerida donde la función de masa de probabilidad está definida por

$$\mathbb{P}(Y = k|\alpha, \beta) = \binom{k+\alpha-1}{\alpha-1} \left(\frac{\beta}{\beta+1}\right)^{\alpha} \left(\frac{1}{\beta+1}\right)^{k}, \tag{3}$$

donde $Y|\alpha, \beta \sim \mathsf{NegBinom}(\alpha, \beta)$ y cuyos estadísticos básicos están definidos como

$$\mathbb{E}[Y] = \frac{\alpha}{\beta}, \qquad \mathbb{V}(Y) = \frac{\alpha}{\beta^2}(\beta+1)\,. \tag{4}$$

Lo cual es informativo, pero poco útil para generar un poco de intuición. Asi que optamos por una segunda `parametrización` (Neg-Binom) la cual tiene como masa de probabilidad

$$\mathbb{P}(Y = k|\mu, \phi) = \binom{k+\phi-1}{k} \left(\frac{\mu}{\mu+\phi}\right)^{k} \left(\frac{\phi}{\mu+\phi}\right)^{\phi}, \tag{5}$$

donde $Y|\mu, \phi \sim \mathsf{NegBinom}(\mu, \phi)$ y cuyos estadísticos básicos están definidos como

$$\mathbb{E}[Y] = \mu, \qquad \mathbb{V}(Y) = \mu + \frac{\mu^2}{\phi}\,, \tag{6}$$

donde $\mu > 0$ es el número esperado de casos y $\phi > 0$ controla el factor adicional de dispersión de la `binomial negativa`. Al parámetro ϕ le llamamos precisión del modelo. Si ϕ es pequeño entonces el modelo tiene que compensar más con sobre-dispersión para los conteos.

> Para entender una conexión adicional entre la `binomial negativa` y la `poisson` pensemos en que si marginalizamos una $\mathsf{Poisson}(Y|\lambda)$ con respecto a una $\mathsf{Gamma}(\lambda|\alpha, \beta)$ obtenemos la `binomial negativa`. Esto quiere decir que el componente adicional de dispersión del modelo `poisson` es el resultado de marginalizar bajo distintas configuraciones provenientes de una `gamma` el parámetro que controla la media del modelo de conteo. Es por esto que también a una binomial negativa se le conoce como `Poisson-Gamma` [6].

Escribimos el modelo donde igual que antes utilizamos una distribución previa sobre el parámetro adicional poco informativa. En este momento lo que queremos es probar si podemos ajustar este modelo a nuestros datos.

```
data {
  int N;
  int y[N];
}

parameters {
  real<lower=0> lambda;
  real<lower=0> phi;
}

model {
  lambda ~ normal(50, 10);
```

```
  phi    ~ gamma(1, 1);
  y      ~ neg_binomial_2(lambda, phi);
}

generated quantities {
  int y_tilde = neg_binomial_2_rng(lambda, phi);
}
```

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

```
data_list ← list(N = nrow(reclamos), y = reclamos$atendidos)
previa ← modelo$sample(data = list(N = 0, y = c()), refresh = 0)
posterior ← modelo$sample(data = data_list, refresh = 500, seed = 108727)
```

Vemos algunas alertas en el ajuste de la posterior. Las cuales podemos explorar mejor utilizando la opción `refresh` del muestreador. Con esto vemor que las alertas suceden en el periodo de calentamiento del muestreador. Podemos ver los resúmenes y ver que efectivamente parece no haber problemas con el ajuste.

```
# A tibble: 4 × 10
  variable    mean   median      sd     mad       q5      q95  rhat ess_...¹b
  <chr>      <dbl>    <dbl>   <dbl>   <dbl>    <dbl>    <dbl> <dbl>    <dbl>
1 lp__     -1.69e3 -1.69e3   0.943   0.712 -1.69e3  -1.68e3   1.00    1988.
2 lambda    5.49e1  5.48e1   1.94    1.98   5.18e1   5.80e1   1.00    3504.
3 phi       2.23e0  2.23e0   0.169   0.172  1.96e0   2.51e0   1.00    3577.
4 y_tilde   5.52e1  4.7 e1  38.1    32.6    1.10e1   1.29e2   1.00    3938.
# ... with 1 more variable: ess_tail <dbl>, and abbreviated variable
#   name ¹ess_bulk
#  Use ‘colnames()‘ to see all variable names
```

2.3. Reparametrizando

Posiblemente nos sintamos incómodos por las alertas asi que podemos buscar una parametrización alternativa del modelo. Como siempre, buscamos ayuda y encontramos que se puede parametrizar distinto con $\log \mu$ dadas las inicializaciones del modelo. Al tener un modelo previo normal truncado para λ es natural pensar que podemos asumir una distribución para $\log \lambda$ como alternativa.

Es buen momento para refinar la distribución previa de los demás parámetros (en este caso ϕ).

2.3.1. Calibración de previa: Parte de las alertas tienen que ver con la restricción misma del modelo. Asi que podemos utilizar `Stan` para `elicitar` (proceso de calibración de una distribución de probabilidad) la previa.

```
functions {
  // Diferencias para las colas de una Gamma
  vector tail_delta(vector y, vector theta, real[] x_r, int[] x_i) {
    vector[2] deltas;
    deltas[1] = gamma_cdf(theta[1] | exp(y[1]), exp(y[2])) - 0.005;
    deltas[2] = 1 - gamma_cdf(theta[2] | exp(y[1]), exp(y[2])) - 0.005;
    return deltas;
```

```
  }
}

transformed data {
  vector[2] y_guess = [log(9), log(0.5)]';//Valores␣iniciales
␣␣vector[2]␣theta␣=␣[1.5,␣15]';           //Cotas del intervalo
  vector[2] y;
  real x_r[0];
  int x_i[0];

  // Encuentra los parametros de la Gamma para satisfacer que
  // con 1% de probabilidad estemos en el intervalo [0.5, 20]
  y = algebra_solver(tail_delta, y_guess, theta, x_r, x_i);

  print("alpha = ", exp(y[1]));
  print("beta = ", exp(y[2]));
}

generated quantities {
  real alpha = exp(y[1]);
  real beta = exp(y[2]);
}
```

```
solucion ← modelo$sample(iter = 1, iter_warmup = 0,
                         chains = 1, fixed_param = TRUE)
previa.params ← solucion$draws(format = "df")
previa.params
```

```
Running MCMC with 1 chain...

Chain 1 alpha = 5.61803
Chain 1 beta = 0.904107
Chain 1 Iteration: 1 / 1 [100%]  (Sampling)
Chain 1 finished in 0.0 seconds.
# A draws_df: 1 iterations, 1 chains, and 2 variables
  alpha beta
1   5.6  0.9
# ... hidden reserved variables {'.chain', '.iteration', '.draw'}
```

