EST-46115: Modelación Bayesiana

Profesor: Alfredo Garbuno Iñigo — Primavera, 2022 — Flujo de trabajo bayesiano.
Objetivo. Que veremos.
Lectura recomendada: Leer Capítulo 9 de [5] y el artículo [4].

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 computo 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.

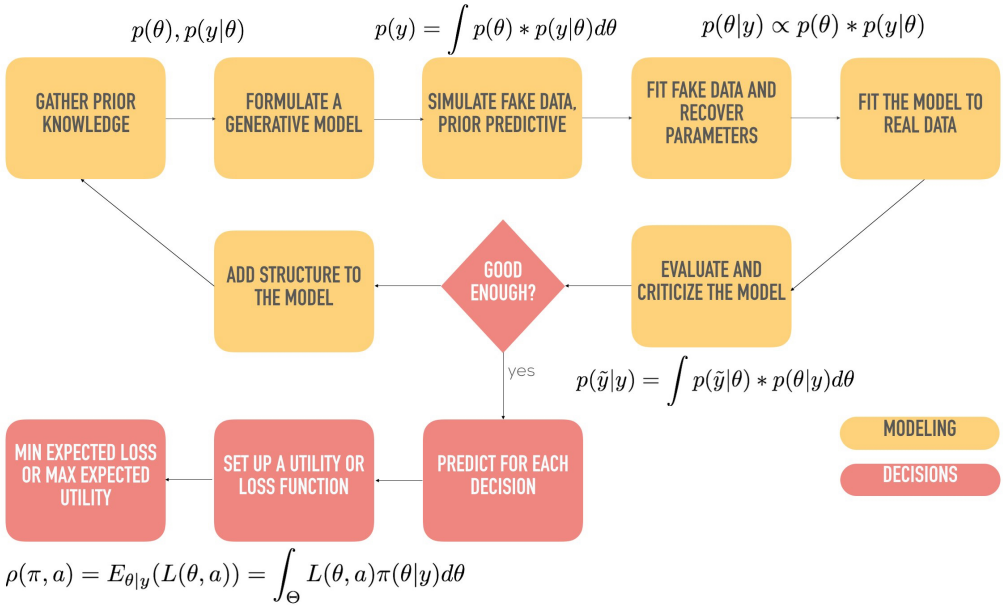


FIGURA 1. *Tomada de @BayesDose, Generable.*

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.

```
reclamos ▷ head(3) ▷ as.data.frame()
```

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

```
  promedio
1    54.88
```

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:

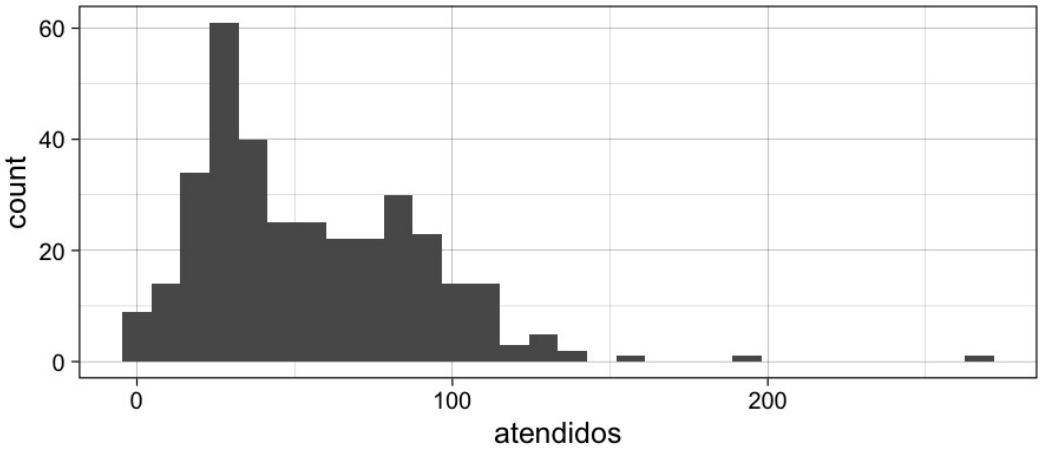


FIGURA 2. *Conteos de reclamos diarios atendidos por las aerolíneas.*

- `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 $X|\lambda \sim \text{Poisson}(\lambda)$ con $\lambda > 0$.

Dada nuestra ignorancia sobre el problema escogemos una distribución previa para λ como una $\text{Gamma}(100, 2)$ 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)
```

```
Running MCMC with 4 sequential chains...

Chain 1 finished in 0.0 seconds.
Chain 2 finished in 0.0 seconds.
Chain 3 finished in 0.0 seconds.
Chain 4 finished in 0.0 seconds.

All 4 chains finished successfully.
Mean chain execution time: 0.0 seconds.
Total execution time: 0.6 seconds.
Running MCMC with 4 sequential chains...

Chain 1 finished in 0.0 seconds.
Chain 2 finished in 0.0 seconds.
Chain 3 finished in 0.0 seconds.
Chain 4 finished in 0.0 seconds.

All 4 chains finished successfully.
Mean chain execution time: 0.0 seconds.
Total execution time: 0.6 seconds.
```

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

Podemos extraer resúmenes como sigue

```
  variable      mean    median      sd     mad       q5      q95  rhat ess_bulk
1     lp__ 57355.55 57355.80 0.7042 0.2965 57354.2 57356.0 1.002     1761
2   lambda    54.85    54.84 0.3947 0.3822    54.2    55.5 1.003     1334
  ess_tail
1     2176
2     1670
```

Y finalmente podemos crear histogramas para comparar la distribución previa contra la posterior sobre el parámetro de interés

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

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

2.1.1. Conclusiones: Lo que observamos 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 contorladas 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:

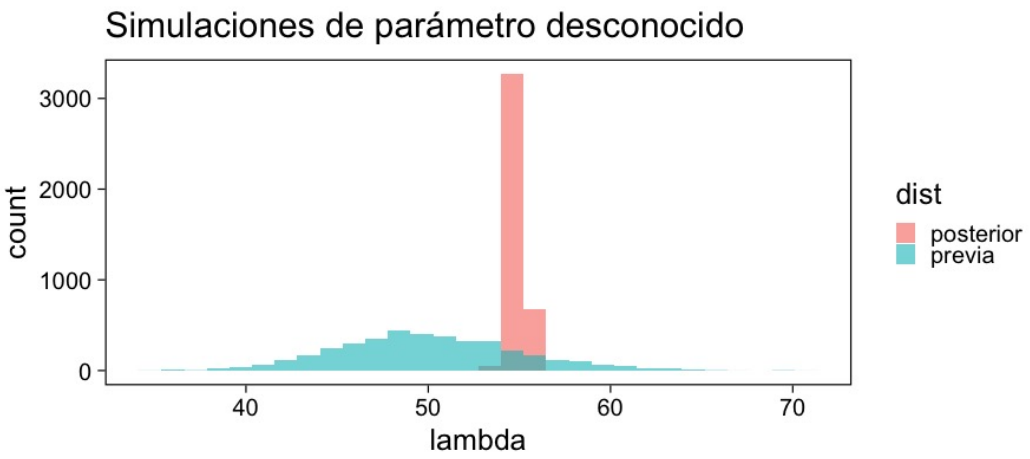


FIGURA 3. *Histograma de λ bajo la distribución previa (azul) y posterior (salmón).*

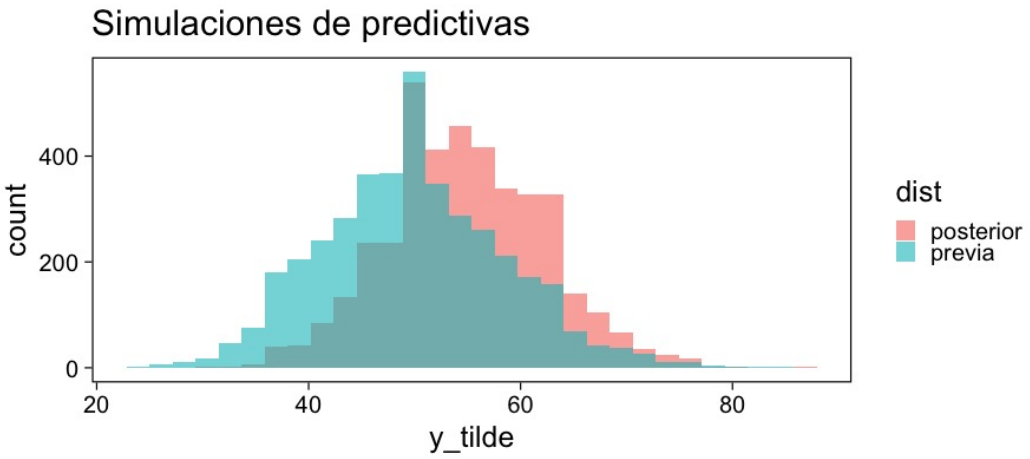


FIGURA 4. *Histogramas de observaciones hipotéticas del modelo bajo la distribución previa (azul) y posterior (salmón).*

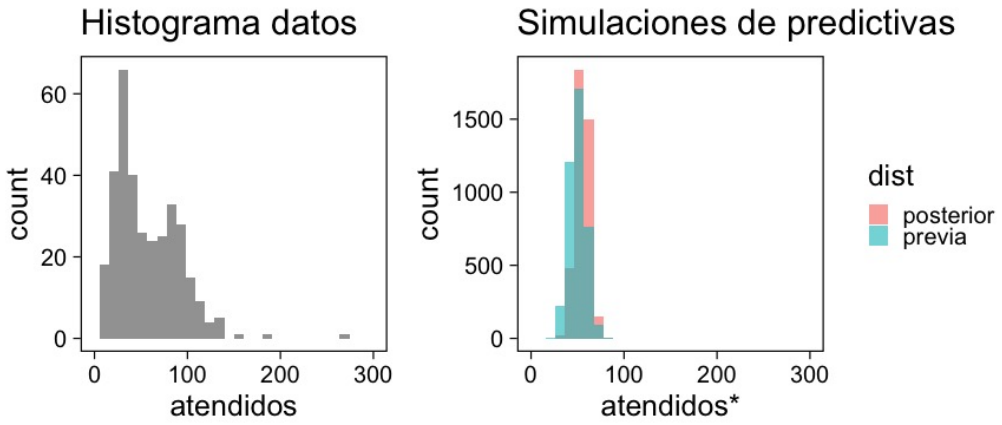


FIGURA 5. *Histogramas de observaciones hipotéticas del modelo bajo la distribución previa (azul) y posterior (salmón).*

