Resumen lectura 4

Otros modelos lineales generalizados

Podremos aplicar el principio de regresión logística (link function) `y = a + bx` y extenderlo a través de transformaciones no lineales y modelos de probabilidad que permiten predecir datos dicretos. Se revisará regresión poisson, regresión binomial negativa, modelos logístico-binomial y probit, regresión logística ordenada, regresión robusta, y otras extensiones.

Definiciones

Modelos lineales generalizados es un marco de trabajo que incluye regresión logística y otros casos especiales. En regresión lineal se predice una variable continua, donde "y" es una predicción de la forma $XB = B_0 + X_1B_1 + X_2B_2 + X_kB_k$. Regresión logística predice $P(Y = 1)$. Un modelo lineal generalizado involucra:

1. vector de resultados $y = (y_1, y_2, ..., y_k)$
2. Matriz de prediccionesX y vector de coeficientes B, formando el vector predictor XB.
3. Función liga "g", que produce el vector transformado de datos $g^{-1}(XB)$ que es usado para modelar los datos.
4. Una distribución $p(y|\hat{y})$.
5. Otros posibles parámetros, tales como varianza, sobredeispersión, puntos de corte, etc. incolucrados.

Regresión Poisson y binomial negativa

En regresiones del conteo de datos, cada unidad y corresponde a un ajuste en el cual y_i eventos son observados. Por ejemplo y podría indexar intersecciones de calles en una ciudad. y_i podría ser la cantidad de accidentes de tráfico en la intersección y en un año dado.

Tal como con regresión logística y lineal, la variación en y puede ser explicada con predictores lineales X. En el ejemplo de accidentes de tráfico estos predictores pueden incluir un término constante coma una medida de velocidad promedio del tráfico cerca de la intersección, y un indicador de si la intersección de las calles tenía una señal de tráfico.

Modelo Poisson

El modelo más simple de regresión para conteos de datos es,

$$y_i \sim Poisson(e^{X_iB})$$

tal que el predictor lineal X_iB es el logaritmo del valor esperado de medición y_i. Bajo el modelo poisson, $sd(y_i) = \sqrt{E(y_i)}$; tal que si el modelo describe los datos de modo preciso, también tendremos idea de cuanta variación podríamos esperar de la curva ajustada. Ejemplo:

```r
n <- 50
x <- runif(n, -2, 2)
a <- 1
b <- 2
linpred <- a + b*x
y <- rpois(n, exp(linpred))
fake <- data.frame(x=x, y=y)
```

Luego, se ajusta el modelo a los datos simulados

```r
library(rstanarm)
```

```
## Loading required package: Rcpp

## This is rstanarm version 2.21.1

## - See https://mc-stan.org/rstanarm/articles/priors for changes to default priors!

## - Default priors may change, so it's safest to specify priors, even if equivalent to the defaults.

## - For execution on a local, multicore CPU with excess RAM we recommend calling

##   options(mc.cores = parallel::detectCores())
```

```r
fit_fake <- stan_glm(y ~ x, family=poisson(link="log"), data=fake)
```

```
##
## SAMPLING FOR MODEL 'count' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 5.9e-05 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0.59 seconds.
## Chain 1: Adjust your expectations accordingly!
## Chain 1:
## Chain 1:
## Chain 1: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 1: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 1: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 1: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 1: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 1: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 1: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 1: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 1: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 1: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 1: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 1: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 0.053448 seconds (Warm-up)
## Chain 1:                0.058812 seconds (Sampling)
## Chain 1:                0.11226 seconds (Total)
```

```
## Chain 1:
##
## SAMPLING FOR MODEL 'count' NOW (CHAIN 2).
## Chain 2:
## Chain 2: Gradient evaluation took 1e-05 seconds
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0.1 seconds.
## Chain 2: Adjust your expectations accordingly!
## Chain 2:
## Chain 2:
## Chain 2: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 2: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 2: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 2: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 2: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 2: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 2: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 2: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 2: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 2: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 2: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 2: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 2:
## Chain 2:  Elapsed Time: 0.060488 seconds (Warm-up)
## Chain 2:                0.058933 seconds (Sampling)
## Chain 2:                0.119421 seconds (Total)
## Chain 2:
##
## SAMPLING FOR MODEL 'count' NOW (CHAIN 3).
## Chain 3:
## Chain 3: Gradient evaluation took 1e-05 seconds
## Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0.1 seconds.
## Chain 3: Adjust your expectations accordingly!
## Chain 3:
## Chain 3:
## Chain 3: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 3: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 3: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 3: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 3: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 3: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 3: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 3: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 3: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 3: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 3: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 3: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 3:
## Chain 3:  Elapsed Time: 0.057082 seconds (Warm-up)
## Chain 3:                0.058528 seconds (Sampling)
## Chain 3:                0.11561 seconds (Total)
## Chain 3:
##
## SAMPLING FOR MODEL 'count' NOW (CHAIN 4).
## Chain 4:
```