Listing 1. *Resultados de la elicitación.*

2.4. Modelo jerárquico

No lo hemos mencionado pero por lo que vemos parece existir la presencia de un proceso adicional de generación de reclamos no explicado (dos modas aparentes en el histograma). `Una solución` posible es considerar distintos grupos dentro de la población de reclamos y modelar lso reclamos de manear jerárquica de acuerdo a la aerolínea.

```
# A tibble: 6 × 5
  compania        promedio varianza exceso precision
  <fct>              <dbl>    <dbl>  <dbl>     <dbl>
1 AlaskaAir           83.6    1341. 0.180       5.56
2 VirginAtlantic      36.4     186. 0.113       8.88
3 British_Airways     46.4     734. 0.319       3.14
4 JetBlue             85.2     637. 0.0761     13.1
5 VirginAmerica       32.3     154. 0.117       8.55
```

```
6 AirAsiaSupport      41.6    1718.  0.968       1.03
```

Listing 2. *Estadisticos por aerolínea.*

Vemos que las aerolíneas tienen descriptivos distintos entre ellas. Asi que optaremos por un modelo que utilice una estructura por niveles.

```
data {
  int N;
  int y[N];
  int compania[N];
  real<lower=0> gamma_alpha;
  real<lower=0> gamma_beta;
}

parameters {
  real log_lambda[6];
  real<lower=0> phi[6];
}

model {
  log_lambda ~ normal(4, 0.5);
  phi    ~ gamma(gamma_alpha, gamma_beta);
  y      ~ neg_binomial_2_log(log_lambda[compania], phi[compania]);
}

generated quantities {
  int y_tilde[6];
  for (ii in 1:6){
    y_tilde[ii] = neg_binomial_2_log_rng(log_lambda[ii], phi[ii]);
  }
}
```

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

```
data_list ← list(N = nrow(reclamos),
                 y = reclamos$atendidos,
                 compania = as.numeric(reclamos$compania),
                 gamma_alpha = previa.params$alpha,
                 gamma_beta  = previa.params$beta)
posterior ← modelo$sample(data = data_list, refresh = 500, seed = 108727)
```

El modelo parece ajustar bien y podemos explorar los diagnósticos.

```
# A tibble: 19 × 10
   variable      mean   median       sd      mad        q5      q95  rhat ess_...¹b
   <chr>        <dbl>    <dbl>    <dbl>    <dbl>     <dbl>    <dbl> <dbl>    <dbl>
 1 lp__       -1.56e3 -1.56e3  2.50     2.40    -1.57e+3 -1.56e3  1.00    1499.
 2 log_...lamb  4.42e0  4.42e0  0.0619   0.0591   4.32e+0  4.52e0  1.00    8523.
 3 log_...lamb  3.60e0  3.60e0  0.0558   0.0560   3.51e+0  3.69e0  1.00    7864.
 4 log_...lamb  3.85e0  3.85e0  0.0788   0.0796   3.72e+0  3.98e0  1.00    6136.
 5 log_...lamb  4.44e0  4.44e0  0.0512   0.0513   4.36e+0  4.53e0  1.00    7305.
 6 log_...lamb  3.48e0  3.48e0  0.0484   0.0488   3.40e+0  3.56e0  1.00    7819.
 7 log_...lamb  3.76e0  3.75e0  0.130    0.128    3.54e+0  3.98e0  1.00    7997.
```

```
 8 phi[1]      4.52e0   4.46e0   0.834    0.811    3.27e+0   6.04e0   1.00    7244.
 9 phi[2]      6.39e0   6.29e0   1.33     1.28     4.43e+0   8.76e0   1.00    7781.
10 phi[3]      2.70e0   2.66e0   0.499    0.479    1.94e+0   3.59e0   1.00    7090.
11 phi[4]      7.55e0   7.45e0   1.47     1.45     5.34e+0   1.01e1   1.00    8263.
12 phi[5]      9.44e0   9.29e0   1.92     1.89     6.57e+0   1.29e1   1.00    9427.
13 phi[6]      1.37e0   1.35e0   0.259    0.251    9.83e-1   1.83e0   1.00    9598.
14 y_tilde...[  8.33e1   7.6 e1  42.1     38.5      2.8 e+1   1.62e2   1.00    3871.
15 y_tilde...[  3.66e1   3.4 e1  16.5     14.8      1.4 e+1   6.6 e1   1.00    3861.
# ... with 4 more rows, 1 more variable: ess_tail <dbl>, and abbreviated
#   variable name ¹ess_bulk
#  Use 'print(n = ...)' to see more rows, and 'colnames()' to see all variable
    names
```

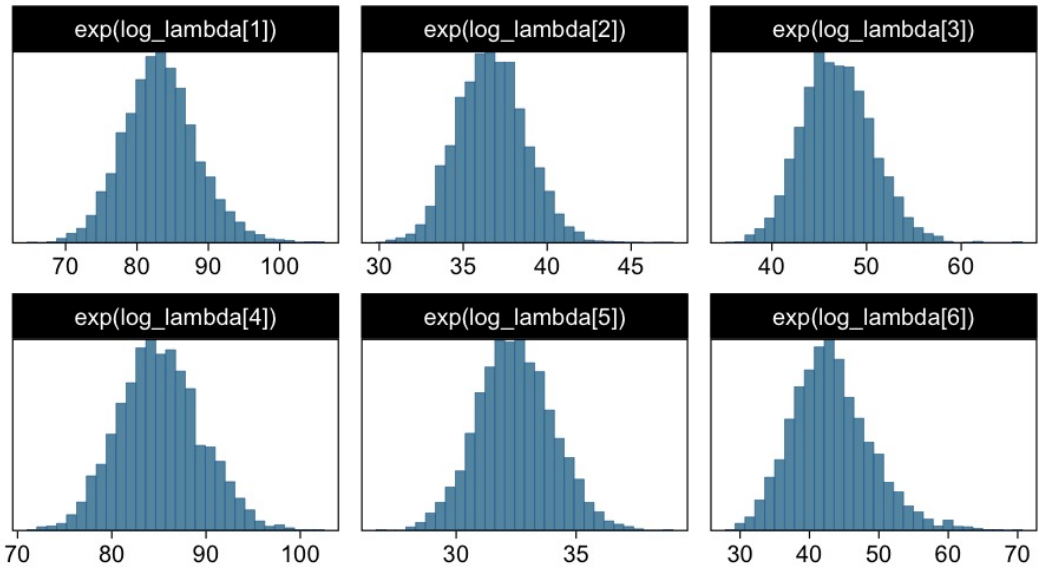


FIGURA 5. *Histogramas del parámetro de media.*

2.5. Conclusiones

Ajustamos un modelo con complejidad cada vez mayor. Identificando problemas (conceptuales y algorítmicas) en el desarrollo. No hemos discutido cómo comparar estos modelos pues podríamos estar cayendo en problemas de `sobre-ajuste` o de `sobre-parametrización`. ¿Cuántos parámetros tiene el último modelo?