```
  promedio varianza
1    54.88     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 $Y|\alpha, \beta \sim \text{NegBinom}(\alpha, \beta)$ 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 $Y|\mu, \phi \sim \text{NegBinom}(\mu, \phi)$ 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.

> Para entender una conexión adicional entre la `binomial negativa` y la `poisson` pensemos en que si marginalizamos una $\text{Poisson}(Y|\lambda)$ con respecto a una $\text{Gamma}(\lambda|\alpha, \beta)$ 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)
```

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.

```
  variable      mean    median       sd      mad         q5        q95   rhat
1     lp__ -1685.84 -1685.570   0.9435   0.7116 -1687.670 -1684.920 1.0001
2   lambda    54.88    54.818   1.9401   1.9842    51.821    58.050 0.9998
3      phi     2.23     2.228   0.1687   0.1719     1.958     2.511 1.0001
4  y_tilde    55.19    47.000 38.1400 32.6172    10.950   129.000 1.0000
  ess_bulk ess_tail
1     1988     2752
2     3504     2764
3     3577     2834
4     3938     3887
```

2.3. Reparametrizando

Posiblemente nos sintamos incómodos por las alertas asi que podemos buscar la parametrización 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
```

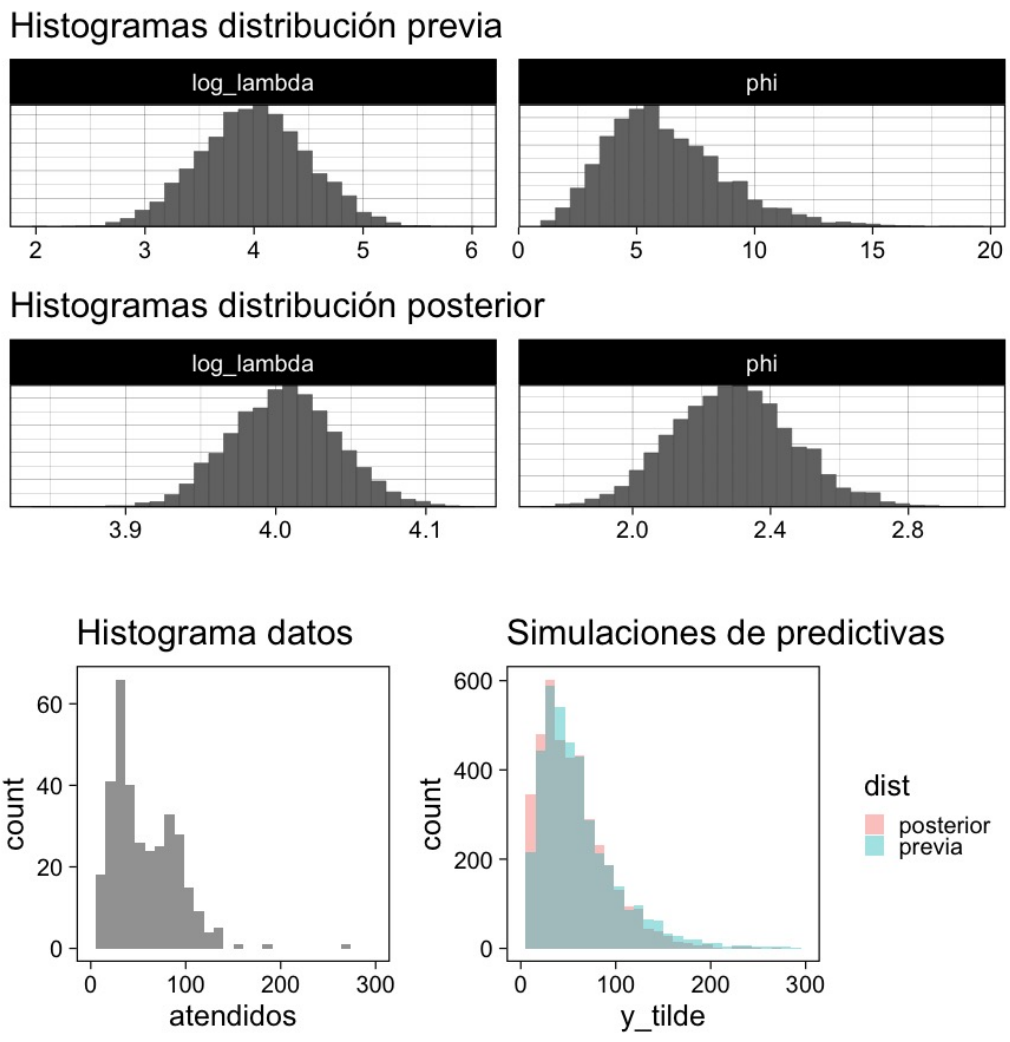
```
Running MCMC with 1 chain...

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'}
```

2.4. Definición del modelo

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

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

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

generated quantities {
  int y_tilde = neg_binomial_2_log_rng(log_lambda, phi);
}
```

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

```
    variable       mean     median        sd       mad         q5        q95   rhat
1         lp__ -1685.769 -1685.460   1.01656   0.71165 -1687.810 -1684.810 1.0009
2 log_lambda      4.006      4.006   0.03572   0.03608      3.948      4.065 1.0009
3        phi      2.292      2.290   0.17700   0.17579      2.010      2.585 1.0000
4    y_tilde     54.847     47.500 36.58278 33.35850     11.000    124.000 0.9999
  ess_bulk ess_tail
1     1695     2520
2     3671     2970
3     3933     2805
4     4243     3841
```



2.5. Modelo jerarquico

