INSTITUTO TECNOLÓGICO AUTÓNOMO
DE MÉXICO

PRÁCTICA FINAL METODOS BAYESIANOS

PRIMAVERA 2021

EQUIPO 08

Dorely Morales Santiago 178095

Rodrigo Suárez Segovia 191351

Zarazúa Cruz Guillermo 159396

Entrega:

Enviar por correo electrónico una carpeta comprimida (`equipo-xx.zip`) que incluya datos y codigo de solución a mas tardar el 18 de Mayo antes de las 11:59pm (medianoche). El asunto deberá ser `[MB - 2021] Final Equipo XX` , donde reemplazarás `XX` con el codigo de tu equipo. No se aceptarán entregas extemporáneas. Será mejor entregar un examen resuelto parcialmente, que no entregar nada.

Instrucciones:

- Tus respuestas deben ser claras y debes explicar los resultados, incluye también tus procedimientos/código de manera ordenada, y el código comentado.
- Se evaluará la presentación de resultados (calidad de las gráficas, tablas, …).
- Las sesión del Martes 11 de Mayo será destinada a responder dudas del examen. Para esto se reservará una media hora para dudas (dependerá de la agenda cuál será el momento mas oportuno para abrir el espacio).
- Se podrá usar el foro de discusión para realizar preguntas y afinar detalles que no queden claros.

- No pueden compartir soluciones entre diferentes equipos.
- Al entregar este examen afirmas que el trabajo se realizó sólo con tu compañeros de equipo. El material que utilizaste para apoyarte consistió de las notas en clase (pdfs en Canvas), el codigo fuente de las notas en el repositorio de Github.
- Al entregar estás dando tu consentimiento para que bajo sospecha y suficiente evidencia de copia se anule tu evaluación.
- La carpeta comprimida deberá incluir la resolución del examen también en formato `.html`. La evaluación será completamente sobre el `html` y el código fuente será utilizado para verificar detalles adicionales. Si el `html` no incluye alguna sección de la evaluación se tomará dicha sección como **no entregado**.

Ponderación:

El examen está compuesto por cuatro apartados cuyos pesos son los siguientes:
- Águilas (15%),
- Huracanes (45%),
- Omega-3 (15%),
- Vacas (25%).

```
sin_lineas <- theme(panel.grid.major = element_blank(), panel.grid.minor = element_blank())
sin_leyenda <- theme(legend.position = "none")
sin_ejes <- theme(axis.ticks = element_blank(),
                  axis.text = element_blank())
```

```
# Carga de las librerias adicionales utilizadas por el Equipo 8

# Librería:             Funciones utilizadas:
# summarytools          dfSummary
# latex2exp             TeX
# MASS                  Base de datos eagles
# kableExtra            Formato tablas (html)
# tidybayes
# rethinking            Base de datos Hurricanes
# brms                  modelos binomiales

# Creamos una lista con las librerías, luego obtenemos las que no se tienen
# instaladas, luego instalamos las que hagan falta
lib_proy <- c('summarytools','latex2exp', 'MASS', 'kableExtra', 'tidybayes', 'brms', 'ggplot2')
lib_proy_faltantes <- lib_proy[!(lib_proy %in% installed.packages()[ , "Package"])]
if(length(lib_proy_faltantes)) install.packages(lib_proy_faltantes)

# Para instalar la librería "rethinking" usar los siguientes comandos:
# install.packages(c("coda","mvtnorm","devtools","loo","dagitty")) #CHECAR SI ESTE ES NECESARIO
# devtools::install_github("rmcelreath/rethinking")

# Cargamos todas las librerías
lapply(c(lib_proy, 'rethinking'), require, character.only = TRUE)


# Los modelos se guardarán en la carpeta fits, la cual debe existir donde está el este .Rmd
# La siguiente línea checa si existe dicha carperta, y si no la crea.
dir.create(file.path(getwd(), 'fits'), showWarnings = FALSE)
```

1. Modelos de conteo: Águilas

Los datos contenidos en `MASS` (eagles) son registros intento de robo entre águilas blancas en el estado de Washington. Ve la ayuda para mayor detalle en el conjunto de datos:

```
data(eagles)

#?eagles
eagles %>%
  kbl(digits=2, format.args = list(big.mark = ",")) %>%
  kable_styling(bootstrap_options = "striped", full_width = F)
```

y	n	P	A	V
17	24	L	A	L
29	29	L	A	S
17	27	L	I	L
20	20	L	I	S
1	12	S	A	L
15	16	S	A	S
0	28	S	I	L
1	4	S	I	S

```
descr(eagles) %>%
  kbl(digits=2, format.args = list(big.mark = ",")) %>%
  kable_styling(bootstrap_options = "striped", full_width = F)
```

	n	y
Mean	20.00	12.50
Std.Dev	8.83	10.66
Min	4.00	0.00
Q1	14.00	1.00
Median	22.00	16.00
Q3	27.50	18.50
Max	29.00	29.00
MAD	8.90	12.60
IQR	12.25	16.75
CV	0.44	0.85
Skewness	-0.55	0.01
SE.Skewness	0.75	0.75
Kurtosis	-1.29	-1.67
N.Valid	8.00	8.00
Pct.Valid	100.00	100.00

```
# Si se quisiera visualizar el summary corriendo solo este chunck hacerlo con:
# print(dfSummary(eagles, plain.ascii=FALSE, style="grid", valid.col=FALSE), method="render")

# Con lo sisuientes parámetros (incluido la opción del chunk "results='asis'", puede
# visualizarse el summary cuando se genera el html)
dfSummary(eagles, plain.ascii = FALSE, style = "grid", graph.magnif = 0.75, valid.col = FALSE, tmp.img.dir = "/tmp")
```

Data Frame Summary

eagles

Dimensions: 8 x 5
Duplicates: 0

No	Variable	Stats / Values	Freqs (% of Valid)	Graph	Missing
1	y [integer]	Mean (sd) : 12.5 (10.7) min < med < max: 0 < 16 < 29 IQR (CV) : 16.8 (0.9)	0 : 1 (12.5%) 1 : 2 (25.0%) 15 : 1 (12.5%) 17 : 2 (25.0%) 20 : 1 (12.5%) 29 : 1 (12.5%)		0 (0.0%)
2	n [integer]	Mean (sd) : 20 (8.8) min < med < max: 4 < 22 < 29 IQR (CV) : 12.2 (0.4)	4 : 1 (12.5%) 12 : 1 (12.5%) 16 : 1 (12.5%) 20 : 1 (12.5%) 24 : 1 (12.5%) 27 : 1 (12.5%) 28 : 1 (12.5%) 29 : 1 (12.5%)		0 (0.0%)
3	P [factor]	1. L 2. S	4 (50.0%) 4 (50.0%)		0 (0.0%)
4	A [factor]	1. A 2. I	4 (50.0%) 4 (50.0%)		0 (0.0%)
5	V [factor]	1. L 2. S	4 (50.0%) 4 (50.0%)		0 (0.0%)

Mientras un águila se alimenta, a veces otra se abalanza y trata de robar el salmón. Llamemos al águila que se está alimentando la “víctima” y al ladrón el “pirata”. Utiliza los datos disponibles para construir un GLM binomial para predecir los intentos exitosos de piratería.

Inciso **1.a**

Considera el modelo:

$$
\begin{aligned}
y_i &\sim \text{Binomial}(n_i, p_i)\,, \\
\text{logit}(p_i) &= \alpha + \beta_P P_i + \beta_V V_i + \beta_A A_i\,, \\
\alpha &\sim N(0, 1.5)\,, \\
\beta_P, \beta_V, \beta_A &\sim N(0, 0.5)\,,
\end{aligned}
$$

yi~Binomial(ni,pi),logit(pi)=α+βPPi+βVVi+βAAi,α~N(0,1.5),βP,βV,βA~N(0,0.5),

donde y es el número de intentos exitosos, n es el número total de intentos, P es una variable ficticia que indica si el pirata tenía un tamaño corporal grande o no, V es una variable ficticia que indica si la víctima tenía o no un tamaño corporal grande, y finalmente A es una variable ficticia que indica si el pirata era o no un adulto. Ajusta el modelo anterior a los datos de las águilas con la herramienta de tu preferencia e interpreta las estimaciones.