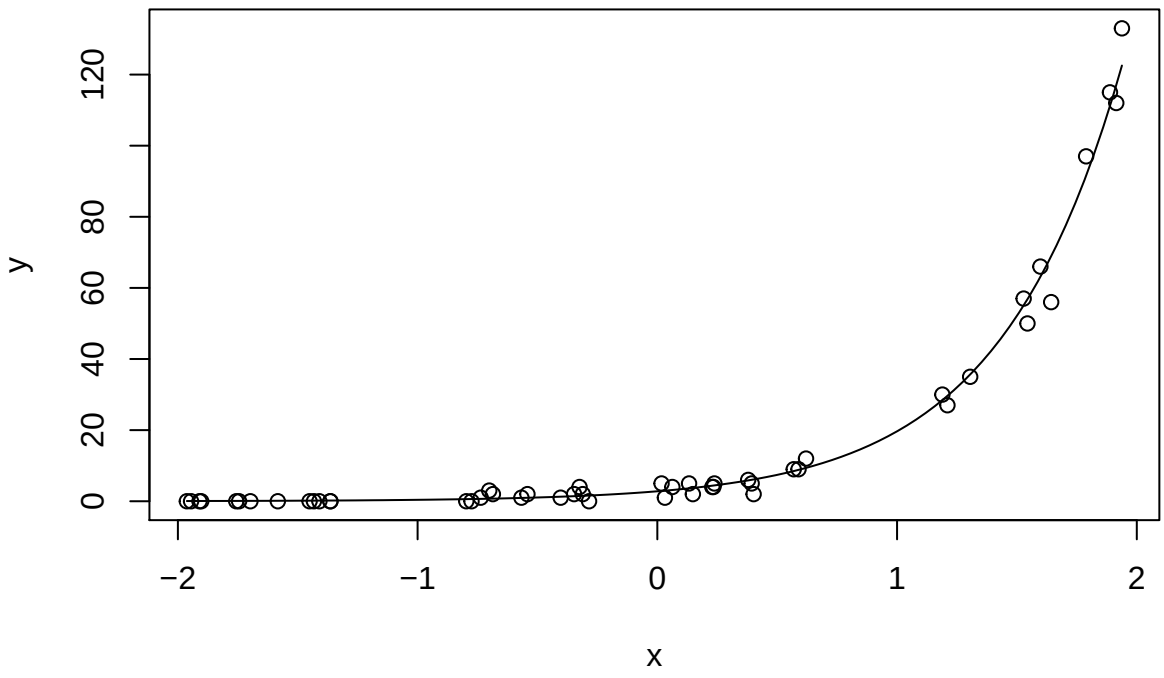
```
## Chain 4: Gradient evaluation took 1.3e-05 seconds
## Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 0.13 seconds.
## Chain 4: Adjust your expectations accordingly!
## Chain 4:
## Chain 4:
## Chain 4: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 4: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 4: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 4: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 4: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 4: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 4: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 4: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 4: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 4: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 4: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 4: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 4:
## Chain 4:  Elapsed Time: 0.053469 seconds (Warm-up)
## Chain 4:                0.058517 seconds (Sampling)
## Chain 4:                0.111986 seconds (Total)
## Chain 4:
```

```
print(fit_fake)
```

```
## stan_glm
##  family:       poisson [log]
##  formula:      y ~ x
##  observations: 50
##  predictors:   2
## ------
##             Median MAD_SD
## (Intercept) 1.0    0.1
## x           2.0    0.1
##
## ------
## * For help interpreting the printed output see ?print.stanreg
## * For info on the priors used see ?prior_summary.stanreg
```

y se grafican los datos a través de la curva ajustada

```
plot(x, y)
curve(exp(coef(fit_fake)[1] + coef(fit_fake)[2]*x), add=TRUE)
```




```
knitr::include_graphics("pic1.png")
```