```
         compania promedio varianza  exceso precision
1       AlaskaAir    83.59   1341.1 0.17997     5.557
```

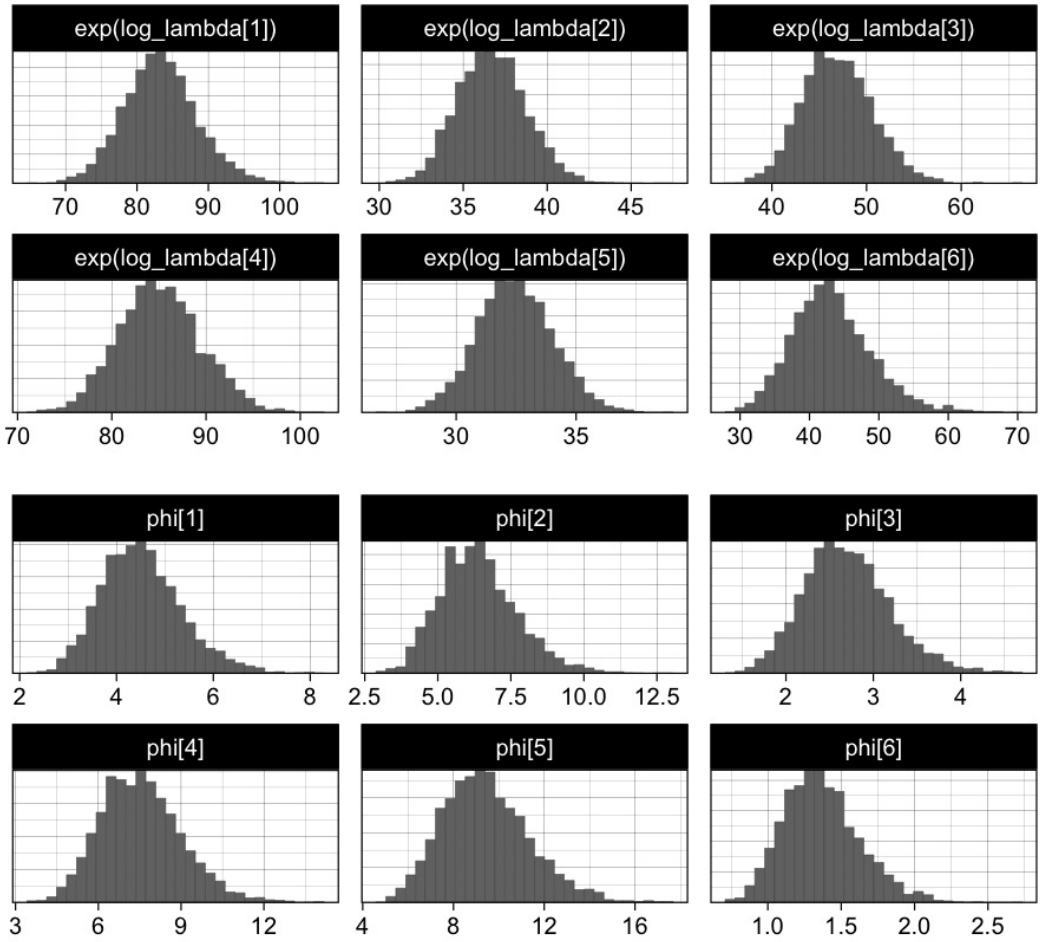
```
2  VirginAtlantic    36.44    185.9 0.11255     8.885
3 British_Airways    46.45    734.0 0.31867     3.138
4         JetBlue    85.20    637.3 0.07606    13.148
5   VirginAmerica    32.28    154.2 0.11698     8.548
6  AirAsiaSupport    41.61   1718.4 0.96831     1.033
```

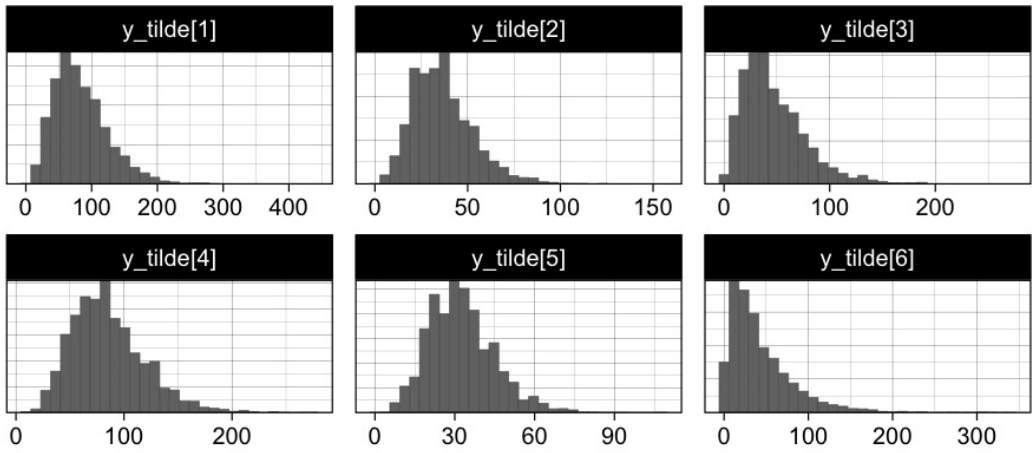
```
data {
  int N;
  int y[N];
  int compania[N];
  real<lower=0> gamma_alpha;
  real<lower=0> gamma_beta;
}

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)
```

```
        variable      mean    median        sd      mad          q5        q95
1           lp__ -1561.142 -1560.810  2.49926  2.40181 -1565.7410 -1557.720
2  log_lambda[1]     4.421     4.421  0.06188  0.05912     4.3196     4.525
3  log_lambda[2]     3.601     3.602  0.05583  0.05598     3.5115     3.692
4  log_lambda[3]     3.846     3.845  0.07883  0.07965     3.7189     3.975
5  log_lambda[4]     4.443     4.442  0.05118  0.05135     4.3578     4.528
6  log_lambda[5]     3.479     3.479  0.04844  0.04876     3.3978     3.558
7  log_lambda[6]     3.757     3.754  0.13021  0.12758     3.5439     3.975
8         phi[1]     4.519     4.461  0.83398  0.81118     3.2707     6.038
9         phi[2]     6.392     6.294  1.32577  1.27740     4.4290     8.763
10        phi[3]     2.702     2.664  0.49926  0.47912     1.9402     3.591
11        phi[4]     7.546     7.452  1.47012  1.45133     5.3441    10.094
12        phi[5]     9.437     9.290  1.91648  1.89374     6.5655    12.851
13        phi[6]     1.368     1.348  0.25863  0.25124     0.9832     1.831
14    y_tilde[1]    83.304    76.000 42.10947 38.54760    28.0000   162.000
15    y_tilde[2]    36.572    34.000 16.48838 14.82600    14.0000    66.000
16    y_tilde[3]    46.980    41.000 30.21969 26.68680    11.0000   104.000
17    y_tilde[4]    85.118    81.000 33.93328 32.61720    39.0000   147.000
18    y_tilde[5]    32.483    31.000 12.37788 11.86080    15.0000    55.000
19    y_tilde[6]    43.594    33.000 39.31701 29.65200     4.0000   119.000
```

```
      rhat ess_bulk ess_tail
1   1.0013     1499     2238
2   1.0010     8523     2855
3   1.0004     7864     3024
4   1.0006     6136     2849
5   1.0000     7305     3165
6   1.0012     7819     2843
7   0.9996     7997     2674
8   1.0011     7244     2829
9   1.0002     7781     2356
10  1.0016     7090     3164
11  0.9998     8263     2965
12  1.0031     9427     2987
13  1.0017     9598     3014
14  0.9997     3871     3714
15  0.9999     3861     3929
16  0.9996     3799     3721
17  0.9999     3763     3453
18  0.9995     4171     3791
19  1.0002     4232     3463
```



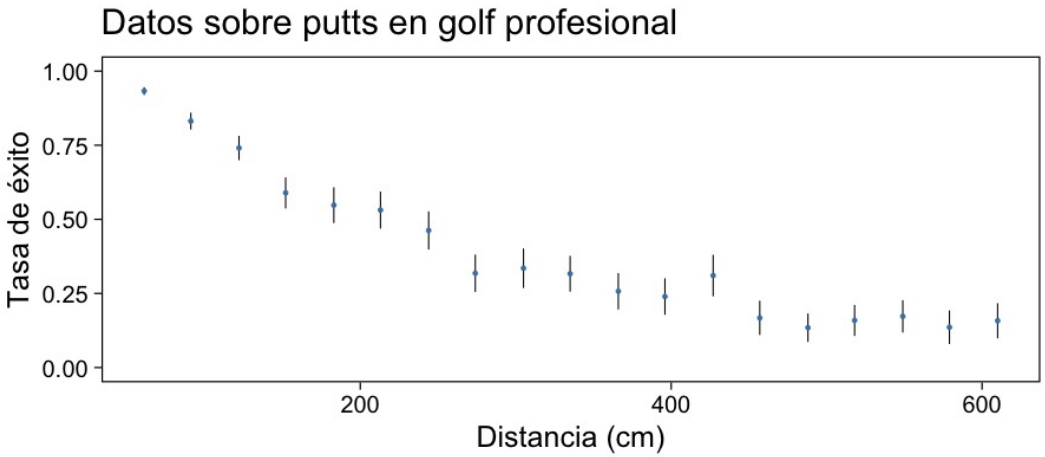


3. EJEMPLO: TIROS DE GOLF

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 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.



3.1. Modelo logístico

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

$$y_j \sim \text{Binomial}\left(n_j, \text{logit}^{-1}(a + bx_j)\right),$$

para cada $j = 1, \ldots, J$. 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 1. *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)
```

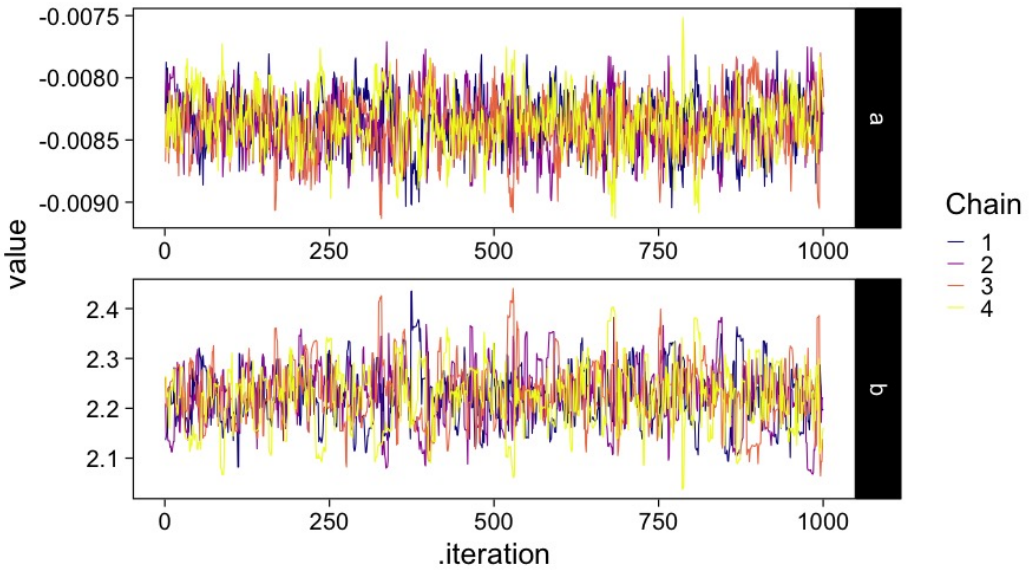
Utilicemos la siguiente función para evitar *overhead* en el ajuste del modelo.

```
Running MCMC with 4 sequential chains...

Chain 1 finished in 0.2 seconds.
Chain 2 Rejecting initial value:
Chain 2   Log probability evaluates to log(0), i.e. negative infinity.
Chain 2   Stan can't start sampling from this initial value.
Chain 2 finished in 0.2 seconds.
Chain 3 finished in 0.1 seconds.
Chain 4 Rejecting initial value:
Chain 4   Log probability evaluates to log(0), i.e. negative infinity.
Chain 4   Stan can't start sampling from this initial value.
Chain 4 finished in 0.1 seconds.

All 4 chains finished successfully.
Mean chain execution time: 0.2 seconds.
Total execution time: 1.0 seconds.
```