3. EJEMPLO: TIROS DE GOLF

Este ejemplo lo hemos tomado de [1]. El objetivo de este **no** es volvernos expertos en modelar tiros de golf. El objetivo es **conocer de un proceso iterativo para construcción y validación de modelos**.

Queremos `entender` y `modelar` la **probabilidad de éxito de /putts** de Golf (*putts*: tiros relativamente cerca del hoyo que buscan que la pelota ruede al hoyo o muy cerca de él). Asi como entender la dependencia entre el éxito y la distancia del tiro. Como conclusiones quisiéramos inferir qué tan precisos son los profesionales en sus tiros [2].

`Definición (datos)`: El espacio de observaciones que esperaríamos son del tipo (x, y) donde x es la distancia del *putt* y y indica si se logró o no. Sin embargo, los datos que tenemos

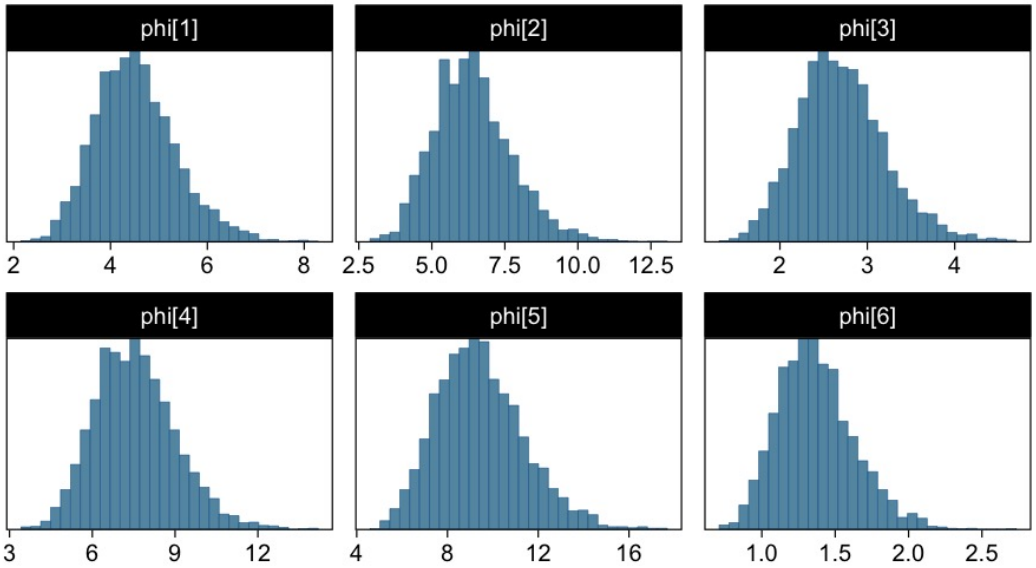


FIGURA 6. *Histogramas del parámetro de sobredispersión (precisión).*

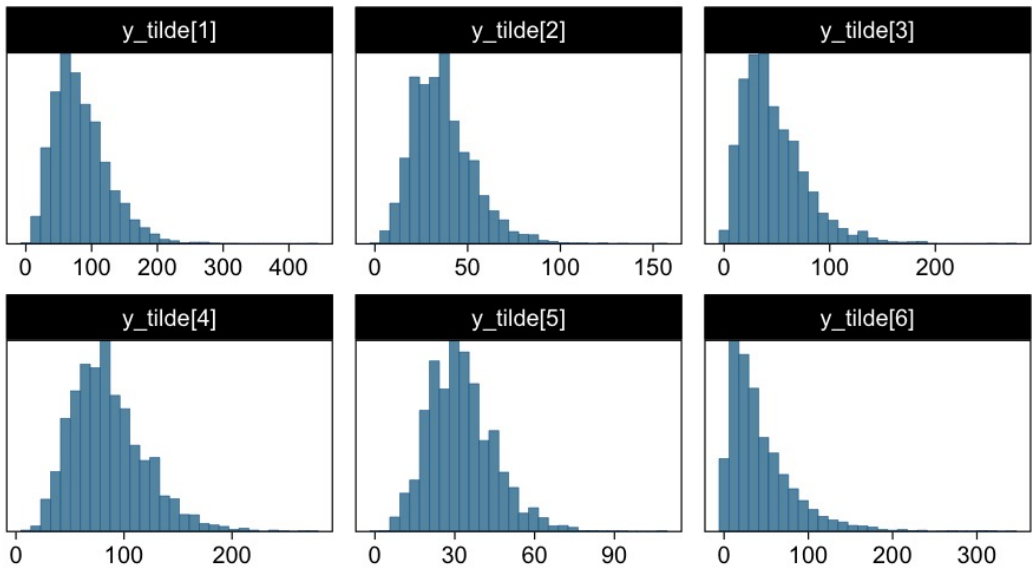


FIGURA 7. *Histogramas de cantidades observables.*

son agregados: para cada distancia aproximada x_j tendremos un conteo de intentos n_j y éxitos y_j sobre los tiros de los jugadores profesionales. En total las distancias han sido redondeadas y obtenemos $J = 19$ distancias distintas. En Fig. 8 se muestran los datos disponibles.