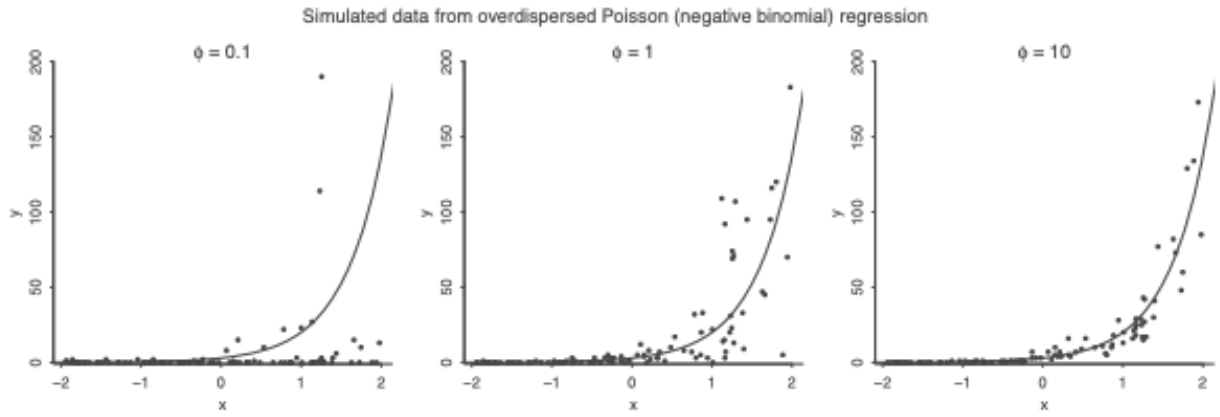


Figure 15.2 *Simulated data from the model, $y_i \sim$ negative binomial(e^{a+bx_i}, ϕ), for three different values of the reciprocal dispersion parameter ϕ. For each, we also plot the fitted curve, $y = e^{\hat{a}+\hat{b}x}$. As can be seen from this example, the lower the parameter ϕ, the greater the vertical deviations of the points from the line.*

A diferencia de la distribución normal, no hay necesidad de ajustar un parámetro sigma.

Sobredispersión y subdispersión

Estos conceptos hacer referencia a datos que muestran más o menos variación de la esperada en un modelo de probabilidad ajustada. Continuando con el ejemplo de accidentes, supóngase que miras los datos de accidentes de tráfico y se observa mucho más variación de la esperada por parte del modelo Poisson: Eso es la sobredispersión.

Los conteos de datos en donde la variación del modelo es menor a la raíz cuadrada de los valores predichos, podría ser un indicador de subdisepersión.

Modelo Binomial negativo para sobredispersión

Para generalizar el modelo de Poisson para permitir la sobredispersión, usamos el modelo binomial negativo, el cual incluye un parámetro adicional de "dispersión recíproca" ϕ tal que $sd(y|x) = \sqrt{E(y|x) + E(y|x)^2/\phi}$. En esta reparametrización, el parámetro ϕ se restringe ser positivo, con valores bajos correspondientes a mayor sobredispresión, y en el límite $\phi \to \infty$ representa el modelo Poisson (es decir, cero sobredispersión).

Para tener idea de cómo luce este modelo, simulamos 3 conjuntos de datos y los mismos valores del predictor x, pero reemplazando el poisson con la distribución binomial con parámetros ϕ a 0.1, 1 o 10.

```
phi_grid <- c(0.1, 1, 10)
K <- length(phi_grid)
y_nb <- as.list(rep(NA, K))
fake_nb <- as.list(rep(NA, K))
fit_nb <- as.list(rep(NA, K))
```