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

```
  variable       mean     median         sd        mad          q5         q95
1     lp__ -3.022e+03 -3.021e+03 1.0604122 0.7635390 -3.024e+03 -3.021e+03
2        a -8.372e-03 -8.365e-03 0.0002235 0.0002222 -8.750e-03 -8.008e-03
3        b  2.225e+00  2.225e+00 0.0602230 0.0601194  2.126e+00  2.325e+00
   rhat ess_bulk ess_tail
1 1.002    956.7   1213.5
2 1.001    814.1   1205.8
3 1.002    585.3    552.3
```

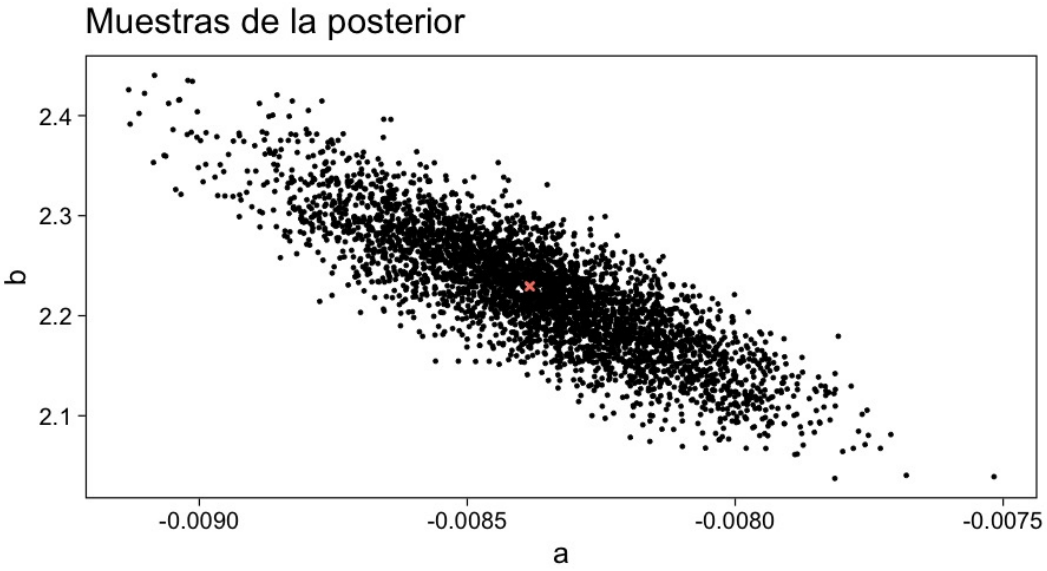
Podemos explorar las trayectorias marginales. Todo indica que el ajuste está bien y no hay problemas aparentes con el modelo.

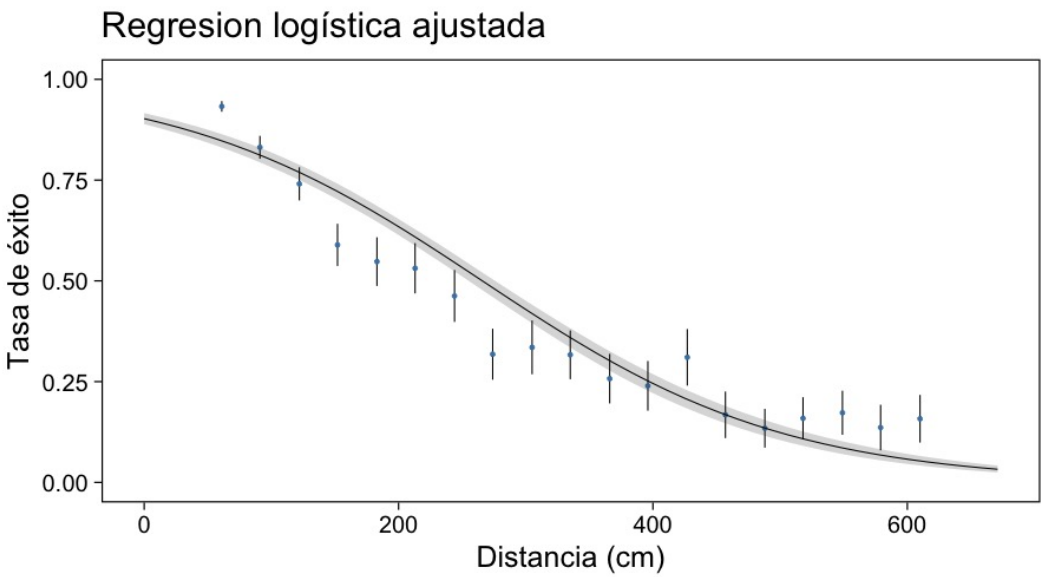


Fun fact: como exploraron en la tarea podemos extraer los puntos que maximizan la distribución posterior, ¿en serio?

```
params_map ← modelo$optimize(data = data_list, seed = 108)
params_map ← params_map$summary() ▷
  pivot_wider(values_from = estimate, names_from = variable)
params_map ▷ as.data.frame()
```

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





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

```
  pelota hoyo
1    2.1  5.4
```

Listing 2. *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.

1. 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

$$|\theta| < \tan^{-1}\left(\frac{R-r}{x}\right). \tag{7}$$

Incorporamos un esquema de esta situación a continuación.

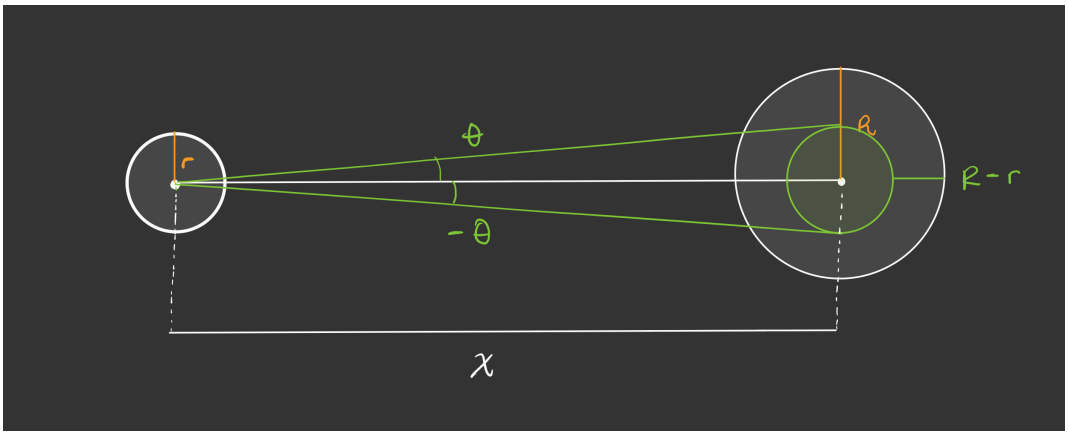


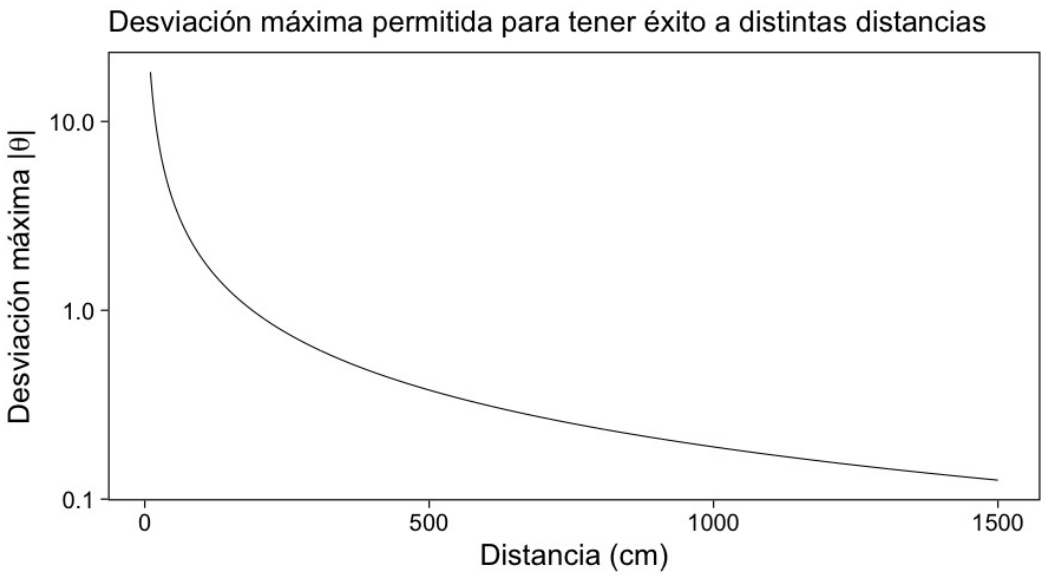
FIGURA 6. *Esquema de tiro y condiciones para un tiro exitoso.*

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

$$|\theta| < \tan^{-1}\left(\frac{3.3}{x}\right). \tag{8}$$

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



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).