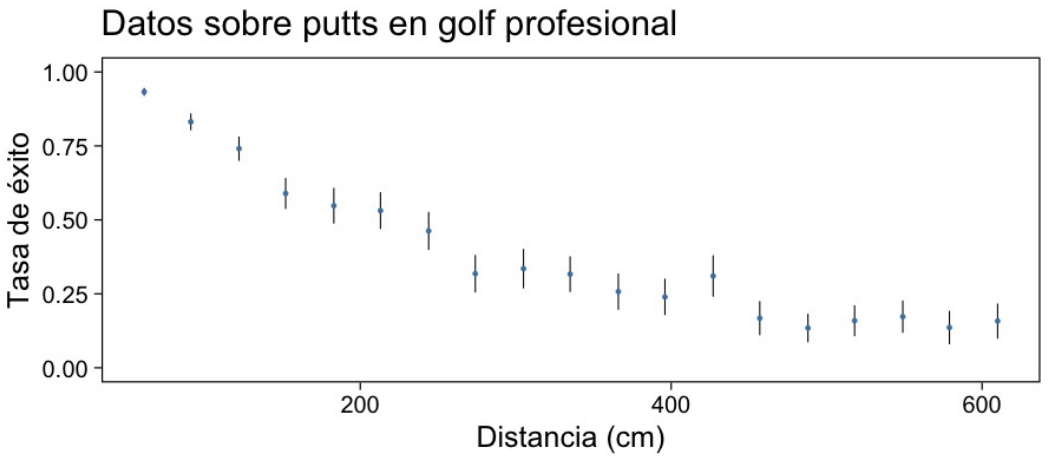


FIGURA 8. *Datos disponibles para análisis de éxitos de tiros.*

3.1. Modelo logístico

Un primer intento es modelar la probabilidad de éxito a través de una regresión logística.

$$p_j = \text{logit}^{-1}(a + bx_j)\,, \tag{7a}$$
$$y_j \sim \text{Binomial}\,(n_j, p_j)\,, \tag{7b}$$

para cada $j = 1, \ldots, J$. Esto es equivalente a

$$\log\left(\frac{p_j}{1-p_j}\right) = a + b\,x_j\,. \tag{8}$$

Este modelo lo escribimos en `Stan` como sigue

```
data {
    int J;
    int n[J];
    vector[J] x;
    int y[J];
}
parameters {
    real a;
    real b;
}
model {
    y ~ binomial_logit(n, a*x + b);
}
```

LISTING 3. *Modelo logístico para tasa de éxito de tiros de golf.*

Notemos que no hemos especificado una distribución inicial explícita para nuestros parámetros. Por default `Stan` está incorporando una distribución **plana** en todo el espacio $(a, b) \in \mathbb{R}^2$.

Podríamos debatir si esto es aceptable y las consecuencias de incluir una distribución inicial de esta naturaleza.

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

Utilicemos la siguiente función para evitar *overhead* en el ajuste del modelo.

```
ajustar_modelo ← function(modelo, datos, iter_sampling = 1000, iter_warmup =
    1000, seed = 2210){
  ajuste ← modelo$sample(data = datos,
                         seed = seed,
                         iter_sampling = iter_sampling,
                         iter_warmup = iter_sampling,
                         refresh = 0,
                         show_messages = FALSE)
  ajuste
}
```

```
data_list ← c(datos, list("J" = nrow(datos)))
ajuste ← ajustar_modelo(modelo, data_list)
```

A pesar de los problemas en la semillas iniciales parece ser que no hay problema en muestrear del modelo posterior.

```
# A tibble: 3 × 10...¹
  varia      mean    median       sd      mad        q5       q95  rhat ess_...²b
  <chr>       <dbl>     <dbl>    <dbl>    <dbl>     <dbl>     <dbl> <dbl>    <dbl>
1 lp__     -4.38e+5 -4.38e+5 9.58e-1 0        -4.38e+5 -4.38e+5  1.01     970.
2 a        -8.12e-3 -8.12e-3 1.47e-5 1.48e-5 -8.14e-3 -8.09e-3  1.01     850.
3 b         2.83e+0  2.83e+0 4.45e-3 4.43e-3  2.82e+0  2.83e+0  1.01     698.
# ... with 1 more variable: ess_tail <dbl>, and abbreviated variable names
# ¹  variable, ²ess_bulk
#  Use ‘colnames()‘ to see all variable names
```

Podemos explorar las trayectorias marginales en Fig. 9 . Todo indica que el ajuste está bien y no hay problemas aparentes con el modelo.

Fun fact: ¿cómo exploraron en la tarea podemos extraer los puntos que maximizan la distribución posterior?

```
params_map ← modelo$optimize(data = data_list, seed = 108)
```

```
params_map ← params_map$summary() ▷
  pivot_wider(values_from = estimate, names_from = variable)
params_map
```

```
# A tibble: 1 × 3
    lp__         a     b
   <dbl>     <dbl> <dbl>
1 -3020. -0.00838  2.23
```