Respuesta

```
# Guardamos el dataset en otra variable para agregar unas variables
aguilas <- eagles
# Creamos variables indicadoras (dummies) para las varaibles asociadas a los atributos del ave pirata y víctima,
# con la finalidad de poderlo meter así al modelo
aguilas$pirataGrande <- ifelse(aguilas$P == "L", 1, 0)
aguilas$victimaGrande <- ifelse(aguilas$V == "L", 1, 0)
aguilas$pirataAdulto <- ifelse(aguilas$A == "A", 1, 0)
```

```
fit_1a<-brm(data=aguilas,
            family = binomial,
            y | trials(n) ~ 1 + pirataGrande + victimaGrande + pirataAdulto, #el evento si el robo fue exitoso como un
a binomial condicionada en el número de intentos
            prior = c(prior(normal(0, 1.5), class = Intercept),
                      prior(normal(0, 0.5), class = b)),
            seed=SEED,
            sample_prior = T, #pedimos muestras de la previa
            file = "fits/aguilas.fit1a")

summary(fit_1a, digits = 2)
```

```
##  Family: binomial
##   Links: mu = logit
## Formula: y | trials(n) ~ 1 + pirataGrande + victimaGrande + pirataAdulto
##    Data: aguilas (Number of observations: 8)
## Samples: 4 chains, each with iter = 2000; warmup = 1000; thin = 1;
##          total post-warmup samples = 4000
##
## Population-Level Effects:
##               Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## Intercept         0.31      0.39    -0.41     1.09 1.00     3465     2849
## pirataGrande      1.64      0.31     1.03     2.24 1.00     3715     2980
## victimaGrande    -1.70      0.32    -2.34    -1.07 1.00     3940     3205
## pirataAdulto      0.65      0.31     0.05     1.28 1.00     3866     2646
##
## Samples were drawn using sampling(NUTS). For each parameter, Bulk_ESS
## and Tail_ESS are effective sample size measures, and Rhat is the potential
## scale reduction factor on split chains (at convergence, Rhat = 1).
```

```
print(fit_1a) #imprimimos el objeto con el modelo
```

```
##  Family: binomial
##   Links: mu = logit
## Formula: y | trials(n) ~ 1 + pirataGrande + victimaGrande + pirataAdulto
##    Data: aguilas (Number of observations: 8)
## Samples: 4 chains, each with iter = 2000; warmup = 1000; thin = 1;
##          total post-warmup samples = 4000
##
## Population-Level Effects:
##               Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## Intercept         0.31      0.39    -0.41     1.09 1.00     3465     2849
## pirataGrande      1.64      0.31     1.03     2.24 1.00     3715     2980
## victimaGrande    -1.70      0.32    -2.34    -1.07 1.00     3940     3205
## pirataAdulto      0.65      0.31     0.05     1.28 1.00     3866     2646
##
## Samples were drawn using sampling(NUTS). For each parameter, Bulk_ESS
## and Tail_ESS are effective sample size measures, and Rhat is the potential
## scale reduction factor on split chains (at convergence, Rhat = 1).
```

```
# Una forma alternativa para ajustar el modelo es con la función "ulam" de la librería de "rethinking"

# Definimos el modelo
f <- alist(
  y ~ dbinom(n, p),
  logit(p) <- a + bP*pirataGrande + bV*victimaGrande + bA*pirataAdulto,
  a ~ dnorm(0, 1.5),
  bP ~ dnorm(0, .5),
  bV ~ dnorm(0, .5),
  bA ~ dnorm(0, .5)
)

# Ajustamos un modelo con la función "ulam" (Hamiltonian Monte Carlo with)
fit_1a_2 <- ulam(f, data = aguilas, chains = 4, log_lik = TRUE) #fit_1a_2 object itself is in the @stanfit slot. Anyth
ing you'd do with a Stan model can be done with that slot directly
```

```
##
## SAMPLING FOR MODEL '1be8945cf149fcc715f9be51bd55a3ac' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 0 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0 seconds.
## Chain 1: Adjust your expectations accordingly!
## Chain 1:
## Chain 1:
## Chain 1: Iteration:   1 / 1000 [  0%]  (Warmup)
## Chain 1: Iteration: 100 / 1000 [ 10%]  (Warmup)
## Chain 1: Iteration: 200 / 1000 [ 20%]  (Warmup)
## Chain 1: Iteration: 300 / 1000 [ 30%]  (Warmup)
## Chain 1: Iteration: 400 / 1000 [ 40%]  (Warmup)
## Chain 1: Iteration: 500 / 1000 [ 50%]  (Warmup)
## Chain 1: Iteration: 501 / 1000 [ 50%]  (Sampling)
## Chain 1: Iteration: 600 / 1000 [ 60%]  (Sampling)
## Chain 1: Iteration: 700 / 1000 [ 70%]  (Sampling)
## Chain 1: Iteration: 800 / 1000 [ 80%]  (Sampling)
## Chain 1: Iteration: 900 / 1000 [ 90%]  (Sampling)
## Chain 1: Iteration: 1000 / 1000 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 0.046 seconds (Warm-up)
## Chain 1:                0.038 seconds (Sampling)
## Chain 1:                0.084 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL '1be8945cf149fcc715f9be51bd55a3ac' NOW (CHAIN 2).
## Chain 2:
## Chain 2: Gradient evaluation took 0 seconds
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0 seconds.
## Chain 2: Adjust your expectations accordingly!
## Chain 2:
## Chain 2:
## Chain 2: Iteration:   1 / 1000 [  0%]  (Warmup)
## Chain 2: Iteration: 100 / 1000 [ 10%]  (Warmup)
## Chain 2: Iteration: 200 / 1000 [ 20%]  (Warmup)
## Chain 2: Iteration: 300 / 1000 [ 30%]  (Warmup)
## Chain 2: Iteration: 400 / 1000 [ 40%]  (Warmup)
## Chain 2: Iteration: 500 / 1000 [ 50%]  (Warmup)
## Chain 2: Iteration: 501 / 1000 [ 50%]  (Sampling)
## Chain 2: Iteration: 600 / 1000 [ 60%]  (Sampling)
## Chain 2: Iteration: 700 / 1000 [ 70%]  (Sampling)
## Chain 2: Iteration: 800 / 1000 [ 80%]  (Sampling)
## Chain 2: Iteration: 900 / 1000 [ 90%]  (Sampling)
## Chain 2: Iteration: 1000 / 1000 [100%]  (Sampling)
## Chain 2:
## Chain 2:  Elapsed Time: 0.043 seconds (Warm-up)
## Chain 2:                0.032 seconds (Sampling)
## Chain 2:                0.075 seconds (Total)
## Chain 2:
##
## SAMPLING FOR MODEL '1be8945cf149fcc715f9be51bd55a3ac' 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 / 1000 [  0%]  (Warmup)
## Chain 3: Iteration: 100 / 1000 [ 10%]  (Warmup)
## Chain 3: Iteration: 200 / 1000 [ 20%]  (Warmup)
## Chain 3: Iteration: 300 / 1000 [ 30%]  (Warmup)
## Chain 3: Iteration: 400 / 1000 [ 40%]  (Warmup)
## Chain 3: Iteration: 500 / 1000 [ 50%]  (Warmup)
```

```
## Chain 3: Iteration: 501 / 1000 [ 50%]  (Sampling)
## Chain 3: Iteration: 600 / 1000 [ 60%]  (Sampling)
## Chain 3: Iteration: 700 / 1000 [ 70%]  (Sampling)
## Chain 3: Iteration: 800 / 1000 [ 80%]  (Sampling)
## Chain 3: Iteration: 900 / 1000 [ 90%]  (Sampling)
## Chain 3: Iteration: 1000 / 1000 [100%]  (Sampling)
## Chain 3:
## Chain 3:  Elapsed Time: 0.043 seconds (Warm-up)
## Chain 3:                0.037 seconds (Sampling)
## Chain 3:                0.08 seconds (Total)
## Chain 3:
##
## SAMPLING FOR MODEL '1be8945cf149fcc715f9be51bd55a3ac' NOW (CHAIN 4).
## Chain 4:
## Chain 4: Gradient evaluation took 0 seconds
## Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 0 seconds.
## Chain 4: Adjust your expectations accordingly!
## Chain 4:
## Chain 4:
## Chain 4: Iteration:   1 / 1000 [  0%]  (Warmup)
## Chain 4: Iteration: 100 / 1000 [ 10%]  (Warmup)
## Chain 4: Iteration: 200 / 1000 [ 20%]  (Warmup)
## Chain 4: Iteration: 300 / 1000 [ 30%]  (Warmup)
## Chain 4: Iteration: 400 / 1000 [ 40%]  (Warmup)
## Chain 4: Iteration: 500 / 1000 [ 50%]  (Warmup)
## Chain 4: Iteration: 501 / 1000 [ 50%]  (Sampling)
## Chain 4: Iteration: 600 / 1000 [ 60%]  (Sampling)
## Chain 4: Iteration: 700 / 1000 [ 70%]  (Sampling)
## Chain 4: Iteration: 800 / 1000 [ 80%]  (Sampling)
## Chain 4: Iteration: 900 / 1000 [ 90%]  (Sampling)
## Chain 4: Iteration: 1000 / 1000 [100%]  (Sampling)
## Chain 4:
## Chain 4:  Elapsed Time: 0.037 seconds (Warm-up)
## Chain 4:                0.039 seconds (Sampling)
## Chain 4:                0.076 seconds (Total)
## Chain 4:
```

Inciso **1.b**

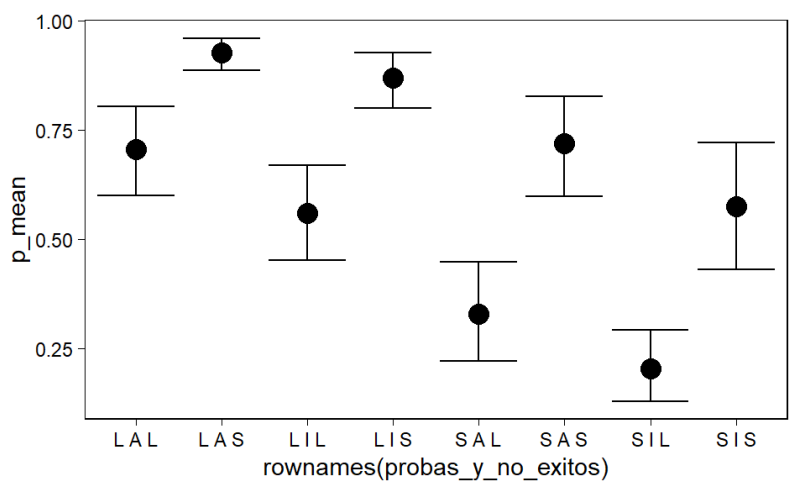
Luego grafica las predicciones posteriores. Para esto calcula y muestra tanto: 1) la predicción de la probabilidad de éxito y su intervalo de credibilidad de 89% para cada observación en los datos; como: 2) el número de éxitos y su intervalo del 89%. ¿Qué información proporciona cada tipo de predicción posterior?

Respuesta

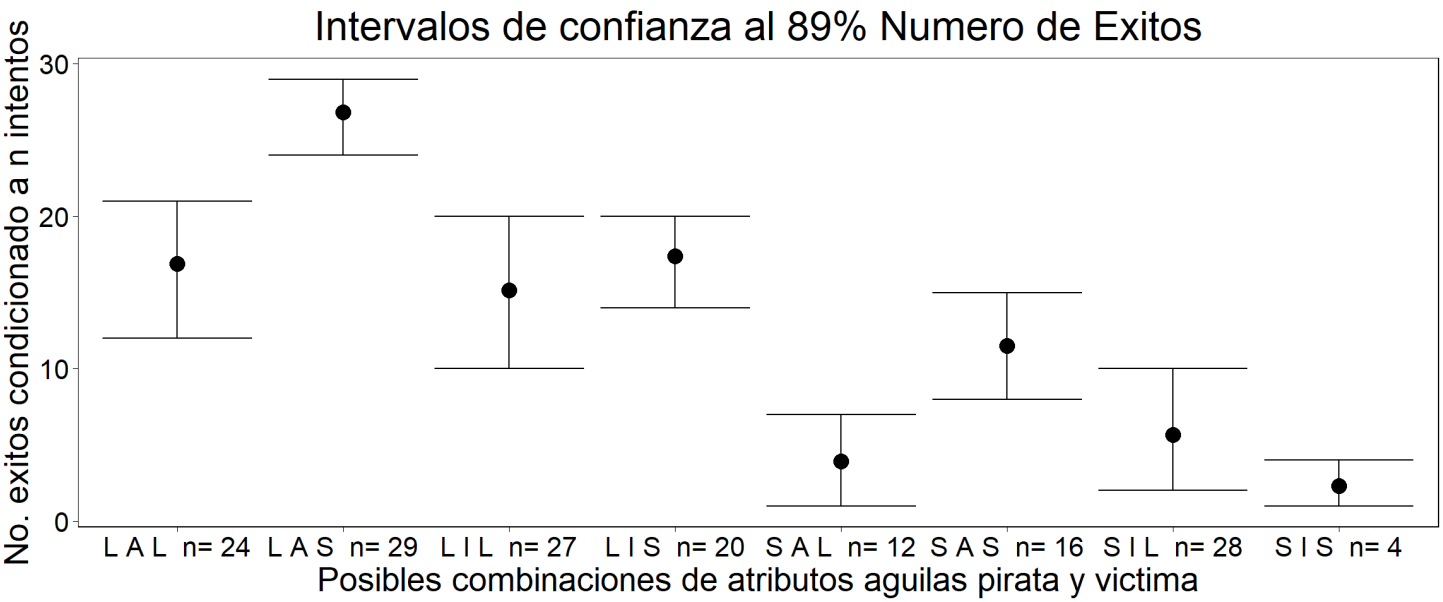
```
probas_y_no_exitos <- setNames(data.frame(matrix(ncol = 6, nrow = 8)), c("p_mean", "p_li_0.055", "p_ls_0.945", "ne_mea
n", "ne_li_0.055", "ne_ls_0.945"))
rownames(probas_y_no_exitos) <- paste(aguilas$P,aguilas$A,aguilas$V)

simulacion_coeficientes <- as.matrix(posterior_samples(fit_1a)[,1:4]) # Fijas semilla (obtiene 4000)
posibles_resultados <- as.matrix(cbind(rep(1,8),aguilas[1:8,c(6,7,8)]))
simulacion_probas <- logistic(simulacion_coeficientes%*%t(posibles_resultados))

probas_y_no_exitos$p_mean <- apply(X = simulacion_probas, MARGIN = 2, FUN = mean)
probas_y_no_exitos[,c("p_li_0.055","p_ls_0.945")] <- t(apply(X = simulacion_probas, MARGIN = 2, FUN = PI, prob=c(0.89
)))

# Para obtener la media, error e intervalos de confianza de las simulacion de la posteriro puede ejecutarse: predict(f
it_1a)
# Para obtener todas las simulaciones de la posterior (4000) (con las cuales se pueden obtener los resultados de predi
t(fit_1a))
aux<-posterior_predict(fit_1a) # recordar fijar semilla (obtiene 1000)
probas_y_no_exitos$ne_mean <- apply(X=aux, MARGIN=2, FUN=mean)
probas_y_no_exitos[,c("ne_li_0.055","ne_ls_0.945")] <- t(apply(X = aux, MARGIN = 2, FUN = PI, prob=c(0.89)))

probas_y_no_exitos
```

```
##          p_mean p_li_0.055 p_ls_0.945  ne_mean ne_li_0.055 ne_ls_0.945
## L A L 0.7062038  0.5992820  0.8036829 16.89000          12          21
## L A S 0.9272902  0.8860220  0.9599489 26.84750          24          29
## L I L 0.5596749  0.4517629  0.6681632 15.13050          10          20
## L I S 0.8698191  0.7995165  0.9269190 17.37725          14          20
## S A L 0.3275514  0.2201343  0.4478833  3.91325           1           7
## S A S 0.7182579  0.5976732  0.8260330 11.48550           8          15
## S I L 0.2041579  0.1285411  0.2916357  5.67400           2          10
## S I S 0.5746004  0.4308049  0.7211697  2.29225           1           4
```

```
ggplot(probas_y_no_exitos, aes(x = rownames(probas_y_no_exitos), y = p_mean)) +
  geom_point(size = 4) +
  geom_errorbar(aes(ymax = p_li_0.055, ymin = p_ls_0.945)) +
  sin_lineas
```