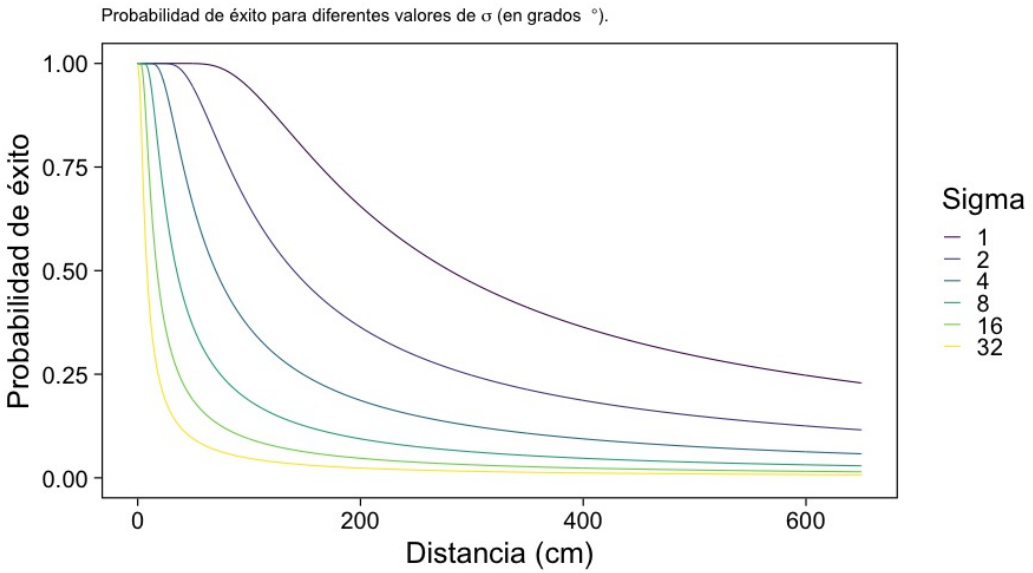
Para modelar θ de manera probabilista asumimos una distribución Gaussiana con media 0 y desviación estándar σ. Este modelo codifica nuestra suposición de que los jugadores en promedio tirarán en la dirección correcta, sin embargo puede haber diversos factores que afectarán este resultado.

Siguiendo esta distribución, la probabilidad de éxito se calcula como

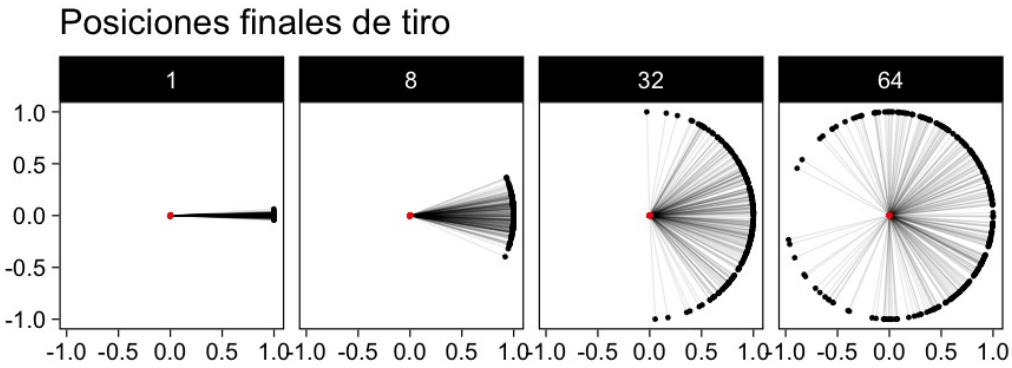
$$\mathbb{P}\left\{ |\theta| < \tan^{-1}\left(\frac{R-r}{x}\right)\right\} = 2\,\Phi\left[\frac{\tan^{-1}((R-r)/x)}{\sigma}\right] - 1\,, \tag{9}$$

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. El gráfico siguiente 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. 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

$$p_j = 2\,\Phi\left(\frac{\tan^{-1}((R-r)/x_j)}{\sigma}\right) - 1\,, \tag{10}$$

$$y_j \sim \text{Binomial}\,(n_j, p_j)\ , \tag{11}$$

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

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

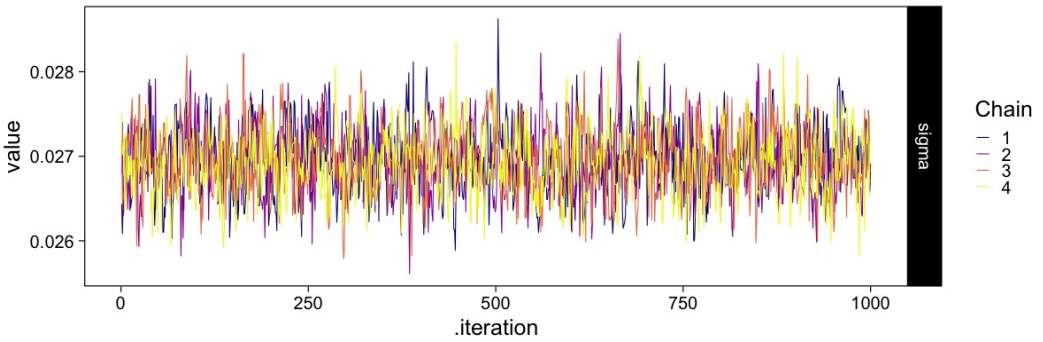
El modelo en `Stan` queda como se muestra. Nota que utilizamos la función de acumulación de una normal estándar Phi.

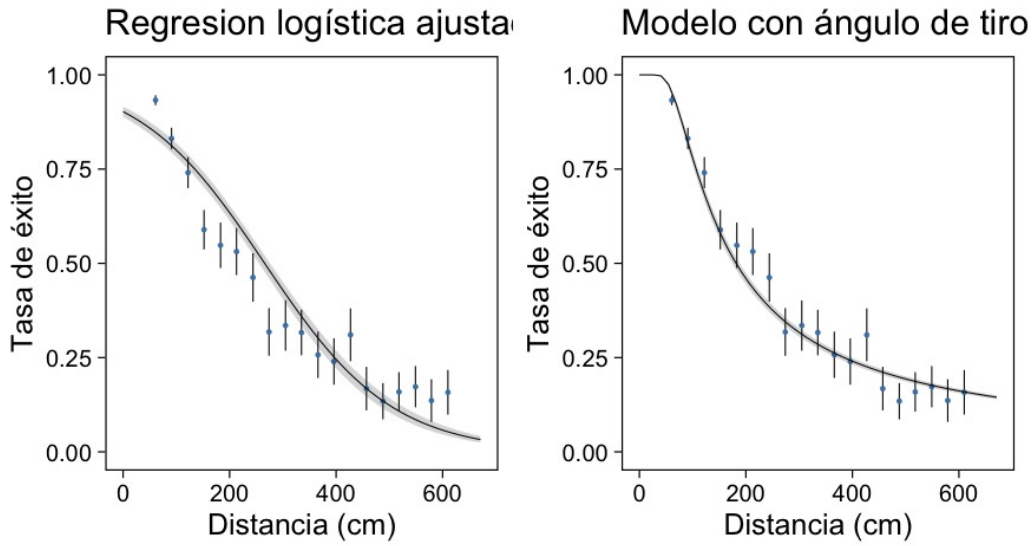
```
data {
    int J;
    int n[J];
    vector[J] x;
    int y[J];
    real r;
    real R;
}
transformed data {
    vector[J] threshold_angle = atan((R-r) ./ x);
}
parameters {
    real<lower=0> sigma;
}
model {
    vector[J] p = 2*Phi(threshold_angle / sigma) - 1;
    y ~ binomial(n, p);
}
generated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

Listing 3. *Modelo con ángulo de tiro y su desviación estándar.*

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

Chain 1 finished in 0.0 seconds.
Chain 2 finished in 0.0 seconds.
Chain 3 finished in 0.0 seconds.
Chain 4 finished in 0.0 seconds.

All 4 chains finished successfully.
Mean chain execution time: 0.0 seconds.
Total execution time: 0.6 seconds.
       variable       mean     median         sd       mad          q5
           q95
1          lp__ -2.926e+03 -2.926e+03 0.6751595 0.2965200 -2.928e+03 -2.926e
    +03
2         sigma  2.696e-02  2.695e-02 0.0003906 0.0003904  2.631e-02  2.761e
    -02
3 sigma_degrees  1.545e+00  1.544e+00 0.0223814 0.0223724  1.508e+00  1.582e
    +00
   rhat ess_bulk ess_tail
1 1.002     1994       NA
2 1.001     1530     2002
3 1.001     1530     2002
```

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.