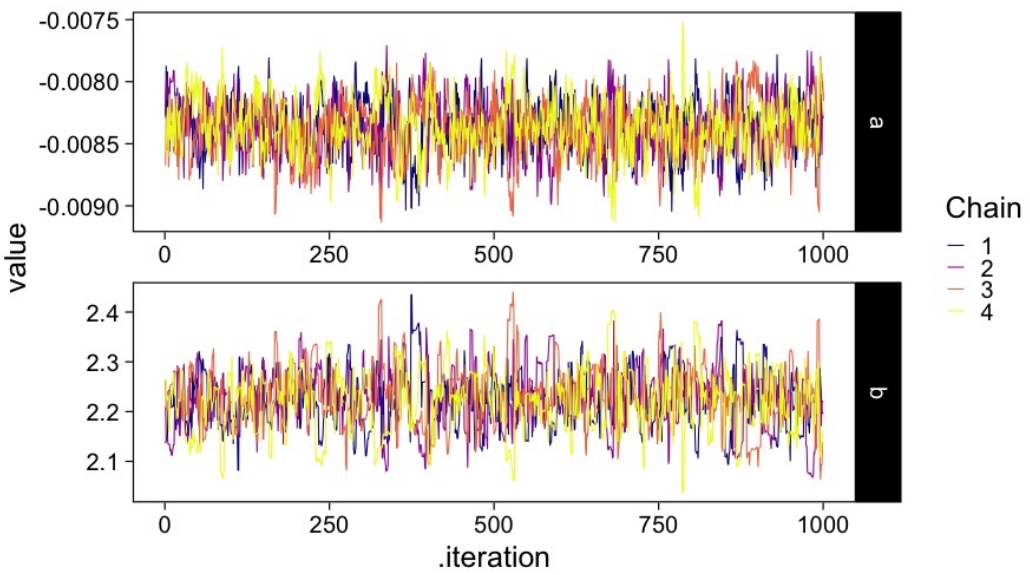


FIGURA 9. *Trayectorias de simulación.*

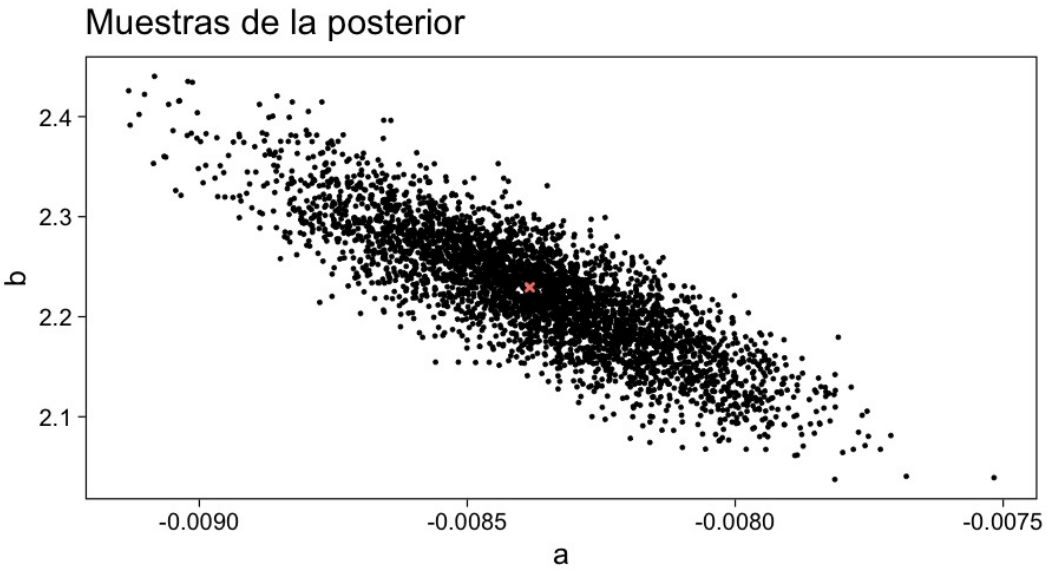


FIGURA 10. *Gráfico de dispersión.*

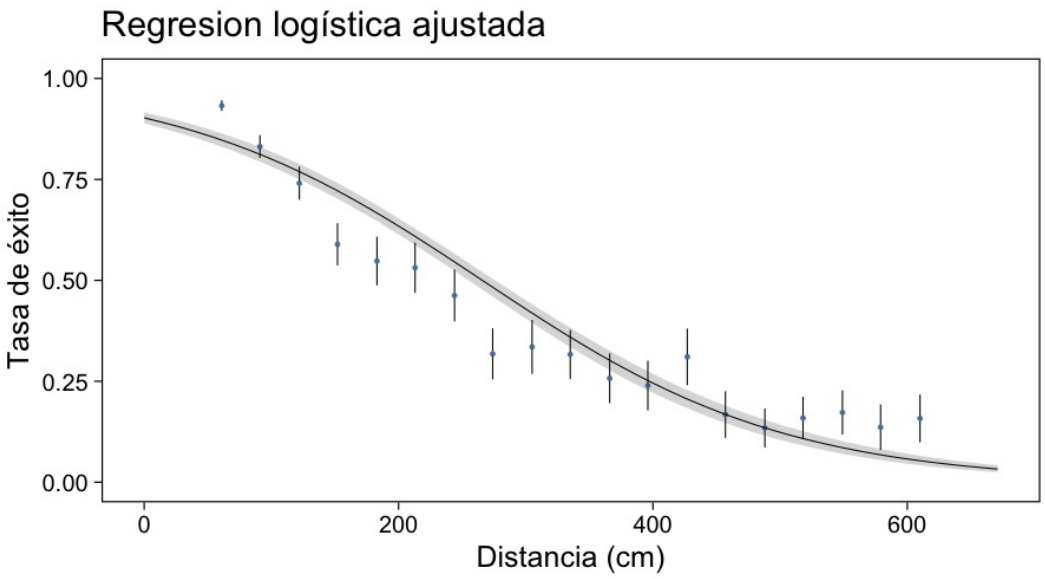


FIGURA 11. *Predictiva posterior del modelo logístico.*

Podríamos explorar un gráfico de dispersión para visualizar la correlación posterior de nuestros parámetros y ubicar el valor que maximiza la pseudo-posterior, Fig. 10.

En Fig. 11 la línea solida representa la mediana de la curva de regresión calculada entre las muestras de la posterior obtenidas. La región sombreada corresponde a la banda del 99 % de credibilidad calculada a partir del mismo conjunto de muestras.

El modelo es razonable, en el sentido de que los parámetros tienen los valores que esperaríamos. La pendiente del modelo de regresión logística es negativa, lo cual interpretamos como la falta de precisión del tirador mientras mas alejado del hoyo. Mientras que para el caso base ($x = 0$) el modelo da una probabilidad de éxito relativamente alta.

En las siguientes secciones ilustraremos el procedimiento para complementar el modelo.

3.2. Análisis conceptual

Podemos pensar en cada intento que hace un golfista como una prueba independiente que puede resultar en éxito o fracaso. El modelo anterior estable la probabilidad de éxito como una función no lineal de la distancia.

El problema es considerablemente complicado conceptualmente ([7]) si consideramos todas las fuentes de variación: ángulo de tiro, potencia de tiro, declive en *greens* y así sucesivamente.

Los supuestos que criticaremos son los siguientes. Seguiremos haciendo la simplificación de superficie plana, pero consideramos dos parámetros para el tiro con distintas condiciones de éxito:

1. El ángulo del tiro.
2. La velocidad con la que la pelota llega (o no llega) al hoyo.

Los radios de una pelota de golf y el hoyo (en centímetros) son de

```
# A tibble: 1 × 2
  pelota  hoyo
   <dbl> <dbl>
1    2.1   5.4
```

LISTING 4. *Radios para pelota y hoyo en una configuración de golf profesional.*

Supondremos por el momento que los *greens* de golf (áreas cerca del hoyo) son perfectamente planos (lo cual no es cierto, pero refinaremos después), de modo que el éxito depende de:

1. Tirar la pelota con un ángulo suficientemente cercano a cero con respecto a la línea que va del centro de la pelota al centro del hoyo.
2. Tirar la pelota con una velocidad suficiente para que llegue al hoyo pero no tan alta que vuele por encima del hoyo.

Mejores datos de los tipos de fallo sería útil, pero por el momento no los tenemos disponibles.

3.3. Angulo de tiro

Supongamos que la distancia del centro de la pelota al centro del hoyo es $x.$ Idealmente ésta es la trayectoria que el golfista tendría que ejecutar. Sin embargo, el tiro puede ser inexacto y denotamos por θ el ángulo del tiro realizado. El tiro es exitoso cuando el angulo de tiro satisface

$$|\theta| < \tan^{-1}\left(\frac{R-r}{x}\right). \tag{9}$$

Incorporamos un esquema de esta situación en Fig. 12.