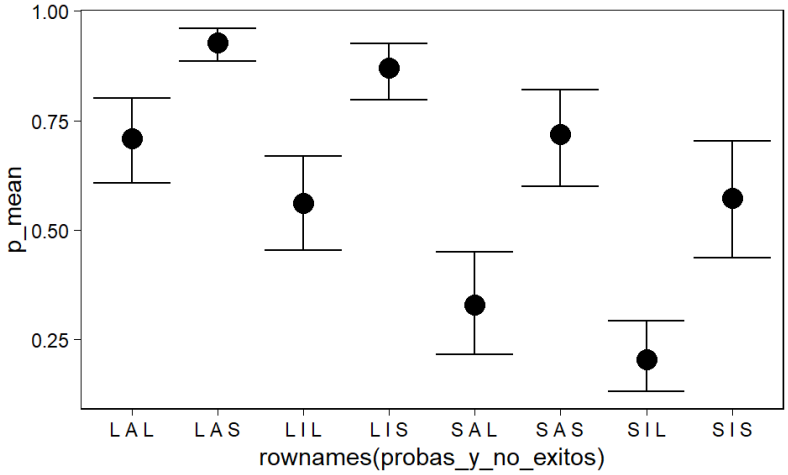

```
g1<-ggplot(probas_y_no_exitos, aes(x = paste(rownames(probas_y_no_exitos), " n=", aguilas$n), y = ne_mean)) +
  geom_point(size = 4) +
  geom_errorbar(aes(ymax = ne_li_0.055, ymin = ne_ls_0.945), axis.text.x = element_text(vjust=1)) +
  sin_lineas +
  theme(text = element_text(size=20), plot.title=element_text(hjust=0.5)) +
  ggtitle("Intervalos de confianza al 89% Numero de Exitos") +
  xlab("Posibles combinaciones de atributos aguilas pirata y victima") + ylab("No. exitos condicionado a n intentos")
g1
```



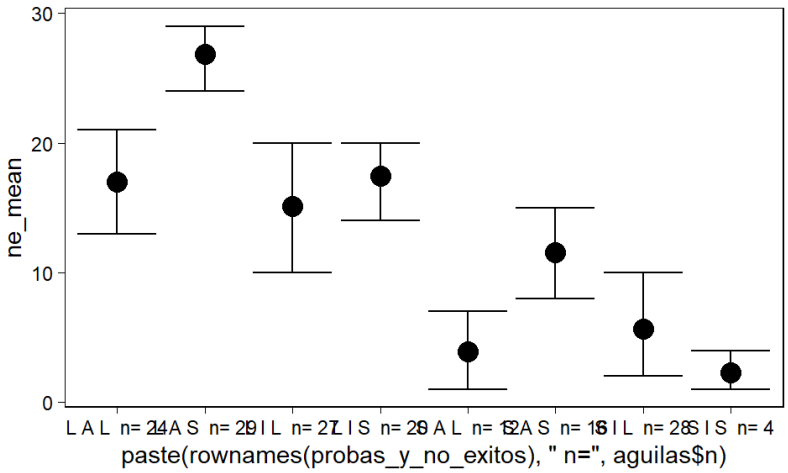

```
probas_y_no_exitos <- setNames(data.frame(matrix(ncol = 6, nrow = 8)), c("p_mean", "p_li_0.055", "p_ls_0.945", "ne_mea
n", "ne_li_0.055", "ne_ls_0.945"))
rownames(probas_y_no_exitos) <- paste(aguilas$P,aguilas$A,aguilas$V)


post <- extract.samples(fit_1a_2) # Fijas semilla (obtiene 2000)
simulacion_coeficientes <- as.matrix(cbind(post$a,post$bP, post$bV, post$bA)[1:2000,1:4])
posibles_resultados <- as.matrix(cbind(rep(1,8),aguilas[1:8,c(6,7,8)]))
simulacion_probas <- logistic(simulacion_coeficientes%*%t(posibles_resultados))

probas_y_no_exitos$p_mean <- apply(X = simulacion_probas, MARGIN = 2, FUN = mean)
probas_y_no_exitos[,c("p_li_0.055","p_ls_0.945")] <- t(apply(X = simulacion_probas, MARGIN = 2, FUN = PI, prob=c(0.89
)))

aux<-sim(fit_1a_2) # recordar fijar semilla (obtiene 1000)
probas_y_no_exitos$ne_mean <- apply(X=aux, MARGIN=2, FUN=mean)
probas_y_no_exitos[,c("ne_li_0.055","ne_ls_0.945")] <- t(apply(X = aux, MARGIN = 2, FUN = PI, prob=c(0.89)))


probas_y_no_exitos
```

```
##          p_mean p_li_0.055 p_ls_0.945 ne_mean ne_li_0.055 ne_ls_0.945
## L A L 0.7090378  0.6073064  0.8017895  16.979          13          21
## L A S 0.9276172  0.8857920  0.9604662  26.851          24          29
## L I L 0.5615006  0.4541474  0.6691173  15.141          10          20
## L I S 0.8697386  0.7972632  0.9262196  17.435          14          20
## S A L 0.3284835  0.2169809  0.4501799   3.891           1           7
## S A S 0.7179415  0.5998005  0.8203133  11.559           8          15
## S I L 0.2037992  0.1322530  0.2938424   5.651           2          10
## S I S 0.5732153  0.4369451  0.7032049   2.311           1           4
```

```
ggplot(probas_y_no_exitos, aes(x = rownames(probas_y_no_exitos), y = p_mean)) +
  geom_point(size = 4) +
  geom_errorbar(aes(ymax = p_li_0.055, ymin = p_ls_0.945)) +
  sin_lineas
```




```
ggplot(probas_y_no_exitos, aes(x = paste(rownames(probas_y_no_exitos), " n=", aguilas$n), y = ne_mean)) +
  geom_point(size = 4) +
  geom_errorbar(aes(ymax = ne_li_0.055, ymin = ne_ls_0.945)) +
  sin_lineas
```



Inciso **1.c**

Ahora intenta mejorar el modelo. Considera una interacción entre el tamaño y edad de los piratas. Compara la capacidad predictiva de los modelos. Interpreta los resultados.

Respuesta

Utilizaremos brm para el fit del modelo

```
fit_1c<-brm(data=aguilas,
            family = binomial,
            y | trials(n) ~ 1 + pirataGrande+victimaGrande+pirataAdulto+pirataGrande:pirataAdulto, #el evento si el ro
bo fue exitoso como una binomial condicionada en el número de intentos
            prior = c(prior(normal(0, 1.5), class = Intercept),
                      prior(normal(0, 0.5), class = b)),
            seed=SEED,
            sample_prior = T, #pedimos muestras de la previa
            file = "fits/aguilas.fit1c")
summary(fit_1c, digits = 2)
```

```
##  Family: binomial
##   Links: mu = logit
## Formula: y | trials(n) ~ 1 + pirataGrande + victimaGrande + pirataAdulto + pirataGrande:pirataAdulto
##    Data: aguilas (Number of observations: 8)
## Samples: 4 chains, each with iter = 2000; warmup = 1000; thin = 1;
##          total post-warmup samples = 4000
##
## Population-Level Effects:
##                           Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS
## Intercept                     0.33      0.37    -0.40     1.06 1.00     3813
## pirataGrande                  1.55      0.32     0.91     2.17 1.00     3969
## victimaGrande                -1.71      0.32    -2.34    -1.11 1.00     3988
## pirataAdulto                  0.55      0.33    -0.10     1.20 1.00     3315
## pirataGrande:pirataAdulto     0.30      0.38    -0.41     1.03 1.00     3475
##                           Tail_ESS
## Intercept                     3303
## pirataGrande                  2849
## victimaGrande                 3137
## pirataAdulto                  3240
## pirataGrande:pirataAdulto     3409
##
## Samples were drawn using sampling(NUTS). For each parameter, Bulk_ESS
## and Tail_ESS are effective sample size measures, and Rhat is the potential
## scale reduction factor on split chains (at convergence, Rhat = 1).
```

```
waic_1a<-waic(fit_1a) #calcula: elpd_waic que corresponde a la log-densidad; p_waic: el número efectivo de parámetros;
waic: la devianza del modelo así como suss errores estándar variando sobre todas las observaciones
waic_1c<-waic(fit_1c) #calcula: elpd_waic que corresponde a la log-densidad; p_waic: el número efectivo de parámetros;
waic: la devianza del modelo así como suss errores estándar variando sobre todas las observaciones
comparativo_waic_1c<-cbind(waic_1a,waic_1c) #combina resultados para comparar
comparativo_waic_1c
```

```
##              waic_1a    waic_1c
## estimates    Numeric,6  Numeric,6
## pointwise    Numeric,24 Numeric,24
## elpd_waic    -29.45912  -30.24113
## p_waic       8.288274   8.940337
## waic         58.91823   60.48227
## se_elpd_waic 6.151007   6.270307
## se_p_waic    2.999563   3.017251
## se_waic      12.30201   12.54061
```

Y los resultados arrojan que el modelo del inciso a) y c) (con y sin interacción respectivamente) tienen un desempeño prácticamente igual ya que la devianza es prácticamente igual 59.16 en contraste con 60.68.

Luego generamos un comparativo en términos de la log-densidad predictiva con el método loo_compare().

```
comparativo_diff_waic_1c<-loo_compare(waic_1a,waic_1c) #calcula comparativo de diferencia en términos de la log-densid
ad predictiva y un error estándar sobre esta diferencia (se_diff). Ordenando los modelos del mejor al peor
comparativo_diff_waic_1c #imprime comparativo
```

```
##        elpd_diff se_diff
## fit_1a  0.0       0.0
## fit_1c -0.8       0.8
```

Estos resultados si bien identifican al modelo del inciso a como el mejor pues tiene una mayor log-densidad predictiva. En realidad se aprecia una diferencia mínima en la log-densidad predictiva y el error estándar del modelo c) con respecto al del inciso a).