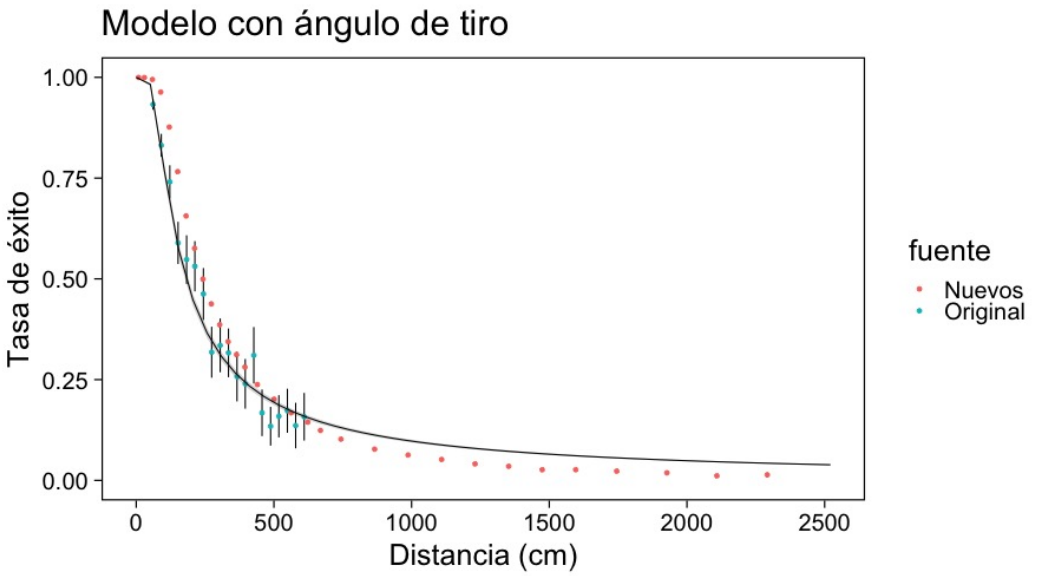




3.5. Nuevo conjunto de datos

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. Subestimamos las tasa de éxito cuando estamos cerca y sobre-estimamos cuando nos encontramos muy lejos.



3.6. Incorporando ángulo de tiro

Para poder hacer un tiro exitoso no sólo es necesario controlar el ángulo de tiro. También es importante tirar con la fuerza suficiente. Siguiendo [7], existe un rango de velocidades iniciales que determinan la condición de éxito.

La condición de éxito en un tiro recto es que la velocidad final v_f (en metros por segundo) de la pelota cumpla con las siguientes condiciones

$$0 < v_f < 1.63\,.$$

Por otro lado, la aceleración de la pelota al rodar en el *green* satisface

$$a = \left(\frac{10}{7}\right) \rho_r\, g\,.$$

donde $\rho_r = \rho/r$, y ρ depende de la superficie donde rueda la pelota, r es el radio de la pelota y g la fuerza de gravedad. Datos experimentales indican que la media en *greens* es de $\rho_r = 0.131$, con un rango de 0.065 a 0.196. De momento, tomaremos $\rho_r = 0.131$.

La velocidad final de la pelota, en términos de la velocidad inicial, utiliza la aceleración en el *green*, lo cual da la siguiente cadenca de igualdades

$$v_f^2 = v_0^2 - \left(\frac{10}{7}\right) \rho_r\, g\, x_m = v_0^2 - \left(\frac{10}{7}\right) (0.131)\,(9.81)\, x_m = v_0^2 - 1.835871\, x_m$$

donde x_m es la distancia de la pelota al hoyo en metros. Ahora, podemos despejar para calcular las condiciones de éxito sobre la velocidad inicial v_0

$$c\, x_m < v_0^2 < (1.63)^2 + c\, x_m\,,$$

donde $c = 1.835871$. La condición de éxito se puede escribir en términos de la distancia de la pelota al hoyo. Es decir podemos escribir

$$u \in [\, x,\, x + 145\,]\,,$$

donde $u = v_0^2/c \times 100$ es la distancia en centímetros que la pelota viajaría si no hubiera un hoyo en medio. Esto quiere decir que la pelota debe ser lanzada con fuerza suficiente para alcanzar el hoyo pero no tanta como para sobrepasarse.

Ahora, siguiendo las recomendaciones de Mark Broadie en [1]. Suponemos que los golfistas tienden a tirar con fuerza suficiente para pasarse del hoyo por un pie (30.48 cm), sin embargo la fuerza tiene un error multiplicativo. La intuición es que errores de la misma magnitud afectan en proporción a la distancia de tiro.

La distancia que recorre la pelota esta definida como

$$u = (x + 30.48) \cdot (1 + \varepsilon)\,,$$

donde

$$\varepsilon \sim \mathsf{N}(0, \sigma_f^2)\,,$$

y hemos utilizado la notación σ_f^2 para hace énfasis en el error asociado a la fuerza de tiro. Esto implica que

$$u \sim \mathsf{N}\left(x + 30.48, (x + 30.48)^2\sigma_f^2\right)\,,$$

y por la tanto el `éxito debido a la fuerza de tiro` —la condición $u \in [\,x,\, x + 145\,]$ — tiene probabilidad de éxito igual a

$$\Phi\left(\frac{114.52}{(x + 30.48)\sigma_f}\right) - \Phi\left(\frac{-30.48}{(x + 30.48)\sigma_f}\right)\,,$$

que es un evento que asumimos `independiente del ángulo de tiro`.

Para finalizar, utilizamos las condiciones de éxito que definen ambos eventos que asumimos independientes, el ángulo de tiro y la fuerza. Por lo tanto, el modelo lo escribimos como

$$p_j^u = \Phi\left(\frac{114.52}{(x + 30.48)\sigma_f}\right) - \Phi\left(\frac{-30.48}{(x + 30.48)\sigma_f}\right), \tag{12a}$$

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

$$p_j = p_j^u \cdot p_j^\theta \tag{12c}$$

$$y_j \sim \mathsf{Binomial}\,(n_j, p_j)\,, \tag{12d}$$

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

> Nota cómo el cambio que tenemos en nuestro modelo es la composición de dos eventos que esperamos sean independientes: la fuerza y dirección de tiro.

```
data {
    int J;
    int n[J];
    vector[J] x;
    int y[J];
    real r;
    real R;
    real overshot;
    real distance_tolerance;
}
transformed data {
    vector[J] threshold_angle = atan((R-r) ./ x);
    vector[J] raw_proportion  = to_vector(y) ./ to_vector(n);
}
parameters {
    real<lower=0> sigma_angle;
    real<lower=0> sigma_force;
```

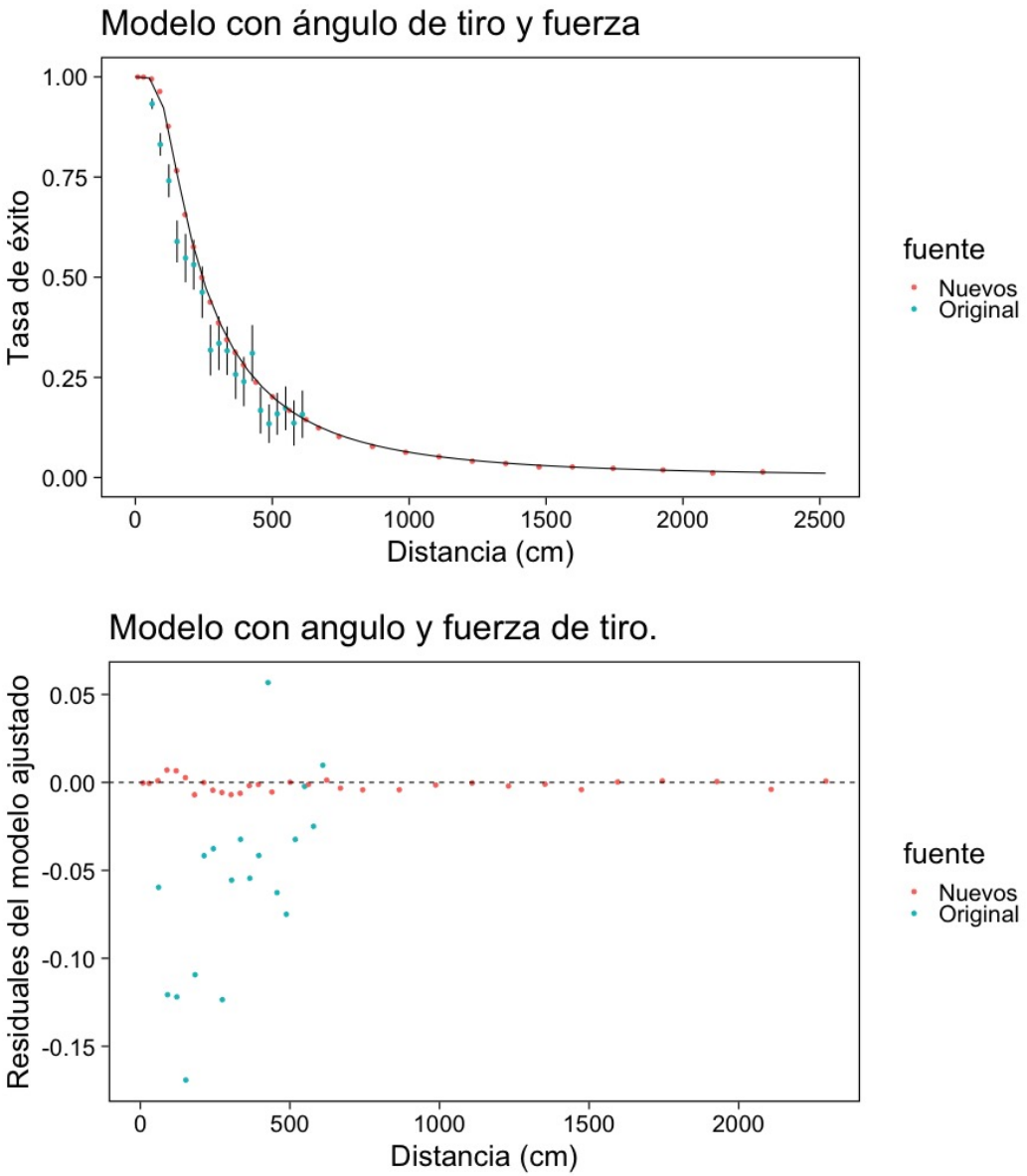
```
}
transformed parameters {
    vector[J] p_angle = 2*Phi(threshold_angle / sigma_angle) - 1;
    vector[J] p_force = Phi((distance_tolerance - overshot) ./ ((x + overshot)
       *sigma_force)) -
                Phi((- overshot) ./ ((x + overshot)*sigma_force));
    vector[J] p = p_angle .* p_force;
}
model {
    y ~ binomial(n, p);
}
generated quantities {
    real sigma_degrees = sigma_angle * 180 / pi();
    vector[J] residual = raw_proportion - p;
}
```