luego, para cada uno de los valores de ϕ, se simulan datos y se ajusta el modelos de regresión binomial negativa:

```
library("MASS")

for (k in 1:K){
y_nb[[k]] <- rnegbin(n, exp(linpred), phi_grid[k])
fake_nb[[k]] <- data.frame(x=x, y=y_nb[[k]])
fit_nb[[k]] <- stan_glm(y ~ x, family=neg_binomial_2(link="log"), data=fake)
print(fit_nb[[k]])
}
```

```
##
## SAMPLING FOR MODEL 'count' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 3.3e-05 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0.33 seconds.
## Chain 1: Adjust your expectations accordingly!
## Chain 1:
## Chain 1:
## Chain 1: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 1: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 1: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 1: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 1: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 1: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 1: Iteration: 1001 / 2000 [ 50%]  (Sampling)
```

```
## Chain 1: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 1: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 1: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 1: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 1: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 0.11686 seconds (Warm-up)
## Chain 1:                0.114916 seconds (Sampling)
## Chain 1:                0.231776 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL 'count' NOW (CHAIN 2).
## Chain 2:
## Chain 2: Gradient evaluation took 1.6e-05 seconds
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0.16 seconds.
## Chain 2: Adjust your expectations accordingly!
## Chain 2:
## Chain 2:
## Chain 2: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 2: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 2: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 2: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 2: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 2: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 2: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 2: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 2: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 2: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 2: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 2: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 2:
## Chain 2:  Elapsed Time: 0.111369 seconds (Warm-up)
## Chain 2:                0.118515 seconds (Sampling)
## Chain 2:                0.229884 seconds (Total)
## Chain 2:
##
## SAMPLING FOR MODEL 'count' NOW (CHAIN 3).
## Chain 3:
## Chain 3: Gradient evaluation took 2.3e-05 seconds
## Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0.23 seconds.
## Chain 3: Adjust your expectations accordingly!
## Chain 3:
## Chain 3:
## Chain 3: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 3: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 3: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 3: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 3: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 3: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 3: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 3: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 3: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 3: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 3: Iteration: 1800 / 2000 [ 90%]  (Sampling)
```

```
## Chain 3: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 3:
## Chain 3:  Elapsed Time: 0.109597 seconds (Warm-up)
## Chain 3:                0.113622 seconds (Sampling)
## Chain 3:                0.223219 seconds (Total)
## Chain 3:
##
## SAMPLING FOR MODEL 'count' NOW (CHAIN 4).
## Chain 4:
## Chain 4: Gradient evaluation took 2.7e-05 seconds
## Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 0.27 seconds.
## Chain 4: Adjust your expectations accordingly!
## Chain 4:
## Chain 4:
## Chain 4: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 4: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 4: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 4: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 4: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 4: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 4: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 4: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 4: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 4: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 4: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 4: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 4:
## Chain 4:  Elapsed Time: 0.116596 seconds (Warm-up)
## Chain 4:                0.094656 seconds (Sampling)
## Chain 4:                0.211252 seconds (Total)
## Chain 4:
## stan_glm
##  family:       neg_binomial_2 [log]
##  formula:      y ~ x
##  observations: 50
##  predictors:   2
## ------
##             Median MAD_SD
## (Intercept) 1.1    0.1
## x           1.9    0.1
##
## Auxiliary parameter(s):
##                       Median MAD_SD
## reciprocal_dispersion 7.0    2.2
##
## ------
## * For help interpreting the printed output see ?print.stanreg
## * For info on the priors used see ?prior_summary.stanreg
##
## SAMPLING FOR MODEL 'count' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 2.4e-05 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0.24 seconds.
## Chain 1: Adjust your expectations accordingly!
```

```
## Chain 1:
## Chain 1:
## Chain 1: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 1: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 1: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 1: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 1: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 1: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 1: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 1: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 1: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 1: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 1: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 1: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 0.11543 seconds (Warm-up)
## Chain 1:                0.116285 seconds (Sampling)
## Chain 1:                0.231715 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL 'count' NOW (CHAIN 2).
## Chain 2:
## Chain 2: Gradient evaluation took 1.6e-05 seconds
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0.16 seconds.
## Chain 2: Adjust your expectations accordingly!
## Chain 2:
## Chain 2:
## Chain 2: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 2: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 2: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 2: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 2: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 2: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 2: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 2: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 2: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 2: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 2: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 2: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 2:
## Chain 2:  Elapsed Time: 0.118187 seconds (Warm-up)
## Chain 2:                0.138964 seconds (Sampling)
## Chain 2:                0.257151 seconds (Total)
## Chain 2:
##
## SAMPLING FOR MODEL 'count' NOW (CHAIN 3).
## Chain 3:
## Chain 3: Gradient evaluation took 1.5e-05 seconds
## Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0.15 seconds.
## Chain 3: Adjust your expectations accordingly!
## Chain 3:
## Chain 3:
## Chain 3: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 3: Iteration:  200 / 2000 [ 10%]  (Warmup)
```

```
## Chain 3: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 3: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 3: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 3: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 3: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 3: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 3: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 3: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 3: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 3: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 3:
## Chain 3:  Elapsed Time: 0.124049 seconds (Warm-up)
## Chain 3:                0.139263 seconds (Sampling)
## Chain 3:                0.263312 seconds (Total)
## Chain 3:
##
## SAMPLING FOR MODEL 'count' NOW (CHAIN 4).
## Chain 4:
## Chain 4: Gradient evaluation took 2.2e-05 seconds
## Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 0.22 seconds.
## Chain 4: Adjust your expectations accordingly!
## Chain 4:
## Chain 4:
## Chain 4: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 4: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 4: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 4: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 4: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 4: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 4: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 4: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 4: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 4: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 4: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 4: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 4:
## Chain 4:  Elapsed Time: 0.113767 seconds (Warm-up)
## Chain 4:                0.135052 seconds (Sampling)
## Chain 4:                0.248819 seconds (Total)
## Chain 4:
## stan_glm
##  family:       neg_binomial_2 [log]
##  formula:      y ~ x
##  observations: 50
##  predictors:   2
## ------
##             Median MAD_SD
## (Intercept) 1.1    0.1
## x           1.9    0.1
##
## Auxiliary parameter(s):
##                       Median MAD_SD
## reciprocal_dispersion 6.9    2.3
##
```