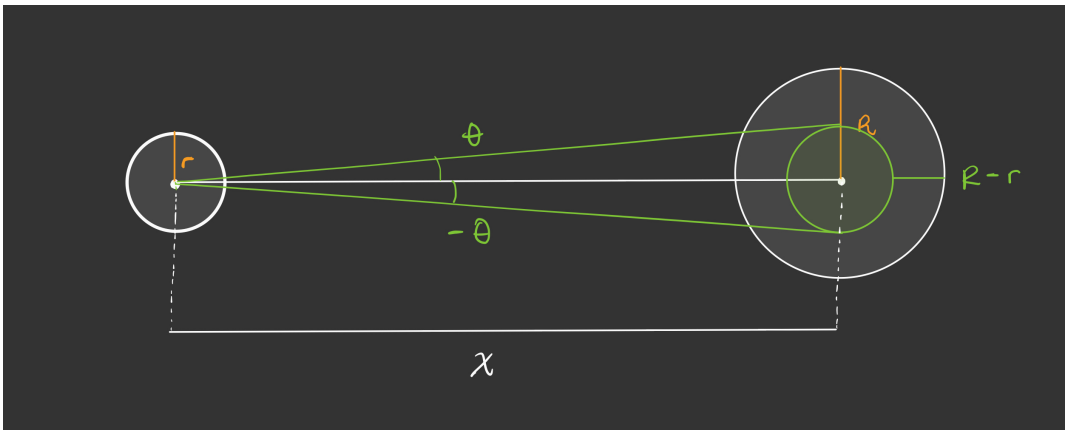


FIGURA 12. *Esquema de tiro y condiciones para un tiro exitoso.*

Observación: Aqui hemos hecho un supuesto importante. La `distancia reportada` en los datos, la cual hemos denotado por x, es la distancia entre el centro de la pelota y el centro del hoyo. ¿Cómo cambiaría nuestra condición de éxito si suponemos que la distancia que viaja la pelota es la registrada?

Para nuestro problema, la condición de éxito es

$$|\theta| < \tan^{-1}\left(\frac{3.3}{x}\right). \tag{10}$$

Mejores golfistas tendrán mejor control sobre θ, y conforme x es más grande, la probabilidad de tener éxito baja, ver Fig. 13

Observación. Esta curva puede variar dependiendo del jugador, pero vamos a modelar el conjunto de tiros de jugadores profesionales. Suponemos homogeneidad, misma que podríamos checar con datos desagregados por jugador. Estos datos podrían tener sobre-representación de tiradores malos (pues quizá hacen más tiros).

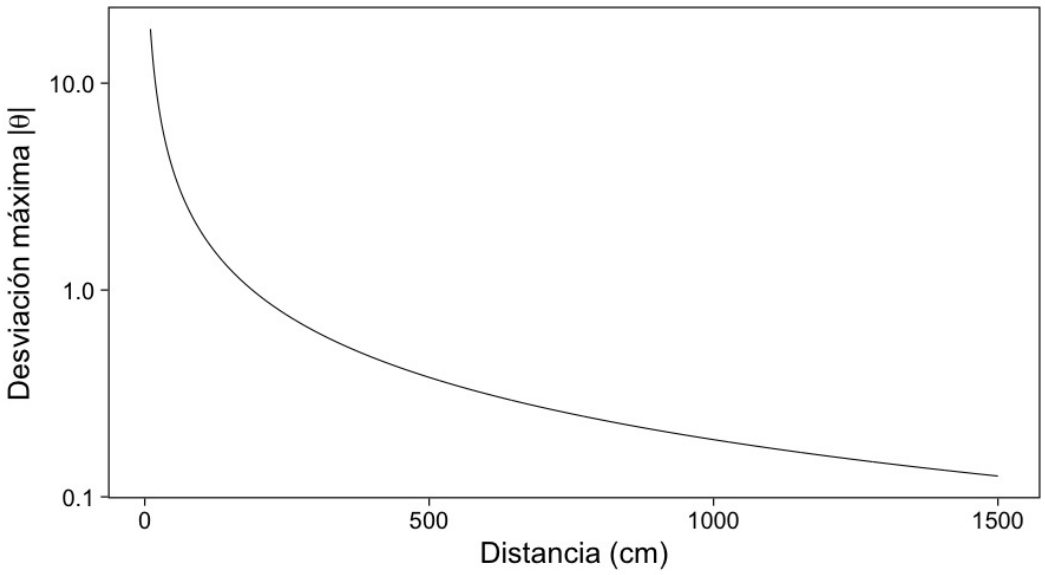


FIGURA 13. *Desviación máxima permitida para tener éxito a distintas distancias.*

Para modelar θ de manera probabilista asumimos una distribución Gaussiana con media 0 y desviación estándar σ. Este modelo codifica nuestra suposición de que los jugadores en promedio tirarán en la dirección correcta, sin embargo puede haber diversos factores que afectarán este resultado.

Siguiendo esta distribución, la probabilidad de éxito se calcula como

$$\mathbb{P}\left\{ |\theta| < \tan^{-1}\left(\frac{R-r}{x}\right)\right\} = 2\,\Phi\left[\frac{\tan^{-1}((R-r)/x)}{\sigma}\right] - 1\,, \tag{11}$$

donde Φ es la función de acumulación de una Normal estándar.

El parámetro σ controla la desviación de los tiros en línea recta. Por lo tanto afecta la probabilidad de éxito conforme mas lejos estemos y más grande sea su valor. Fig. 14 muestra que si el golfista tiene mejor control sobre su tiro, entonces mayor será su resistencia a encontrarse lejos.

Ahora veamos las distintas realizaciones de tiros a 1 metro de distancia bajo distintos valores de σ, Fig. 15. Nota que estamos *traduciendo* el impacto que tiene nuestro modelo previo en términos de observaciones tangibles del modelo.

Notamos que los tiros en general tienen un buen comportamiento. Posiblemente valores de tiros con una desviación de 60° dan lugar a tiros que no tienen sentido. Este punto lo veremos más adelante en caso de que tengamos que refinar. Por el momento, el modelo queda como sigue

$$p_j = 2\,\Phi\left(\frac{\tan^{-1}((R-r)/x_j)}{\sigma}\right) - 1\,, \tag{12}$$

$$y_j \sim \mathsf{Binomial}\,(n_j, p_j)\ , \tag{13}$$

para $j = 1, \ldots, J$.