Listing 4. *Modelo con fuerza y ángulo de tiro.*

```
data_new ← list(x = datos$x, n = datos$n, y = datos$y, J = nrow(datos),
                 r = radios$pelota, R = radios$hoyo,
                 distance_tolerance = 4.5 * 30.48,# 145,
                 overshot = 30.48)
```

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

Chain 1 finished in 0.2 seconds.
Chain 2 finished in 0.2 seconds.
Chain 3 finished in 0.2 seconds.
Chain 4 finished in 0.2 seconds.

All 4 chains finished successfully.
Mean chain execution time: 0.2 seconds.
Total execution time: 1.3 seconds.
       variable   mean median        sd       mad      q5     q95  rhat
1   sigma_angle 0.0150 0.0150 4.275e-05 4.225e-05 0.01492 0.01506 1.001
2 sigma_degrees 0.8592 0.8593 2.449e-03 2.420e-03 0.85491 0.86305 1.001
3   sigma_force 0.1362 0.1362 4.882e-04 4.878e-04 0.13548 0.13711 1.002
  ess_bulk ess_tail
1     1321     1536
2     1321     1536
3     1183     1261
```

Si utilizamos la semilla 2210 (al menos en mi máquina) veríamos que el ajuste del modelo parece indicar ciertos problemas. En particular notemos que podrían ser causados por un punto inicial en una cadena. Después de todo, con 4 cadenas tenemos 25 % del esfuerzo computacional en una sola. Además, tenemos alertas en los demás diagnósticos. Con tales resultados nos mostramos un poco escépticos sobre los siguientes resúmenes gráficos.



Al explorar los residuales encontramos que parece haber cierto patrón. Mas aún, el modelo parece estar **muy** seguro de los valores esperados de probabilidad de éxito —lo cual podemos apreciar al incorporar los intervalos de probabilidad de los residuales que se calculan de las muestras. Esto se puede deber a que el número elevado de registros que la nueva base de datos provee.

Alternativamente, podríamos ajustar sólo en los datos nuevos. Pero no tenemos alguna justificación específica para descartar los que ya teníamos. Por lo pronto usaremos ambos conjuntos sin distinción.

3.7. Otro modelo

Una estrategia es incorporar una `aproximación continua` a las proporciones reportadas, misma que podemos utilizar para incorporar un `error de medición latente` (que en este

caso podría ser acertado). El modelo queda especificado como

$$p_j^u = \Phi\left(\frac{114.52}{(x+30.48)\sigma_f}\right) - \Phi\left(\frac{-30.48}{(x+30.48)\sigma_f}\right), \tag{13a}$$

$$p_j^\theta = 2\,\Phi\left(\frac{\tan^{-1}((R-r)/x_j)}{\sigma_\theta}\right) - 1, \tag{13b}$$

$$p_j = p_j^u \cdot p_j^\theta \tag{13c}$$

$$\frac{y_j}{n_j} \sim \mathsf{N}\left(p_j, \frac{p_j(1-p_j)}{n_j} + \sigma^2_{\mathsf{obs}}\right), \tag{13d}$$

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

Por otro lado, el modelo en `Stan` no cambia mucho y se vuelve un poco mas flexible. Lo cual especificamos en el bloque de modelo.

```
data {
    int J;
    int n[J];
    vector[J] x;
    int y[J];
    real r;
    real R;
    real overshot;
    real distance_tolerance;
}
transformed data {
    vector[J] threshold_angle = atan((R-r) ./ x);
    vector[J] raw_proportion  = to_vector(y) ./ to_vector(n);
}
parameters {
    real<lower=0> sigma_angle;
    real<lower=0> sigma_force;
    real<lower=0> sigma_obs;
}
transformed parameters {
    vector[J] p_angle = 2*Phi(threshold_angle / sigma_angle) - 1;
    vector[J] p_force = Phi((distance_tolerance - overshot) ./ ((x + overshot)
        *sigma_force)) -
                Phi((- overshot) ./ ((x + overshot)*sigma_force));
    vector[J] p = p_angle .* p_force;
}
model {
    raw_proportion ~ normal(p, sqrt(p .* (1-p) ./ to_vector(n) + sigma_obs^2))
        ;
}
generated quantities {
    real sigma_degrees = sigma_angle * 180 / pi();
    vector[J] residual = raw_proportion - p;
}
```

Listing 5. *Modelo con error de medición.*

Podríamos ajustar como lo hemos hecho antes, pero en este caso si tenemos problemas serios en el ajuste.

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

```
Chain 1 finished in 0.9 seconds.
Chain 2 finished in 0.8 seconds.
Chain 3 finished in 0.7 seconds.
Chain 4 finished in 0.6 seconds.

All 4 chains finished successfully.
Mean chain execution time: 0.7 seconds.
Total execution time: 3.3 seconds.

Warning: 1891 of 4000 (47.0%) 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
     variable       mean     median     sd        mad      q5        q95  rhat
1 sigma_angle 4.261e+307 3.169e+305    Inf 4.698e+305 0.01256 1.585e+308 2.238
2   sigma_obs  2.553e-01  2.049e-01 0.2264  2.696e-01 0.02627  5.450e-01 1.760
3 sigma_force 4.291e+307 4.381e+305    Inf 6.495e+305 0.07574 1.619e+308 2.153
  ess_bulk ess_tail
1    6.014    171.6
2    6.171    136.5
3    6.112     97.3
```

Podemos incorporar información **débil** en los parametros de escala, esto es por medio de normales truncadas en la región positiva. El modelo completo sería

$$\sigma^2 \sim \mathsf{N}^+(0,1) \tag{14a}$$

$$p_j^u = \Phi\left(\frac{114.52}{(x+30.48)\sigma_f}\right) - \Phi\left(\frac{-30.48}{(x+30.48)\sigma_f}\right), \tag{14b}$$

$$p_j^\theta = 2\,\Phi\left(\frac{\tan^{-1}((R-r)/x_j)}{\sigma_\theta}\right) - 1, \tag{14c}$$

$$p_j = p_j^u \cdot p_j^\theta, \tag{14d}$$

$$\frac{y_j}{n_j} \sim \mathsf{N}\left(p_j, \frac{p_j(1-p_j)}{n_j} + \sigma^2_{\mathsf{obs}}\right), \tag{14e}$$

para $j = 1, \ldots, J$, donde $\sigma^2 = (\sigma^2_{\mathsf{obs}}, \sigma^2_\theta, \sigma^2_f)$.