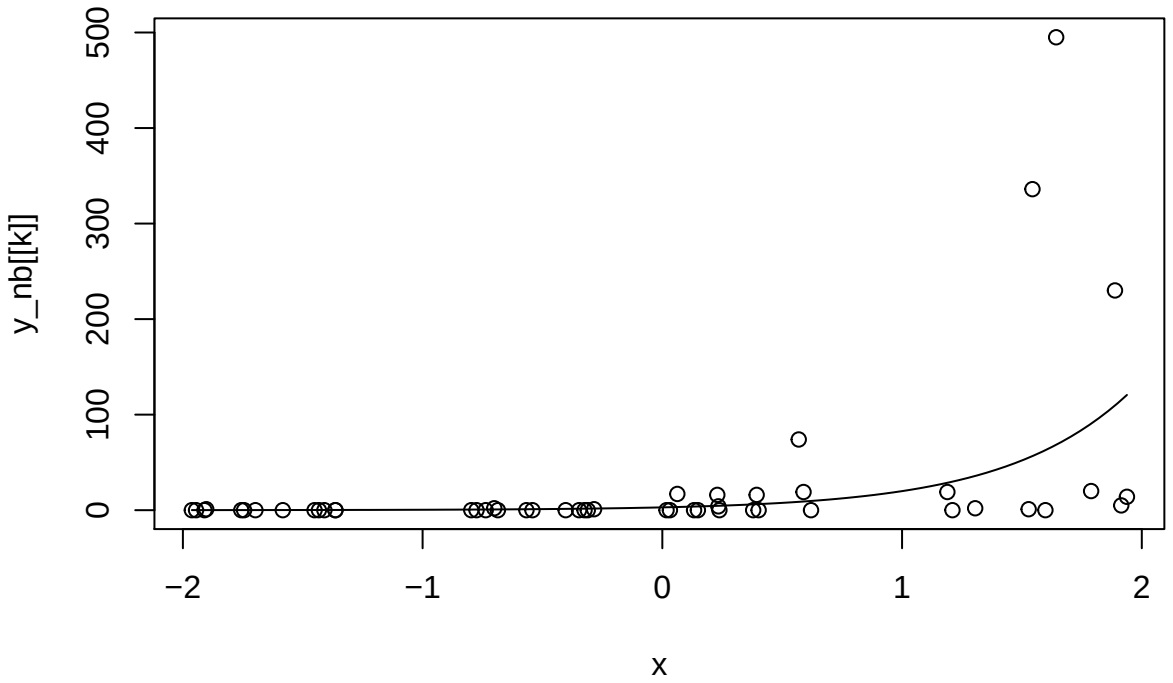
```
## ------
## * For help interpreting the printed output see ?print.stanreg
## * For info on the priors used see ?prior_summary.stanreg
##
## SAMPLING FOR MODEL 'count' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 2.3e-05 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0.23 seconds.
## Chain 1: Adjust your expectations accordingly!
## Chain 1:
## Chain 1:
## Chain 1: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 1: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 1: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 1: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 1: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 1: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 1: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 1: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 1: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 1: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 1: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 1: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 0.131918 seconds (Warm-up)
## Chain 1:                0.110409 seconds (Sampling)
## Chain 1:                0.242327 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL 'count' NOW (CHAIN 2).
## Chain 2:
## Chain 2: Gradient evaluation took 1.7e-05 seconds
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0.17 seconds.
## Chain 2: Adjust your expectations accordingly!
## Chain 2:
## Chain 2:
## Chain 2: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 2: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 2: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 2: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 2: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 2: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 2: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 2: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 2: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 2: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 2: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 2: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 2:
## Chain 2:  Elapsed Time: 0.11307 seconds (Warm-up)
## Chain 2:                0.115077 seconds (Sampling)
## Chain 2:                0.228147 seconds (Total)
## Chain 2:
##
```