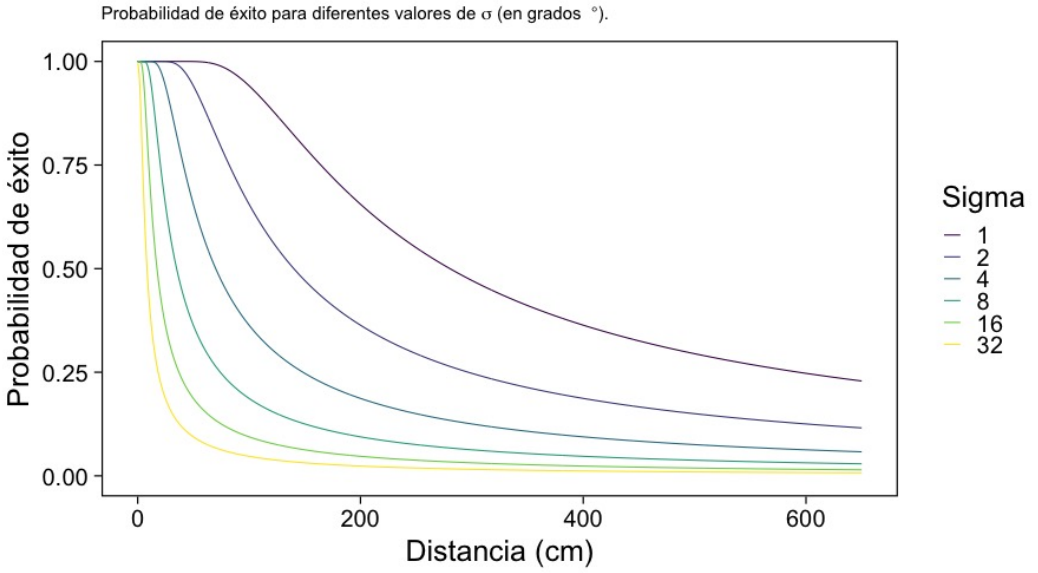


FIGURA 14. *Cómo cambia la probabilidad de éxito con la precisión del jugador.*

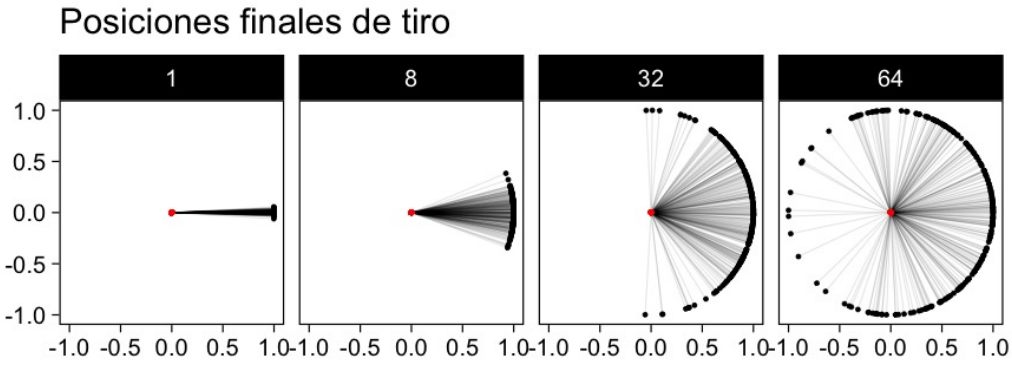


FIGURA 15. *Tiros aleatorios.*

La gran diferencia del modelo es asumir una relación distinta para la probabilidad de éxito de los experimentos binomiales. Este modelo se ha inferido de primeros principios y un poco de geometría.

3.4. Ajuste modelo

El modelo en `Stan` queda como se muestra. Nota que utilizamos la función de acumulación de una normal estándar Phi.

```
data {
    int J;
    int n[J];
    vector[J] x;
    int y[J];
    real r;
    real R;
}
transformed data {
    vector[J] threshold_angle = atan((R-r) ./ x);
}
parameters {
    real<lower=0> sigma;
}
model {
    vector[J] p = 2*Phi(threshold_angle / sigma) - 1;
    y ~ binomial(n, p);
}
generated quantities {
    real sigma_degrees = sigma * 180 / pi();
}
```

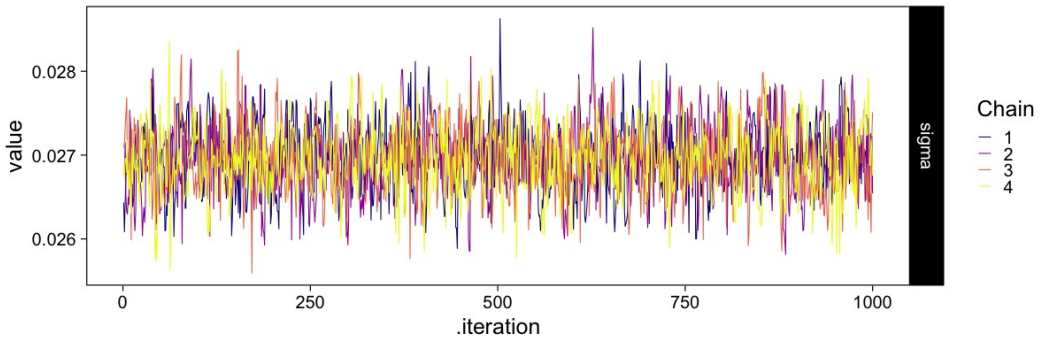
Listing 5. *Modelo con ángulo de tiro y su desviación estándar.*