```
data {
    int J;
    int n[J];
    vector[J] x;
    int y[J];
    real r;
    real R;
    real overshot;
    real distance_tolerance;
}
transformed data {
    vector[J] threshold_angle = atan((R-r) ./ x);
    vector[J] raw_proportion  = to_vector(y) ./ to_vector(n);
}
parameters {
    real<lower=0> sigma_angle;
    real<lower=0> sigma_force;
```

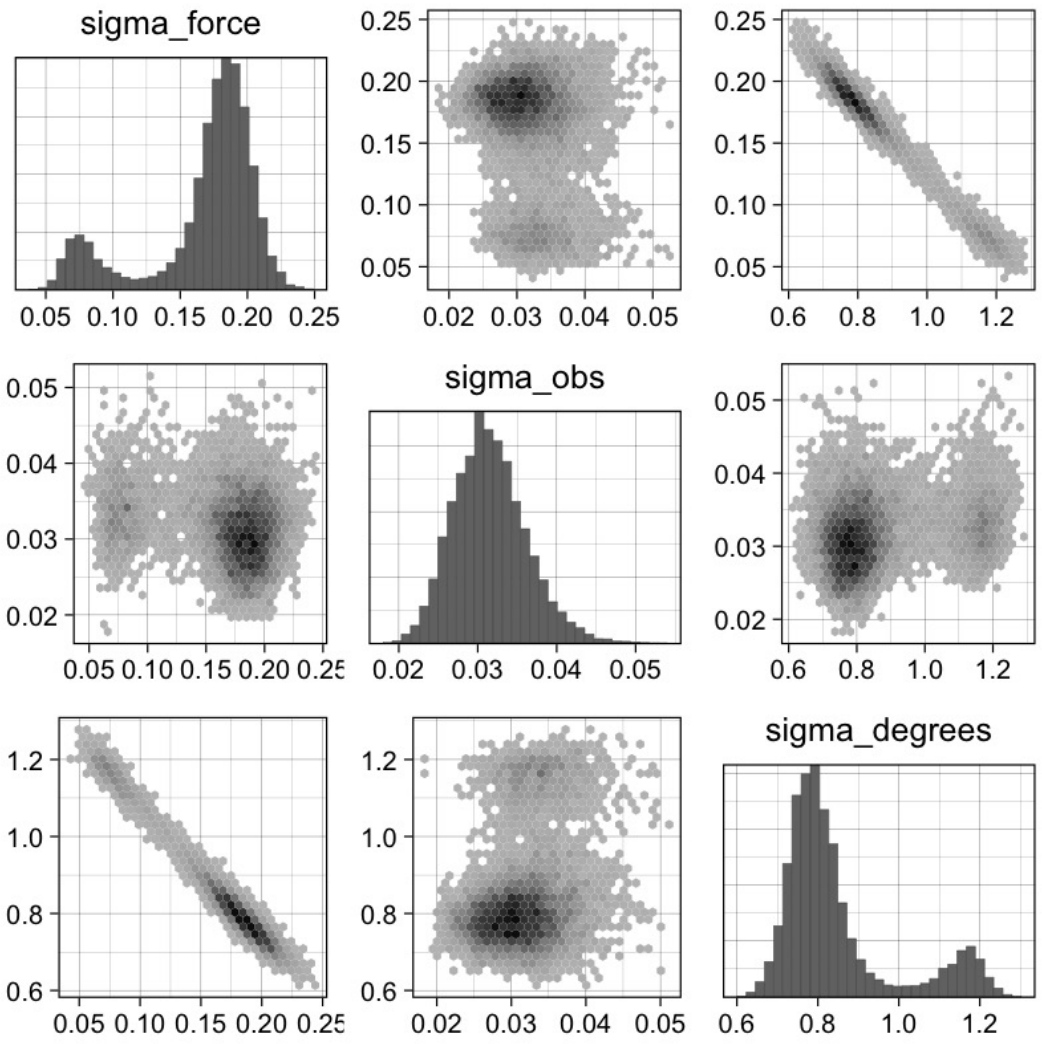
```
    real<lower=0> sigma_obs;
}
transformed parameters {
    vector[J] p_angle = 2*Phi(threshold_angle / sigma_angle) - 1;
    vector[J] p_force = Phi((distance_tolerance - overshot) ./ ((x + overshot)
        *sigma_force)) -
                Phi((- overshot) ./ ((x + overshot)*sigma_force));
    vector[J] p = p_angle .* p_force;
}
model {
    raw_proportion ~ normal(p, sqrt(p .* (1-p) ./ to_vector(n) + sigma_obs^2))
        ;
    [sigma_angle, sigma_force, sigma_obs] ~ normal(0, 1);
}
generated quantities {
    real sigma_degrees = sigma_angle * 180 / pi();
    vector[J] residual = raw_proportion - p;
}
```

Listing 6. *Modelo completo con información débil.*

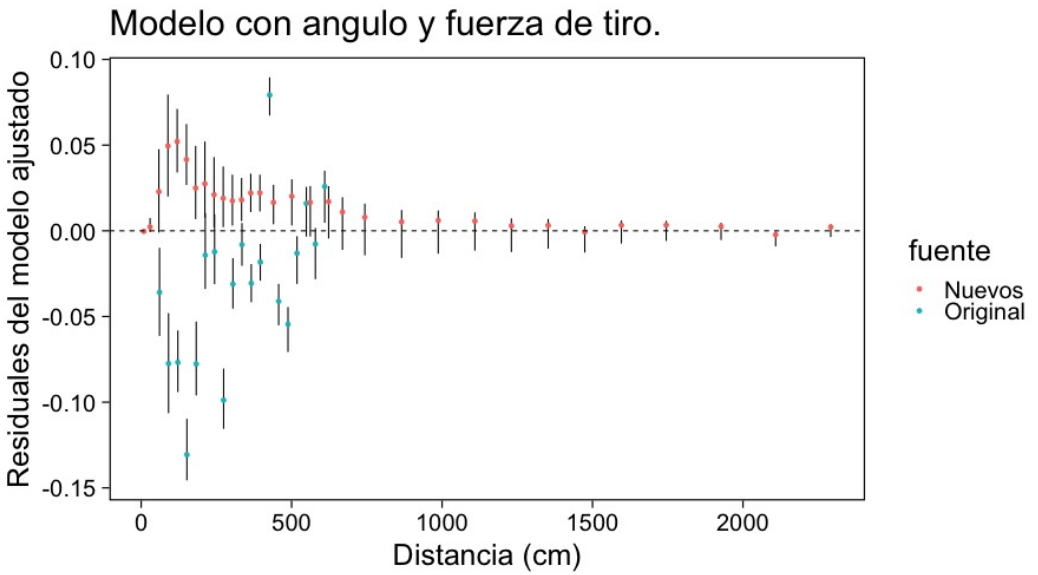
```
Model executable is up to date!
Running MCMC with 4 sequential chains...

Chain 1 finished in 2.7 seconds.
Chain 2 finished in 2.4 seconds.
Chain 3 finished in 2.3 seconds.
Chain 4 finished in 2.3 seconds.

All 4 chains finished successfully.
Mean chain execution time: 2.4 seconds.
Total execution time: 10.0 seconds.
       variable    mean  median       sd      mad      q5     q95  rhat
1   sigma_angle 0.01481 0.01400 0.002499 0.001330 0.01229 0.02059 1.003
2 sigma_degrees 0.84874 0.80225 0.143164 0.076183 0.70402 1.17956 1.003
3   sigma_force 0.16709 0.18002 0.041690 0.023237 0.07159 0.21112 1.003
4     sigma_obs 0.03155 0.03123 0.004613 0.004471 0.02457 0.03965 1.001
  ess_bulk ess_tail
1    921.8     1389
2    921.8     1389
3    926.6     1266
4   2450.5     4272
```

Los parámetros estimados los interpretamos como sigue:

- σ_θ tiene un valor cercano a 0.015 que corresponde a $\sigma_{\text{grados}} = 0.8$. De acuerdo a los datos obtenidos los jugadores de golf cometen errores de ángulo de **casi** un 1°. Si comparamos este valor con el de modelos anteriores podemos notar que al incluir errores de precisión en la fuerza de tiro ésta desviación disminuye. Ya no es necesario corregir con ángulos lo que se puede explicar de otra forma, esta correlación la podemos ver gráficamente por medio de un diagrama de dispersión como abajo.
- σ_f tiene un valor esperado de 0.17, lo cual implica un error del 17 % debido a la errores en distancia producto de la fuerza de tiro.
- σ_{obs} tiene un valor de 0.03 lo cual incide en errores atribuibles a medición del 3 puntos porcentuales.



La aparente bimodalidad de los gráficos de dispersión se podría explicar a traves del efecto de tener mediciones de dos tipos. Un tipo son los datos originales en los que parece haber un número limitado de registrados, y las nuevas observaciones de Broadie que tienen un número muy grande observaciones a distintas distancias.



3.7.1. Tarea: Exploraremos algunas rutas de mejora del modelo.

1. Por un lado exploraremos eliminar uno de los componentes redundantes. Para esto elimina el supuesto de la fuerza de tiro y reajusta el modelo con la aproximación continua.
2. Incorpora un modelo jerárquico para ajustar el modelo que incorpore errores observacionales para las dos poblaciones de datos. Es decir, un modelo que tenga una $\sigma_{\text{obs},1}$ para los datos del primer conjunto de observaciones y $\sigma_{\text{obs},2}$ para los datos del segundo.
3. ¿Qué conclusiones obtienes?

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. https://mc-stan.org/users/documentation/case-studies/golf.html, 2019. 12, 20, 22

[2] A. Gelman and D. Nolan. A Probability Model for Golf Putting. *Teaching Statistics*, 24(3):93–95, 2002. ISSN 1467-9639. . 12

[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, A. Vehtari, D. Simpson, C. C. Margossian, B. Carpenter, Y. Yao, L. Kennedy, J. Gabry, P.-C. Bürkner, and M. Modrák. Bayesian workflow. *arXiv preprint arXiv:2011.01808*, 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, nov 2021. ISBN 978-1-00-301916-9. . 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, 21