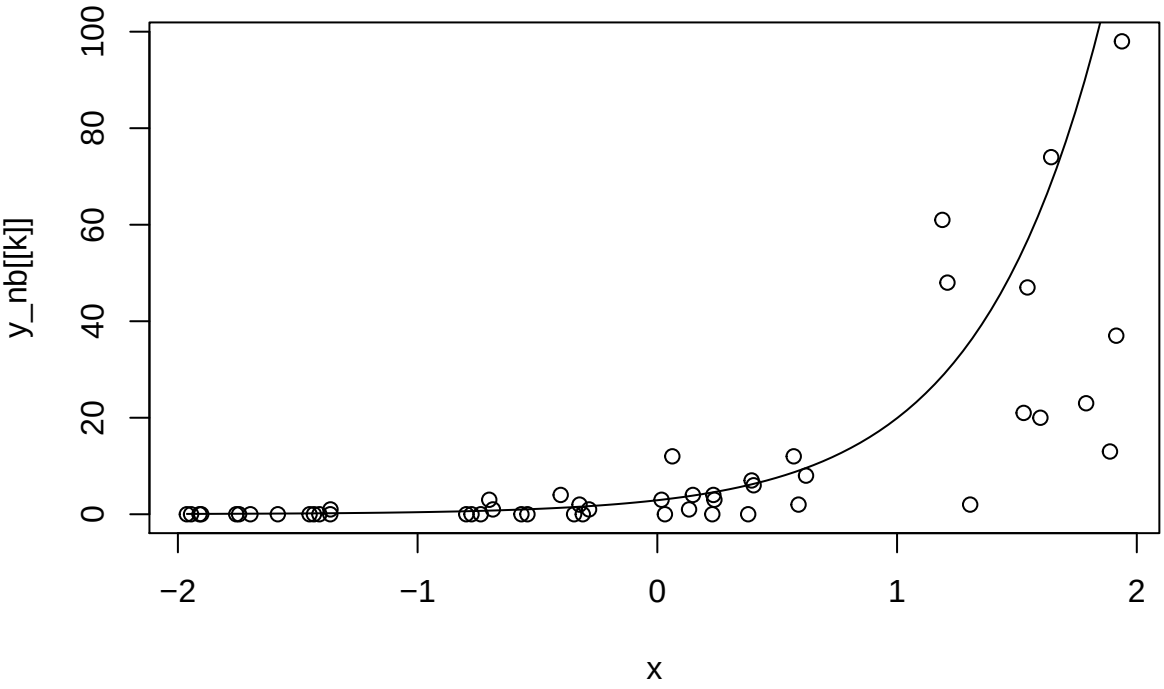
```
## SAMPLING FOR MODEL 'count' NOW (CHAIN 3).
## Chain 3:
## Chain 3: Gradient evaluation took 1.5e-05 seconds
## Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0.15 seconds.
## Chain 3: Adjust your expectations accordingly!
## Chain 3:
## Chain 3:
## Chain 3: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 3: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 3: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 3: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 3: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 3: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 3: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 3: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 3: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 3: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 3: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 3: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 3:
## Chain 3:  Elapsed Time: 0.121262 seconds (Warm-up)
## Chain 3:                0.127803 seconds (Sampling)
## Chain 3:                0.249065 seconds (Total)
## Chain 3:
##
## SAMPLING FOR MODEL 'count' NOW (CHAIN 4).
## Chain 4:
## Chain 4: Gradient evaluation took 2.4e-05 seconds
## Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 0.24 seconds.
## Chain 4: Adjust your expectations accordingly!
## Chain 4:
## Chain 4:
## Chain 4: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 4: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 4: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 4: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 4: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 4: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 4: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 4: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 4: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 4: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 4: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 4: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 4:
## Chain 4:  Elapsed Time: 0.115895 seconds (Warm-up)
## Chain 4:                0.154346 seconds (Sampling)
## Chain 4:                0.270241 seconds (Total)
## Chain 4:
## stan_glm
##  family:       neg_binomial_2 [log]
##  formula:      y ~ x
##  observations: 50
##  predictors:   2
```