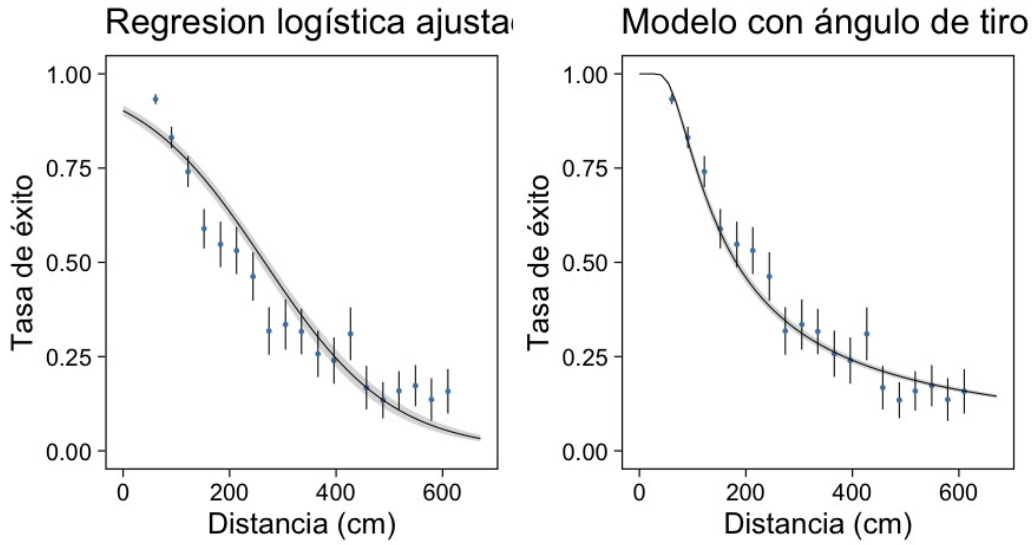
```
data_list$r = radios$pelota
data_list$R = radios$hoyo

ruta ← file.path("modelos/golf/modelo-angulo.stan")
modelo ← cmdstan_model(ruta, dir = modelos_files)

ajuste ← ajustar_modelo(modelo, data_list)
ajuste$summary() ▷ print(width = 75)
```

```
Model executable is up to date!
Running MCMC with 4 sequential chains...

Chain 1 finished in 0.1 seconds.
Chain 2 finished in 0.1 seconds.
Chain 3 finished in 0.1 seconds.
Chain 4 finished in 2.1 seconds.

All 4 chains finished successfully.
Mean chain execution time: 0.6 seconds.
Total execution time: 2.7 seconds.

Warning: 2224 of 4000 (56.0%) transitions ended with a divergence.
```

```
See https://mc-stan.org/misc/warnings for details.
# A tibble: 3 × 10...¹
  varia      mean    median       sd      mad        q5       q95  rhat ess_...²b
  <chr>       <dbl>     <dbl>   <dbl>   <dbl>     <dbl>     <dbl> <dbl>   <dbl>
1 lp__     -5.32e+5 -5.32e+5 1.06e+0 1.48e+0 -5.32e+5 -5.32e+5  1.01     473.
2 sigma     4.47e-2  4.47e-2 1.21e-7 1.48e-7  4.47e-2  4.47e-2  1.01     453.
3 sigma_...  2.56e+0  2.56e+0 6.51e-6 0         2.56e+0  2.56e+0  1.00     481.
# ... with 1 more variable: ess_tail <dbl>, and abbreviated variable names
# ¹  variable, ²ess_bulk
#  Use ‘colnames()‘ to see all variable names
```

El muestreo del modelo posterior parece no tener problemas. Los diagnósticos se ven bien y las capacidades predictivas dan indicios que se ha podido ajustar un modelo satisfactorio.





3.5. Nuevo conjunto de datos

Después de algunos años se consiguieron mas registros. En particular, el profesor Broadie fue el que brindo dichos datos (comunicación con Andrew Gelman documentada en [1]). La cantidad de datos disponibles es impresionante, basta con observar la dispersión de la probabilidad de éxito bajo el supuesto normal. Los intervalos de confianza son casi imperceptibles para las nuevas observaciones (puntos salmón en el gráfico).

Ajustando el modelo a los datos nuevos vemos que parece no haber un buen ajuste, Fig. 16. Subestimamos las tasa de éxito cuando estamos cerca y sobre-estimamos cuando nos encontramos muy lejos.

Esto sugiere regresar a analizar el modelo.

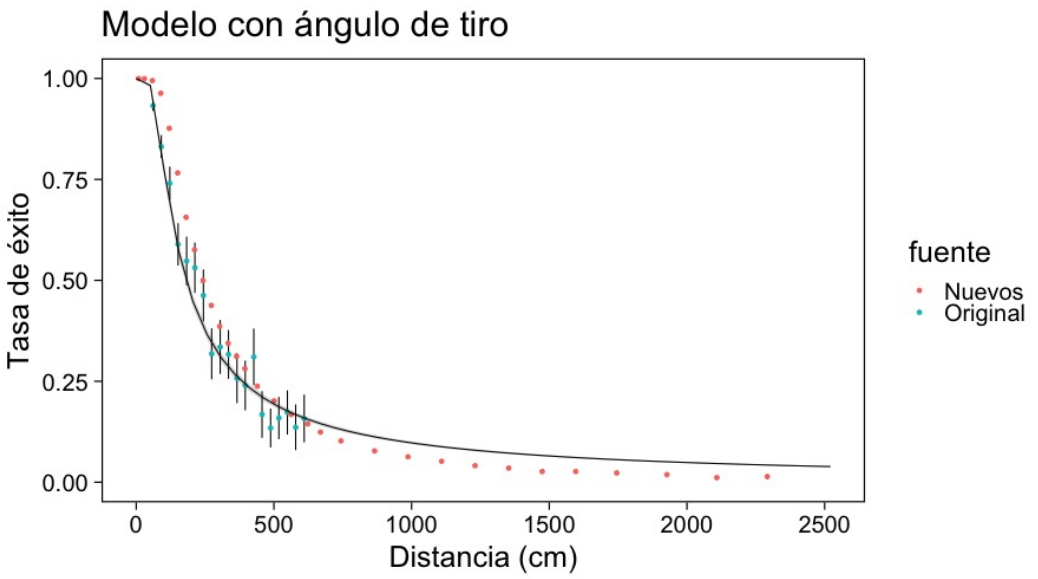


FIGURA 16. *Ajuste a nuevo conjunto de datos.*

4. MENSAJE

- Es fácil escribir modelos Bayesianos y hacer inferencia.
- Difícil mantener en producción: limitar el alcance del modelo.
- Reparametrización, previas informativas.
- El muestreo podría no escalar.

REFERENCIAS

[1] A. Gelman. Model building and expansion for golf putting. *On line*, 2019. URL https://mc-stan.org/users/documentation/case-studies/golf.html. 10, 20

[2] A. Gelman and D. Nolan. A Probability Model for Golf Putting. *Teaching Statistics*, 24(3):93–95, 2002. ISSN 1467-9639. . 10

[3] 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. 6

[4] A. Gelman, J. Hill, and A. Vehtari. *Regression and other stories*. Cambridge University Press, 2020. 1

[5] O. A. Martin, R. Kumar, and J. Lao. *Bayesian Modeling and Computation in Python*. Chapman and Hall/CRC, Boca Raton, First edition, 2021. 1, 2

[6] R. McElreath. *Statistical Rethinking: A Bayesian Course with Examples in R and Stan*. CRC Texts in Statistical Science. Taylor and Francis, CRC Press, Boca Raton, Second edition, 2020. ISBN 978-0-367-13991-9. 6

[7] A. R. Penner. The physics of putting. *Canadian Journal of Physics*, 80(2):83–96, feb 2002. ISSN 0008-4204, 1208-6045. . 15