Por último comparamos los modelos con validación cruzada mediante la función loo():

```
loo_1a<-loo(fit_1a) #calcula: elpd_loo que corresponde a la log-densidad; p_loo: el número efectivo de parámetros; loo
ic: la devianza del modelo así como suss errores estándar (SE) variando sobre todas las observaciones con validación c
ruzada
loo_1a #imprime resultados
```

```
##
## Computed from 4000 by 8 log-likelihood matrix
##
##          Estimate   SE
## elpd_loo    -30.2  6.3
## p_loo         9.0  3.1
## looic        60.4 12.6
## ------
## Monte Carlo SE of elpd_loo is NA.
##
## Pareto k diagnostic values:
##                          Count Pct.    Min. n_eff
## (-Inf, 0.5]   (good)     2     25.0%   1062
##  (0.5, 0.7]   (ok)       4     50.0%   276
##    (0.7, 1]   (bad)      2     25.0%   54
##    (1, Inf)   (very bad) 0      0.0%   <NA>
## See help('pareto-k-diagnostic') for details.
```

```
loo_1c<-loo(fit_1c) #calcula: elpd_loo que corresponde a la log-densidad; p_loo: el número efectivo de parámetros; loo
ic: la devianza del modelo así como suss errores estándar (SE) variando sobre todas las observaciones con validación c
ruzada
loo_1c #imprime resultados
```

```
##
## Computed from 4000 by 8 log-likelihood matrix
##
##          Estimate   SE
## elpd_loo    -31.9  6.7
## p_loo        10.6  3.5
## looic        63.8 13.4
## ------
## Monte Carlo SE of elpd_loo is NA.
##
## Pareto k diagnostic values:
##                          Count Pct.    Min. n_eff
## (-Inf, 0.5]   (good)     1     12.5%   2733
##  (0.5, 0.7]   (ok)       3     37.5%   315
##    (0.7, 1]   (bad)      3     37.5%   28
##    (1, Inf)   (very bad) 1     12.5%   11
## See help('pareto-k-diagnostic') for details.
```

En ambos casos, el diagnóstico que loo arroja es que ambos modelos son malos! REVISAR

```
comparativo_diff_loo_1c<-loo_compare(loo_1a,loo_1c) #calcula comparativo de diferencia en términos de la log-densidad
 predictiva y un error estándar sobre esta diferencia (se_diff). Ordenando los modelos del mejor al peor
comparativo_diff_loo_1c #imprime comparativo
```

```
##        elpd_diff se_diff
## fit_1a  0.0       0.0
## fit_1c -1.7       1.1
```

SOLUCIÓN: Jup, there's not really much of a difference here. For the interaction model: the log-odds of successful piracy is just weakly bigger when the pirating individual is large and an adult. That is counter-intuitive, isn't it? It is worth pointing out that the individual parameters for these conditions show the expected effects and the identified negative effect of their interaction may be down to the sparsity of the underlying data and we are also highly uncertain of it's sign to begin with.

```
fit_1c_2 <- ulam(
  alist(
    y ~ dbinom(n, p),
    logit(p) <- a + bP*pirataGrande + bV*victimaGrande + bA*pirataAdulto + bPA*pirataGrande*pirataAdulto,
    a ~ dnorm(0, 1.5),
    bP ~ dnorm(0, .5),
    bV ~ dnorm(0, .5),
    bA ~ dnorm(0, .5),
    bPA ~ dnorm(0, .5)
  ),
  data = aguilas, chains = 4, log_lik = TRUE
)
```

```
##
## SAMPLING FOR MODEL '8decbc8cd62fc80a192884add8ec2943' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 0 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0 seconds.
## Chain 1: Adjust your expectations accordingly!
## Chain 1:
## Chain 1:
## Chain 1: Iteration:   1 / 1000 [  0%]  (Warmup)
## Chain 1: Iteration: 100 / 1000 [ 10%]  (Warmup)
## Chain 1: Iteration: 200 / 1000 [ 20%]  (Warmup)
## Chain 1: Iteration: 300 / 1000 [ 30%]  (Warmup)
## Chain 1: Iteration: 400 / 1000 [ 40%]  (Warmup)
## Chain 1: Iteration: 500 / 1000 [ 50%]  (Warmup)
## Chain 1: Iteration: 501 / 1000 [ 50%]  (Sampling)
## Chain 1: Iteration: 600 / 1000 [ 60%]  (Sampling)
## Chain 1: Iteration: 700 / 1000 [ 70%]  (Sampling)
## Chain 1: Iteration: 800 / 1000 [ 80%]  (Sampling)
## Chain 1: Iteration: 900 / 1000 [ 90%]  (Sampling)
## Chain 1: Iteration: 1000 / 1000 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 0.046 seconds (Warm-up)
## Chain 1:                0.052 seconds (Sampling)
## Chain 1:                0.098 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL '8decbc8cd62fc80a192884add8ec2943' NOW (CHAIN 2).
## Chain 2:
## Chain 2: Gradient evaluation took 0 seconds
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0 seconds.
## Chain 2: Adjust your expectations accordingly!
## Chain 2:
## Chain 2:
## Chain 2: Iteration:   1 / 1000 [  0%]  (Warmup)
## Chain 2: Iteration: 100 / 1000 [ 10%]  (Warmup)
## Chain 2: Iteration: 200 / 1000 [ 20%]  (Warmup)
## Chain 2: Iteration: 300 / 1000 [ 30%]  (Warmup)
## Chain 2: Iteration: 400 / 1000 [ 40%]  (Warmup)
## Chain 2: Iteration: 500 / 1000 [ 50%]  (Warmup)
## Chain 2: Iteration: 501 / 1000 [ 50%]  (Sampling)
## Chain 2: Iteration: 600 / 1000 [ 60%]  (Sampling)
## Chain 2: Iteration: 700 / 1000 [ 70%]  (Sampling)
## Chain 2: Iteration: 800 / 1000 [ 80%]  (Sampling)
## Chain 2: Iteration: 900 / 1000 [ 90%]  (Sampling)
## Chain 2: Iteration: 1000 / 1000 [100%]  (Sampling)
## Chain 2:
## Chain 2:  Elapsed Time: 0.045 seconds (Warm-up)
## Chain 2:                0.05 seconds (Sampling)
## Chain 2:                0.095 seconds (Total)
## Chain 2:
##
## SAMPLING FOR MODEL '8decbc8cd62fc80a192884add8ec2943' 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 / 1000 [  0%]  (Warmup)
## Chain 3: Iteration: 100 / 1000 [ 10%]  (Warmup)
## Chain 3: Iteration: 200 / 1000 [ 20%]  (Warmup)
## Chain 3: Iteration: 300 / 1000 [ 30%]  (Warmup)
## Chain 3: Iteration: 400 / 1000 [ 40%]  (Warmup)
## Chain 3: Iteration: 500 / 1000 [ 50%]  (Warmup)
```

```
## Chain 3: Iteration: 501 / 1000 [ 50%]  (Sampling)
## Chain 3: Iteration: 600 / 1000 [ 60%]  (Sampling)
## Chain 3: Iteration: 700 / 1000 [ 70%]  (Sampling)
## Chain 3: Iteration: 800 / 1000 [ 80%]  (Sampling)
## Chain 3: Iteration: 900 / 1000 [ 90%]  (Sampling)
## Chain 3: Iteration: 1000 / 1000 [100%]  (Sampling)
## Chain 3:
## Chain 3:  Elapsed Time: 0.048 seconds (Warm-up)
## Chain 3:                0.047 seconds (Sampling)
## Chain 3:                0.095 seconds (Total)
## Chain 3:
##
## SAMPLING FOR MODEL '8decbc8cd62fc80a192884add8ec2943' NOW (CHAIN 4).
## Chain 4:
## Chain 4: Gradient evaluation took 0 seconds
## Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 0 seconds.
## Chain 4: Adjust your expectations accordingly!
## Chain 4:
## Chain 4:
## Chain 4: Iteration:   1 / 1000 [  0%]  (Warmup)
## Chain 4: Iteration: 100 / 1000 [ 10%]  (Warmup)
## Chain 4: Iteration: 200 / 1000 [ 20%]  (Warmup)
## Chain 4: Iteration: 300 / 1000 [ 30%]  (Warmup)
## Chain 4: Iteration: 400 / 1000 [ 40%]  (Warmup)
## Chain 4: Iteration: 500 / 1000 [ 50%]  (Warmup)
## Chain 4: Iteration: 501 / 1000 [ 50%]  (Sampling)
## Chain 4: Iteration: 600 / 1000 [ 60%]  (Sampling)
## Chain 4: Iteration: 700 / 1000 [ 70%]  (Sampling)
## Chain 4: Iteration: 800 / 1000 [ 80%]  (Sampling)
## Chain 4: Iteration: 900 / 1000 [ 90%]  (Sampling)
## Chain 4: Iteration: 1000 / 1000 [100%]  (Sampling)
## Chain 4:
## Chain 4:  Elapsed Time: 0.048 seconds (Warm-up)
## Chain 4:                0.035 seconds (Sampling)
## Chain 4:                0.083 seconds (Total)
## Chain 4:
```

```
compare(fit_1a_2, fit_1c_2)
```

```
##              WAIC       SE     dWAIC      dSE    pWAIC    weight
## fit_1a_2 58.83631 11.56789 0.0000000       NA 8.205043 0.6184739
## fit_1c_2 59.80246 11.44676 0.9661515 1.399166 8.674317 0.3815261
```

```
# plot(coeftab(mH3ulam, mH3c),
#   labels = paste(rep(rownames(coeftab(mH3ulam, mH3c)@coefs), each = 2),
#     rep(c("Base", "Interac"), nrow(coeftab(mH3ulam, mH3c)@coefs) * 2),
#     sep = "-"
#   )
# )
```

2. Extensiones de modelos de conteo: huracanes

En 2014, se publicó un artículo titulado *"Female hurricanes are deadlier than male hurricanes"*. Como sugiere el título, el documento afirmó que los huracanes con nombres femeninos han causado una mayor pérdida de vidas, y la explicación que se da es que las personas inconscientemente califican a los huracanes femeninos como menos peligrosos y, por lo tanto, es menos probable que se necesite evacuar. Los estadísticos criticaron duramente el artículo después de su publicación. En esta sección, explorarás los datos completos utilizados en el artículo y considerarás la hipótesis que los huracanes con nombres femeninos son más letales. Carga los datos con:

```
data(Hurricanes)

Hurricanes %>%
  head() %>%
  kbl(digits=2, format.args = list(big.mark = ",")) %>%
  kable_styling(bootstrap_options = "striped", full_width = F)
```

name	year	deaths	category	min_pressure	damage_norm	female	femininity
Easy	1,950	2	3	960	1,590	1	6.78
King	1,950	4	3	955	5,350	0	1.39
Able	1,952	3	1	985	150	0	3.83
Barbara	1,953	1	1	987	58	1	9.83
Florence	1,953	0	1	985	15	1	8.33
Carol	1,954	60	3	960	19,321	1	8.11

```
descr(Hurricanes) %>%
  kbl(digits=2, format.args = list(big.mark = ",")) %>%
  kable_styling(bootstrap_options = "striped", full_width = F)
```

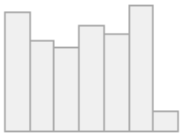
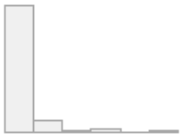
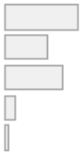
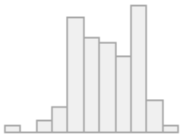
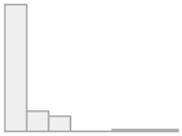
	category	damage_norm	deaths	female	femininity	min_pressure	year
Mean	2.09	7,269.78	20.65	0.67	6.78	964.91	1,982.09
Std.Dev	1.06	12,934.09	40.90	0.47	3.23	19.07	18.77
Min	1.00	1.00	0.00	0.00	1.06	909.00	1,950.00
Q1	1.00	240.00	2.00	0.00	2.67	950.00	1,964.50
Median	2.00	1,650.00	5.00	1.00	8.50	964.00	1,985.00
Q3	3.00	8,195.00	20.50	1.00	9.39	982.50	1,999.00
Max	5.00	75,000.00	256.00	1.00	10.44	1,003.00	2,012.00
MAD	1.48	2,360.30	5.93	0.00	1.98	22.98	26.69
IQR	2.00	7,917.50	18.25	1.00	6.72	32.25	34.25
CV	0.51	1.78	1.98	0.70	0.48	0.02	0.01
Skewness	0.55	3.21	3.62	-0.73	-0.63	-0.27	-0.13
SE.Skewness	0.25	0.25	0.25	0.25	0.25	0.25	0.25
Kurtosis	-0.54	12.02	14.86	-1.48	-1.37	-0.51	-1.35
N.Valid	92.00	92.00	92.00	92.00	92.00	92.00	92.00
Pct.Valid	100.00	100.00	100.00	100.00	100.00	100.00	100.00

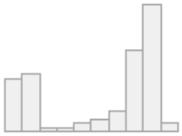
```
# Si se quisiera visualizar el summary corriendo solo este chunck hacerlo con:
# print(dfSummary(Hurricanes, plain.ascii=FALSE, style="grid", valid.col=FALSE), method="render")

# Con lo sisuientes parámetros (incluido la opción del chunk "results='asis'", puede
# visualizarse el summary cuando se genera el html)
dfSummary(Hurricanes, plain.ascii = FALSE, style = "grid", graph.magnif = 0.75, valid.col = FALSE, tmp.img.dir = "/tm
p")
```

Data Frame Summary

Hurricanes

Dimensions: 92 x 8
Duplicates: 0

No	Variable	Stats / Values	Freqs (% of Valid)	Graph	Missing
1	name [factor]	1. Able 2. Agnes 3. Alex 4. Alicia 5. Allen 6. Alma 7. Andrew 8. Babe 9. Barbara 10. Belle [73 others]	1 (1.1%) 1 (1.1%) 1 (1.1%) 1 (1.1%) 1 (1.1%) 1 (1.1%) 1 (1.1%) 1 (1.1%) 1 (1.1%) 1 (1.1%) 82 (89.1%)		0 (0.0%)
2	year [integer]	Mean (sd) : 1982.1 (18.8) min < med < max: 1950 < 1985 < 2012 IQR (CV) : 34.2 (0)	49 distinct values		0 (0.0%)
3	deaths [integer]	Mean (sd) : 20.7 (40.9) min < med < max: 0 < 5 < 256 IQR (CV) : 18.2 (2)	34 distinct values		0 (0.0%)
4	category [integer]	Mean (sd) : 2.1 (1.1) min < med < max: 1 < 2 < 5 IQR (CV) : 2 (0.5)	1 : 36 (39.1%) 2 : 21 (22.8%) 3 : 28 (30.4%) 4 : 5 (5.4%) 5 : 2 (2.2%)		0 (0.0%)
5	min_pressure [integer]	Mean (sd) : 964.9 (19.1) min < med < max: 909 < 964 < 1003 IQR (CV) : 32.2 (0)	52 distinct values		0 (0.0%)
6	damage_norm [integer]	Mean (sd) : 7269.8 (12934.1) min < med < max: 1 < 1650 < 75000 IQR (CV) : 7917.5 (1.8)	86 distinct values		0 (0.0%)
7	female [integer]	Min : 0 Mean : 0.7 Max : 1	0 : 30 (32.6%) 1 : 62 (67.4%)		0 (0.0%)

No	Variable	Stats / Values	Freqs (% of Valid)	Graph	Missing
8	femininity [numeric]	Mean (sd) : 6.8 (3.2) min < med < max: 1.1 < 8.5 < 10.4 IQR (CV) : 6.7 (0.5)	55 distinct values		0 (0.0%)

Familiarízate con las columnas inspeccionando la ayuda `?Hurricanes` . En este problema, te concentrarás en predecir muertes usando la feminidad de cada nombre del huracán.

Inciso **2.a**

Ajustaremos e interpretaremos el modelo más simple posible, un modelo de Poisson de muertes utilizando la feminidad como predictor. Puede utilizar quap o ulam. Compara el modelo a un modelo de muertes Poisson con sólo intercepto. ¿Qué tan fuerte es la asociación entre la feminidad del nombre y el número de muertes? ¿Qué tormentas ajustan bien con el modelo? ¿Qué tormentas son las que no son tan fáciles de predecir?

Respuesta

1. name : Nombre del Huracán
2. year : Año del Huracán
3. deaths : Número de muertes
4. category : Severidad del fenómeno
5. min_pressure : Minimum pressure, a measure of storm strength; low is stronger
6. damage_norm : Normalized estimate of damage in dollars
7. female : Indicator variable for female name
8. femininity : 1-11 scale from totally masculine (1) to totally feminine (11) for name. Average of 9 scores from 9 raters.

//

Question: In 2014, a paper was published that was entitled “Female hurricanes are deadlier than male hurricanes.” As the title suggests, the paper claimed that hurricanes with female names have caused greater loss of life, and the explanation given is that people unconsciously rate female hurricanes as less dangerous and so are less likely to evacuate. Statisticians severely criticized the paper after publication. Here, you’ll explore the complete data used in the paper and consider the hypothesis that hurricanes with female names are deadlier. Load the data with:

Acquaint yourself with the columns by inspecting the help ?Hurricanes. In this problem, you’ll focus on predicting deaths using femininity of each hurricane’s name.

Fit and interpret the simplest possible model, a Poisson model of deaths using femininity as a predictor. You can use quap or ulam. Compare the model to an intercept-only Poisson model of deaths. How strong is the association between femininity of name and deaths? Which storms does the model fit (retrodict) well? Which storms does it fit poorly?

Answer: First, let’s prepare the data:

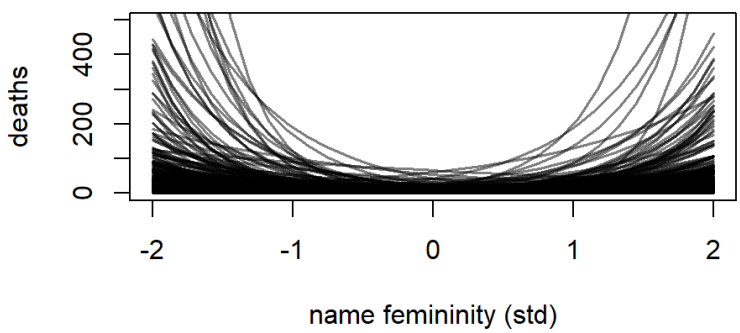
```
d <- Hurricanes # load data on object called d
d$fem_std <- (d$femininity - mean(d$femininity)) / sd(d$femininity) # standardised femininity
dat <- list(D = d$deaths, F = d$fem_std)
```

Now that we have standardised data for the feminity of our hurricane names which makes priors easier to formulate, we can specify our initial model idea:

```
# model formula
f <- alist(
  D ~ dpois(lambda), # poisson outcome distribution
  log(lambda) <- a + bF * F, # log-link for lambda with linear model
  # priors in log-space, 0 corresponds to outcome of 1
  a ~ dnorm(1, 1),
  bF ~ dnorm(0, 1)
)
```

But are these priors any good? Let's simulate them why don't we:

```
N <- 1e3
a <- rnorm(N, 1, 1)
bF <- rnorm(N, 0, 1)
F_seq <- seq(from = -2, to = 2, length.out = 30) # sequence from -2 to 2 because femininity data is standardised
plot(NULL,
  xlim = c(-2, 2), ylim = c(0, 500),
  xlab = "name femininity (std)", ylab = "deaths"
)
for (i in 1:N) {
  lines(F_seq,
    exp(a[i] + bF[i] * F_seq), # inverse link to get outcome scale
    col = grau(), lwd = 1.5
  )
}
```



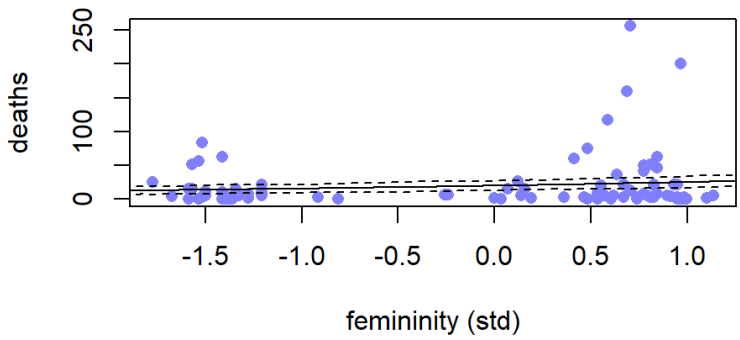
I'd think that's pretty alright. We allow for both positive and negative trends between death toll and femininity of hurricane name, but don't have a lot of explosive trends in our priors. These strong trends are quite unintuitive. Our vast majority of trends however are very ambiguous and so I proceed with these priors and run the model:

```
mH1 <- ulam(f, data = dat, chains = 4, cores = 4, log_lik = TRUE)
precis(mH1)
```

```
##         mean         sd      5.5%     94.5%    n_eff     Rhat4
## a  2.9984201 0.02290187 2.9625448 3.0351293 1259.936 0.9993086
## bF 0.2385046 0.02590928 0.1966015 0.2803355 1033.830 1.0011779
```

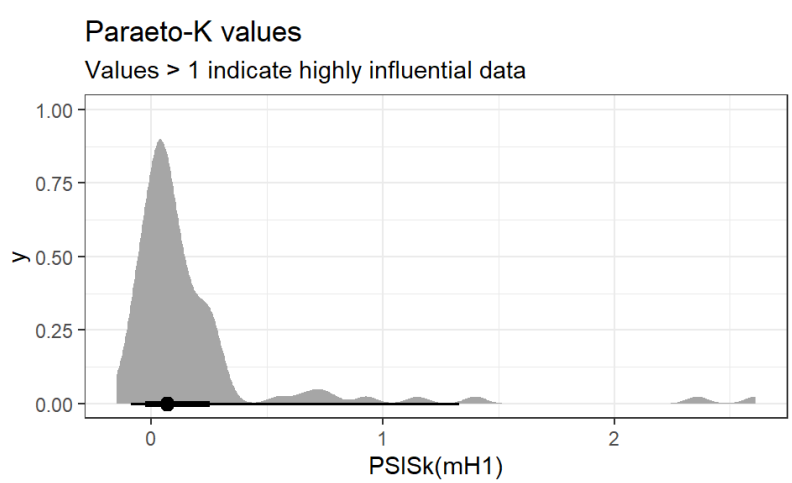
So according to this, there is a positive relationship between hurricane name femininity and death toll. Which hurricanes do we actually retrodict well, though? Let's plot, this:

```
# plot raw data
plot(dat$F, dat$D,
  pch = 16, lwd = 2,
  col = rangi2, xlab = "femininity (std)", ylab = "deaths"
)
# compute model-based trend
pred_dat <- list(F = seq(from = -2, to = 2, length.out = 1e2))
lambda <- link(mH1, data = pred_dat) # predict deaths
lambda.mu <- apply(lambda, 2, mean) # get mean prediction
lambda.PI <- apply(lambda, 2, PI) # get prediction interval
# superimpose trend
lines(pred_dat$F, lambda.mu)
shade(lambda.PI, pred_dat$F)
# compute sampling distribution
deaths_sim <- sim(mH1, data = pred_dat) # simulate posterior observations
deaths_sim.PI <- apply(deaths_sim, 2, PI) # get simulation interval
# superimpose sampling interval as dashed lines
lines(pred_dat$F, deaths_sim.PI[1, ], lty = 2)
lines(pred_dat$F, deaths_sim.PI[2, ], lty = 2)
```



Ok. There is quite a bit to unpack here. First of all, our model does not retrodict many of the hurricanes well even though it is quite certain of its predictions (grey shaded area which is hardly visible). Quite obviously, this model misses many of the hurricane death tolls to the right hand side of the above plot. This is a clear sign of over-dispersion which our model failed to account for. The weak, positive trend we are seeing here seems to be informed largely by these highly influential data points. We can assess whether and how influential some data points are with the Paraeto-K values (anything above 1 indicates an influential data point) following:

```
ggplot(as.data.frame(PSISk(mH1)), aes(x = PSISk(mH1))) +
  stat_halfeye() +
  theme_bw() +
  labs(title = "Paraeto-K values", subtitle = "Values > 1 indicate highly influential data")
```



Boy! Some hurricanes really do drive our model to a big extent!

//

Inciso **2.b**

Los conteo casi siempre están demasiado dispersos en relación con una distribución Poisson. Así que ajusta un Modelo gamma-Poisson (también conocido como binomial-negativo) para predecir muertes utilizando la feminidad. Demuestra que el modelo con sobre-dispersión ya no muestra un resultado positivo tan preciso entre feminidad y muerte, con un intervalo de 89% que se superpone cero. ¿Puedes explicar por qué la asociación disminuyó?

Respuesta

// Question: Counts are nearly always over-dispersed relative to Poisson. So fit a gamma-Poisson (aka negative-binomial) model to predict deaths using femininity. Show that the over-dispersed model no longer shows as precise a positive association between femininity and deaths, with an 89% interval that overlaps zero. Can you explain why the association diminished in strength?

Answer: To start this off, I load the library and data again, so much of the exercise and my solutions can stand by itself:

```
d <- Hurricanes # load data on object called d
d$fem_std <- (d$femininity - mean(d$femininity)) / sd(d$femininity) # standardised femininity
dat <- list(D = d$deaths, F = d$fem_std)
```

Again, with the data prepared, we fit our model - the same model as before just with a different outcome distribution:

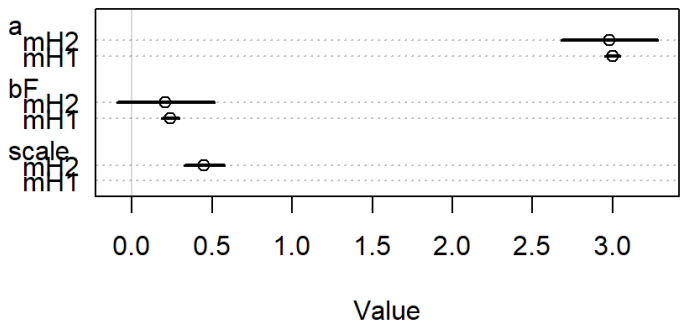
```
mH2 <- ulam(
  alist(
    D ~ dgampois(lambda, scale),
    log(lambda) <- a + bF * F,
    a ~ dnorm(1, 1),
    bF ~ dnorm(0, 1),
    scale ~ dexp(1) # strictly positive hence why exponential prior
  ),
  data = dat, chains = 4, cores = 4, log_lik = TRUE
)
precis(mH2)
```

```
##            mean        sd        5.5%     94.5%    n_eff     Rhat4
## a     2.9754678 0.1526423  2.73288830 3.2195337 1837.278 1.0018939
## bF    0.2085543 0.1536824 -0.04050055 0.4469646 1794.019 0.9990891
## scale 0.4514191 0.0629423  0.35466194 0.5547061 1891.992 0.9986898
```

Cool. Our previously identified positive relationship between standardised femininity of hurricane name and death toll is still there albeit slightly diminished in magnitude. However, the credible interval around it has widened considerably and overlaps zero now.

Let's compare the estimates of our models side by side:

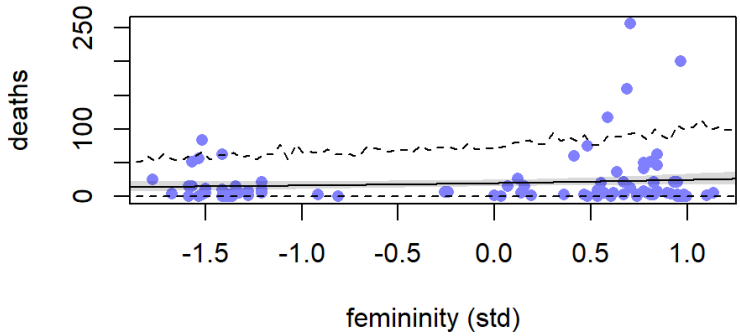
```
plot(coeftab(mH1, mH2))
```



These shows quite clearly how our new model is much more uncertain of the parameters.

So what about the predictions of this new model? I plot them the exact same way as previously:

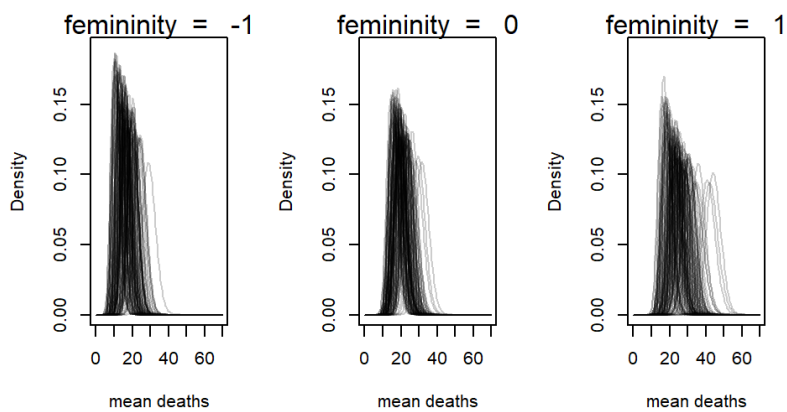
```
# plot raw data
plot(dat$F, dat$D,
  pch = 16, lwd = 2,
  col = rangi2, xlab = "femininity (std)", ylab = "deaths"
)
# compute model-based trend
pred_dat <- list(F = seq(from = -2, to = 2, length.out = 1e2))
lambda <- link(mH2, data = pred_dat)
lambda.mu <- apply(lambda, 2, mean)
lambda.PI <- apply(lambda, 2, PI)
# superimpose trend
lines(pred_dat$F, lambda.mu)
shade(lambda.PI, pred_dat$F)
# compute sampling distribution
deaths_sim <- sim(mH2, data = pred_dat)
deaths_sim.PI <- apply(deaths_sim, 2, PI)
# superimpose sampling interval as dashed lines
lines(pred_dat$F, deaths_sim.PI[1, ], lty = 2)
lines(pred_dat$F, deaths_sim.PI[2, ], lty = 2)
```



What's there left to say other than: "Look at that increased uncertainty of our model" at this point? Well, we can talk about the accuracy of our predictions. They still blow. The uncertainty of our model is nice and all, but with a predictive accuracy like this why would we trust the model?

For now, let's turn to the conceptual part of this exercise: "Why has the association diminished with the new model?" The question comes down to understanding what the gamma distribution does to our model. The gamma distribution allows for a death rate to be calculated for each outcome individually rather than one overall death rate for all hurricanes. These individual rates are sampled from a common distribution which is a function of the femininity of hurricane names. As a matter of fact, we can plot this:

```
post <- extract.samples(mH2)
par(mfrow = c(1, 3))
for (fem in -1:1) {
  for (i in 1:1e2) {
    curve(dgamma2(
      x, # where to calculate density
      exp(post$a[i] + post$bF[i] * fem), # linear model with inverse link applied
      post$scale[i] # scale for gamma
    ),
    from = 0, to = 70, xlab = "mean deaths", ylab = "Density",
    ylim = c(0, 0.19), col = col.alpha("black", 0.2),
    add = ifelse(i == 1, FALSE, TRUE)
    )
  }
  mtext(concat("femininity  =   ", fem))
}
```

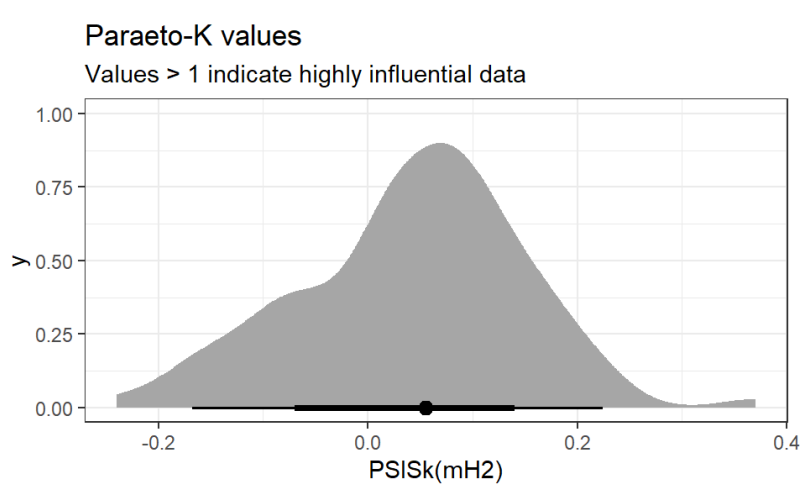


These are the gamma distributions samples from the posterior distribution of death rates when assuming same femininity of name for all of them at three different levels of femininity. Yes, a distribution sampled from another distribution. The above plots simply show the uncertainty of which gamma distribution to settle on.

Since our model and gamma distributions are informed by a, bF, and the scale for the gamma distribution at the same time many combinations of a and bF are consistent with the data which results in a wider posterior distribution.

Finally, let's look at Paraeto-K values and potentially influential data again:

```
ggplot(as.data.frame(PSISk(mH2)), aes(x = PSISk(mH2))) +
  stat_halfeye() +
  theme_bw() +
  labs(title = "Paraeto-K values", subtitle = "Values > 1 indicate highly influential data")
```



MUCH BETTER than before!

//

Inciso **2.c**

En los datos, hay dos medidas del potencial de letalidad de un huracán: `damage_norm` y `min_pressure`. Consulta `?Hurricanes`. Hace algo de sentido imaginar que la feminidad de un nombre importa más cuando el huracán es en sí mismo mortal. Esto implica una interacción entre la feminidad y posiblemente una o las dos `damage_norm` y `min_pressure`. Ajusta una serie de modelos evaluando estas interacciones. Interpreta y compara los modelos. Al interpretar las estimaciones, te puede ayudar a generar predicciones que contrasten los huracanes con nombres masculinos y femeninos. ¿Son probables los coeficientes?

Respuesta

// Question: In order to infer a strong association between deaths and femininity, it's necessary to include an interaction effect. In the data, there are two measures of a hurricane's potential to cause death: damage_norm and min_pressure. Consult ?Hurricanes for their meanings. It makes some sense to imagine that femininity of a name matters more when the hurricane is itself deadly. This implies an interaction between femininity and either or both of damage_norm and min_pressure. Fit a series of models evaluating these interactions. Interpret and compare the models. In interpreting the estimates, it may help to generate counterfactual predictions contrasting hurricanes with masculine and feminine names. Are the effect sizes plausible?

Answer: To start this off, I load the library and data again, so much of the exercise and my solutions can stand by itself:

```
d <- Hurricanes # load data on object called d
d$fem_std <- (d$femininity - mean(d$femininity)) / sd(d$femininity) # standardised femininity
dat <- list(D = d$deaths, F = d$fem_std)
dat$P <- standardize(d$min_pressure)
dat$S <- standardize(d$damage_norm)
```

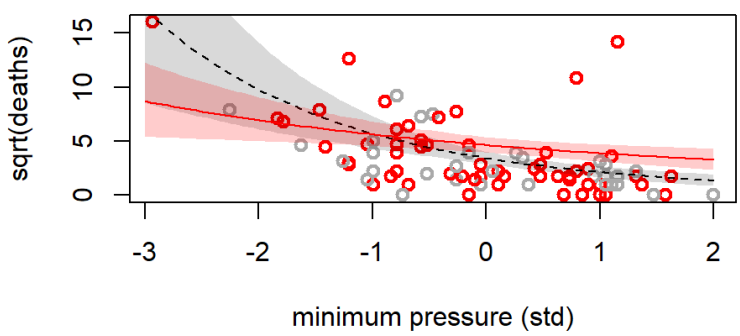
The data is ready and I step into my model fitting procedure. Here, I start with a basic model which builds on the previous gamma-Poisson model by adding an interaction between femininity and min_pressure:

```
mH3a <- ulam(
  alist(
    D ~ dgampois(lambda, scale),
    log(lambda) <- a + bF * F + bP * P + bFP * F * P,
    a ~ dnorm(1, 1),
    c(bF, bP, bFP) ~ dnorm(0, 1),
    scale ~ dexp(1)
  ),
  data = dat, cores = 4, chains = 4, log_lik = TRUE
)
precis(mH3a)
```

```
##             mean         sd        5.5%      94.5%    n_eff     Rhat4
## a      2.7516572 0.14037814  2.53116328  2.9724634 2276.683 0.9986039
## bFP    0.3028091 0.14870602  0.07557365  0.5423556 2276.941 0.9990635
## bP    -0.6763636 0.14029923 -0.90989516 -0.4583701 2145.141 1.0019806
## bF     0.2977328 0.14304800  0.06629476  0.5202383 2552.804 0.9995844
## scale  0.5509295 0.07856721  0.43452667  0.6849383 2788.009 0.9988867
```

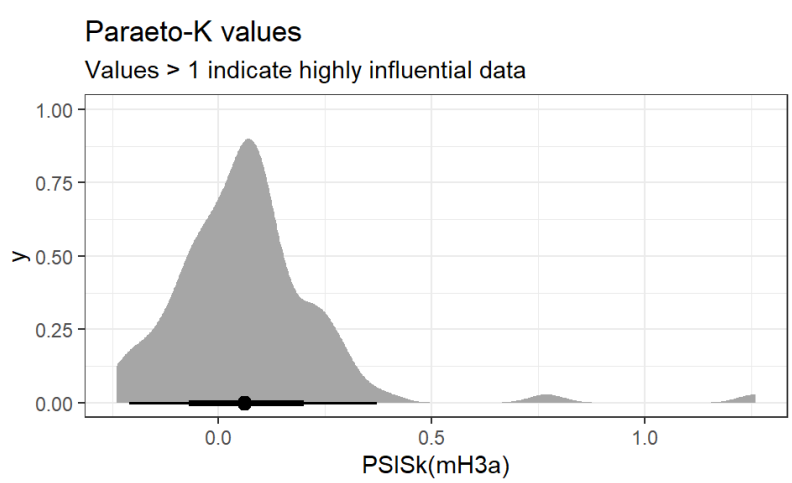
As minimum pressure gets lower, a storm grows stronger (I was confused by that myself when answering these exercises). Quite obviously, the lower the pressure in a storm, the more severe the storm, and the more people die which is reflected by the negative value in bP. bF is still estimated to be positive. This time, the interval doesn't even overlap zero. Meanwhile, the interaction effect bFP is positive. I find it hard to interpret this so I'd rather plot some predictions against real data:

```
P_seq <- seq(from = -3, to = 2, length.out = 1e2) # pressure sequence
# 'masculine' storms
d_pred <- data.frame(F = -1, P = P_seq)
lambda_m <- link(mH3a, data = d_pred)
lambda_m.mu <- apply(lambda_m, 2, mean)
lambda_m.PI <- apply(lambda_m, 2, PI)
# 'feminine' storms
d_pred <- data.frame(F = 1, P = P_seq)
lambda_f <- link(mH3a, data = d_pred)
lambda_f.mu <- apply(lambda_f, 2, mean)
lambda_f.PI <- apply(lambda_f, 2, PI)
# Plotting, sqrt() to make differences easier to spot, can't use log because there are storm with zero deaths
plot(dat$P, sqrt(dat$D),
  pch = 1, lwd = 2, col = ifelse(dat$F > 0, "red", "dark gray"),
  xlab = "minimum pressure (std)", ylab = "sqrt(deaths)"
)
lines(P_seq, sqrt(lambda_m.mu), lty = 2)
shade(sqrt(lambda_m.PI), P_seq)
lines(P_seq, sqrt(lambda_f.mu), lty = 1, col = "red")
shade(sqrt(lambda_f.PI), P_seq, col = col.alpha("red", 0.2))
```



Our model expects masculine (grey) storms to be less deadly, on average, than feminine (red) ones. As pressure drops (toward the rightward side of the plot above), these differences become smaller and smaller. Quite evidently, some of these storms are influencing what our model predicts much more so than others:

```
ggplot(as.data.frame(PSISk(mH3a)), aes(x = PSISk(mH3a))) +
  stat_halfeye() +
  theme_bw() +
  labs(title = "Paraeto-K values", subtitle = "Values > 1 indicate highly influential data")
```



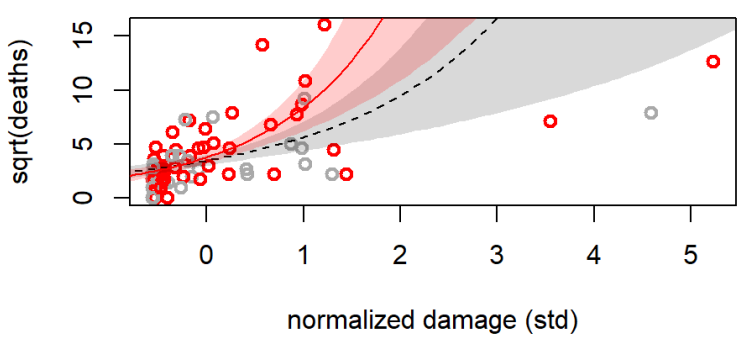
Let's turn to the second variable we may want to add damage_norm - the damage caused by each storm:

```
mH3b <- ulam(
  alist(
    D ~ dgampois(lambda, scale),
    log(lambda) <- a + bF * F + bS * S + bFS * F * S,
    a ~ dnorm(1, 1),
    c(bF, bS, bFS) ~ dnorm(0, 1),
    scale ~ dexp(1)
  ),
  data = dat, chains = 4, cores = 4, log_lik = TRUE
)
precis(mH3b)
```

```
##             mean        sd        5.5%     94.5%    n_eff     Rhat4
## a     2.56757213 0.1343593  2.35618536 2.7998135 1796.484 0.9990252
## bFS   0.30937924 0.1991304 -0.03022903 0.6246079 1795.019 1.0005439
## bS    1.23697580 0.2098261  0.91244715 1.5839509 1861.683 1.0003329
## bF    0.08411631 0.1248017 -0.11513477 0.2823281 2383.325 0.9998757
## scale 0.68210221 0.1006315  0.53167917 0.8524670 2162.298 0.9987214
```

That just eradicated the effect of femininity of hurricane name (bF)! The newly added interaction parameter bFS is incredibly strong and positive. Again, let's visualise this:

```
S_seq <- seq(from = -1, to = 5.5, length.out = 1e2) # damage sequence
# 'masculine' storms
d_pred <- data.frame(F = -1, S = S_seq)
lambda_m <- link(mH3b, data = d_pred)
lambda_m.mu <- apply(lambda_m, 2, mean)
lambda_m.PI <- apply(lambda_m, 2, PI)
# 'feminine' storms
d_pred <- data.frame(F = 1, S = S_seq)
lambda_f <- link(mH3b, data = d_pred)
lambda_f.mu <- apply(lambda_f, 2, mean)
lambda_f.PI <- apply(lambda_f, 2, PI)
# plot
plot(dat$S, sqrt(dat$D),
  pch = 1, lwd = 2, col = ifelse(dat$F > 0, "red", "dark gray"),
  xlab = "normalized damage (std)", ylab = "sqrt(deaths)"
)
lines(S_seq, sqrt(lambda_m.mu), lty = 2)
shade(sqrt(lambda_m.PI), S_seq)
lines(S_seq, sqrt(lambda_f.mu), lty = 1, col = "red")
shade(sqrt(lambda_f.PI), S_seq, col = col.alpha("red", 0.2))
```



We can clearly see how our model makes less of a distinction between masculine and feminine hurricanes overall at this point. Damage norm scales multiplicatively. The distances grow fast as we approach the rightward side of the plot. This is difficult for the model to account for. Hence why the model is underwhelming.

So why is the interaction effect so strong? Probably because of those 3-4 highly influential feminine storms at the upper-righthand corner of our plot above which implies that feminine storms are especially deadly when they are damaging to begin with. Personally, I don't trust this association and would argue that there is no logical reason for it and most likely an artefact of the limited data availability.

//

Inciso **2.d**

En el artículo original sobre huracanes, se utilizó directamente el daño por tormenta (`damage_norm`). Esta suposición implica que la mortalidad aumenta exponencialmente con aumento lineal en la fuerza de la tormenta. Esto debido a que en regresión Poisson usamos un enlace logarítmico. Entonces, vale la pena explorar una hipótesis alternativa: que el logaritmo de la fuerza de la tormenta es lo que importa. Explora esto usando el logaritmo de `damage_norm` como un predictor. Usando la mejor estructura de modelo del inciso anterior, compara un modelo que usa `log(damage_norm)` a un modelo que usa `damage_norm` directamente. Compara la capacidad predictiva, así como sus predicciones implícitas. ¿Qué es lo que concluyes?

Respuesta

// Question: In the original hurricanes paper, storm damage (damage_norm) was used directly. This assumption implies that mortality increases exponentially with a linear increase in storm strength, because a Poisson regression uses a log link. So it's worth exploring an alternative hypothesis: that the logarithm of storm strength is what matters. Explore this by using the logarithm of damage_norm as a predictor. Using the best model structure from the previous problem, compare a model that uses log(damage_norm) to a model that uses damage_norm directly. Compare their DIC/WAIC values as well as their implied predictions. What do you conclude?

Answer: To start this off, I load the library and data again, so much of the exercise and my solutions can stand by itself:

```
d <- Hurricanes # load data on object called d
d$fem_std <- (d$femininity - mean(d$femininity)) / sd(d$femininity) # standardised femininity
dat <- list(D = d$deaths, F = d$fem_std)
dat$S2 <- standardize(log(d$damage_norm))
```

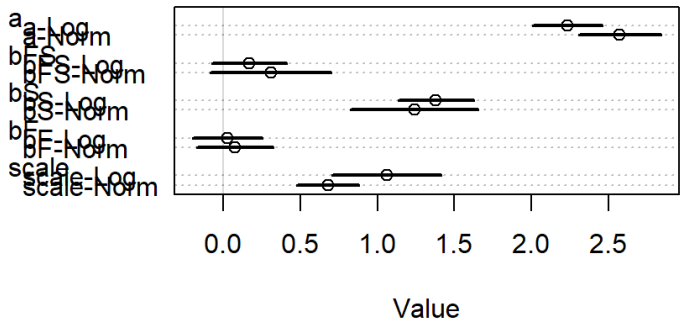
Let's fit the model as before and compare it to the previously identified best model:

```
mH4 <- ulam(
  alist(
    D ~ dgampois(lambda, scale),
    log(lambda) <- a + bF * F + bS * S2 + bFS * F * S2,
    a ~ dnorm(1, 1),
    c(bF, bS, bFS) ~ dnorm(0, 1),
    scale ~ dexp(1)
  ),
  data = dat, chains = 4, cores = 4, log_lik = TRUE
)
compare(mH3b, mH4, func = PSIS)
```

```
##          PSIS       SE    dPSIS      dSE    pPSIS       weight
## mH4  630.6630 31.20196  0.00000       NA 5.342250 1.000000e+00
## mH3b 670.5647 34.06285 39.90172 13.46122 6.747284 2.164972e-09
```

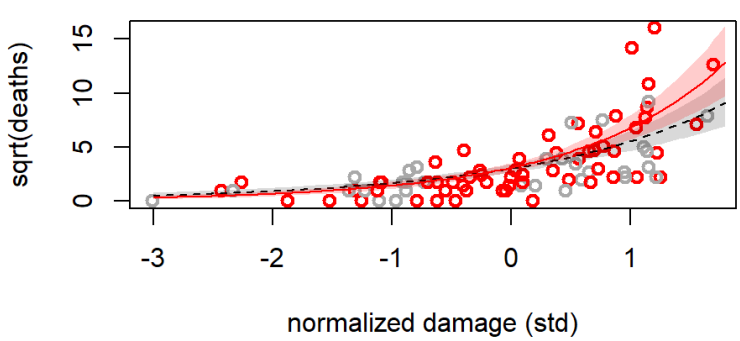
Model mH4 clearly outperforms the earlier (non-logarithmic) model mH3b. How do the parameter estimates look in comparison?

```
plot(coeftab(mH3b, mH4),
  labels = paste(rep(rownames(coeftab(mH3b, mH4)@coefs), each = 2),
    rep(c("Norm", "Log"), nrow(coeftab(mH3b, mH4)@coefs) * 2),
    sep = "-"
  )
)
```



With the log-transformed input, bFS has increased in magnitude. What do the resulting predictions look like?

```
S2_seq <- seq(from = -3, to = 1.8, length.out = 1e2)
# 'masculine' storms
d_pred <- data.frame(F = -1, S2 = S2_seq)
lambda_m <- link(mH4, data = d_pred)
lambda_m.mu <- apply(lambda_m, 2, mean)
lambda_m.PI <- apply(lambda_m, 2, PI)
# 'feminine' storms
d_pred <- data.frame(F = 1, S2 = S2_seq)
lambda_f <- link(mH4, data = d_pred)
lambda_f.mu <- apply(lambda_f, 2, mean)
lambda_f.PI <- apply(lambda_f, 2, PI)
# plot
plot(dat$S2, sqrt(dat$D),
  pch = 1, lwd = 2, col = ifelse(dat$F > 0, "red", "dark gray"),
  xlab = "normalized damage (std)", ylab = "sqrt(deaths)"
)
lines(S2_seq, sqrt(lambda_m.mu), lty = 2)
shade(sqrt(lambda_m.PI), S2_seq)
lines(S2_seq, sqrt(lambda_f.mu), lty = 1, col = "red")
shade(sqrt(lambda_f.PI), S2_seq, col = col.alpha("red", 0.2))
```



Now this model fits the data much better! Still not perfect, but much better.

//

3. Inferencia Causal: experimentos aleatorizados

Distribuciones muestrales bajo aleatorización: Utilice la covariable y el potencial de salida (*potential outcome*) de los datos en la tabla 18.1 del libro *Regresion and Other Stories*. Abajo viene una versión simplificada (aunque hacen falta un par más, incorporalas):

```
omega <- tibble(female = factor(rep(rep(c(1,0), each = 2), 2)),
       age    = rep(c(4,5,6,7), each = 2) * 10,
       treatment = factor(rep(c(0,1), each = 4)),
       outcome = rep(c(140, 150, 155, 160), each = 2))

omega %>%
  head() %>%
  kbl(digits=2, format.args = list(big.mark = ","))  %>%
  kable_styling(bootstrap_options = "striped", full_width = F)
```

female	age	treatment	outcome
1	40	0	140
1	40	0	140
0	50	0	150
0	50	0	150
1	60	1	155
1	60	1	155

como punto de partida para considerar distribuciones de aleatorización de cuatro diseños diferentes mediante la creación de simulaciones en `R`.

Comenta sobre el sesgo relativo y la eficiencia para cada uno de los siguientes diseños: • Diseño completamente aleatorizado, • Diseño aleatorio usando bloques por los cuatro participantes mayores frente a los cuatro más jóvenes, • Diseño de pares combinados,

utilizando cada una de las siguientes estimaciones: • Diferencia de medias, • Regresión del indicador de tratamiento y la edad, • Regresión del indicador de tratamiento, edad y sexo, • Regresión del indicador de tratamiento, edad, sexo e interacción tratamiento × × sexo.

4. Inferencia Causal y Modelos de Regresión: vacas

Aleatorización desordenada: los datos de `vacas.txt` contiene datos de un experimento que se llevó a cabo con 50 vacas para estimar el efecto de un complemento alimenticio en 6 resultados relacionados con la cantidad de grasa láctea producida por cada vaca. Se consideraron cuatro dietas (tratamientos), correspondientes a diferentes niveles del complemento, y se registraron tres variables antes de la asignación del tratamiento: número de lactancia (temporadas de lactancia), edad y peso inicial de la vaca.

Las vacas se asignaron inicialmente a tratamientos completamente al azar, y después se revisaron las distribuciones de las tres covariables para verificar el equilibrio a lo largo de los grupos de tratamiento. Se probaron varias aleatorizaciones, y la que produjo el “mejor” equilibrio con respecto a las tres covariables fue la que se escogió. El tratamiento depende sólo de las covariables completamente observadas y no de las no registradas como el aspecto físico de las vacas o los momentos en los que vacas entraron en el estudio. Esto es porque las decisiones de volver a aleatorizar no son explicados. Consideraremos diferentes estimaciones del efecto del complemento en la grasa láctea media diaria producida.

Inciso **4.a**

Considera la regresión massimple de la grasa láctea media diaria con el nivel de complemento. Calcula el efecto del tratamiento estimado (coeficiente de regresión) y el error estándar, y explica por qué este no es un análisis completamente apropiado dada la aleatorización utilizada.

Inciso **4.b**

Agrega más predictores al modelo. Explica el razonamiento para la elección de covariables en el modelo. Compare el efecto estimado del tratamiento con el resultado de (a).

Inciso **4.c**

Repite (**4. b**), esta vez considerando el nivel del complemento como un predictor categórico con cuatro niveles. Haga una gráfica que muestre la estimación (y el error estándar) del efecto del tratamiento en cada nivel, y también mostrando la inferencia del modelo ajustado en (**4. b**).

```
Vacas <- read_delim("vacas.txt", delim = " ") %>%
  type.convert() %>%
  as.data.frame()

# CHECAR SI SE PUEDE EVITAR CREAR ATRIBUTOS Y/O SI ES NECESARIO ELMINARLOS
attr(Vacas, 'problems') <- NULL
attr(Vacas, 'spec') <- NULL
Vacas<-as.tibble(Vacas)

Vacas %>%
  head() %>%
  kbl(digits=2, format.args = list(big.mark = ","))  %>%
  kable_styling(bootstrap_options = "striped", full_width = F)
```

level	lactation	age	initial.weight	dry	milk	fat	solids	final.weight	protein
0	3	49	1,360	15.43	45.55	3.88	8.96	1,442	3.67
0	3	47	1,498	18.80	66.22	3.40	8.44	1,565	3.03
0	2	36	1,265	17.95	63.03	3.44	8.70	1,315	3.40
0	2	33	1,190	18.27	68.42	3.42	8.30	1,285	3.37
0	2	31	1,145	17.25	59.67	3.01	9.04	1,182	3.61
0	1	22	1,035	13.05	44.05	2.97	8.60	1,043	3.03

```
  # kable_material(c("striped"))
```

```
descr(Vacas) %>%
  kbl(digits=2, format.args = list(big.mark = ",")) %>%
  kable_styling(bootstrap_options = "striped", full_width = F)
```

	age	dry	fat	final.weight	initial.weight	lactation	level	milk	protein	solids
Mean	42.16	16.43	3.58	1,244.40	1,258.06	2.38	0.15	59.54	3.33	8.69
Std.Dev	18.59	2.42	0.48	166.99	181.21	1.32	0.11	9.36	0.23	0.28
Min	21.00	11.42	2.65	968.00	900.00	1.00	0.00	40.24	2.86	7.81
Q1	26.00	14.54	3.24	1,120.00	1,110.00	1.00	0.10	53.10	3.17	8.46
Median	37.00	16.69	3.46	1,234.00	1,266.50	2.00	0.15	59.52	3.31	8.74
Q3	49.00	18.27	3.91	1,353.00	1,369.00	3.00	0.20	66.67	3.47	8.93
Max	95.00	20.46	4.96	1,593.00	1,656.00	6.00	0.30	76.60	3.80	9.19
MAD	17.79	2.97	0.48	172.72	206.08	1.48	0.07	10.24	0.21	0.30
IQR	22.75	3.66	0.64	221.75	250.25	2.00	0.10	13.55	0.29	0.46
CV	0.44	0.15	0.14	0.13	0.14	0.56	0.74	0.16	0.07	0.03
Skewness	1.00	-0.15	0.59	0.23	0.11	0.90	0.00	-0.11	0.20	-0.62
SE.Skewness	0.34	0.34	0.34	0.34	0.34	0.34	0.34	0.34	0.34	0.34
Kurtosis	0.36	-1.06	0.03	-0.90	-0.65	0.39	-1.38	-1.05	-0.48	0.29
N.Valid	50.00	50.00	50.00	50.00	50.00	50.00	50.00	50.00	50.00	50.00
Pct.Valid	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00

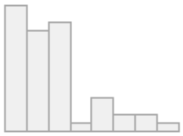
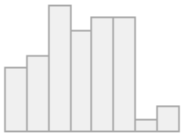
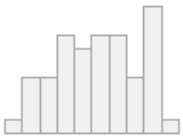
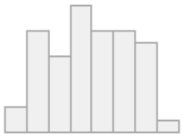
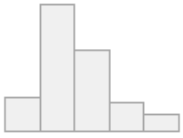
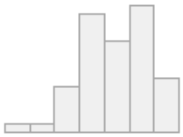
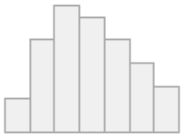
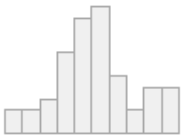
```
# Si se quisiera visualizar el summary corriendo solo este chunck hacerlo con:
# print(dfSummary(Vacas, plain.ascii=FALSE, style="grid", valid.col=FALSE), method="render")

# Con lo sisuientes parámetros (incluido la opción del chunk "results='asis'", puede
# visualizarse el summary cuando se genera el html)
dfSummary(Vacas, plain.ascii = FALSE, style = "grid", graph.magnif = 0.75, valid.col = TRUE, tmp.img.dir = "/tmp")
```

Data Frame Summary

Vacas

Dimensions: 50 x 10
Duplicates: 0

No	Variable	Stats / Values	Freqs (% of Valid)	Graph	Valid	Missing

No	Variable	Stats / Values	Freqs (% of Valid)	Graph	Valid	Missing
1	level [numeric]	Mean (sd) : 0.1 (0.1) min < med < max: 0 < 0.2 < 0.3 IQR (CV) : 0.1 (0.7)	0.00 : 12 (24.0%) 0.10 : 13 (26.0%) 0.20 : 13 (26.0%) 0.30 : 12 (24.0%)		50 (100.0%)	0 (0.0%)
2	lactation [integer]	Mean (sd) : 2.4 (1.3) min < med < max: 1 < 2 < 6 IQR (CV) : 2 (0.6)	1 : 16 (32.0%) 2 : 12 (24.0%) 3 : 15 (30.0%) 4 : 3 (6.0%) 5 : 2 (4.0%) 6 : 2 (4.0%)		50 (100.0%)	0 (0.0%)
3	age [integer]	Mean (sd) : 42.2 (18.6) min < med < max: 21 < 37 < 95 IQR (CV) : 22.8 (0.4)	32 distinct values		50 (100.0%)	0 (0.0%)
4	initial.weight [integer]	Mean (sd) : 1258.1 (181.2) min < med < max: 900 < 1266.5 < 1656 IQR (CV) : 250.2 (0.1)	46 distinct values		50 (100.0%)	0 (0.0%)
5	dry [numeric]	Mean (sd) : 16.4 (2.4) min < med < max: 11.4 < 16.7 < 20.5 IQR (CV) : 3.7 (0.1)	50 distinct values		50 (100.0%)	0 (0.0%)
6	milk [numeric]	Mean (sd) : 59.5 (9.4) min < med < max: 40.2 < 59.5 < 76.6 IQR (CV) : 13.5 (0.2)	50 distinct values		50 (100.0%)	0 (0.0%)
7	fat [numeric]	Mean (sd) : 3.6 (0.5) min < med < max: 2.6 < 3.5 < 5 IQR (CV) : 0.6 (0.1)	43 distinct values		50 (100.0%)	0 (0.0%)
8	solids [numeric]	Mean (sd) : 8.7 (0.3) min < med < max: 7.8 < 8.7 < 9.2 IQR (CV) : 0.5 (0)	35 distinct values		50 (100.0%)	0 (0.0%)
9	final.weight [integer]	Mean (sd) : 1244.4 (167) min < med < max: 968 < 1234 < 1593 IQR (CV) : 221.8 (0.1)	45 distinct values		50 (100.0%)	0 (0.0%)
10	protein [numeric]	Mean (sd) : 3.3 (0.2) min < med < max: 2.9 < 3.3 < 3.8 IQR (CV) : 0.3 (0.1)	35 distinct values		50 (100.0%)	0 (0.0%)