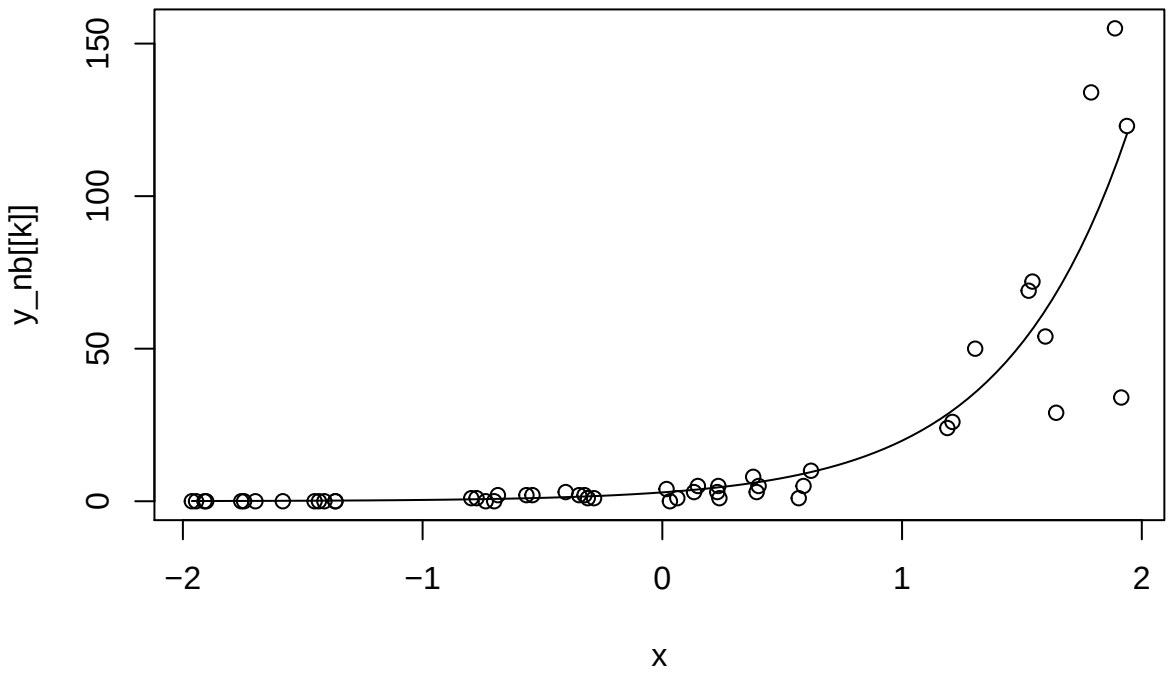
```
## ------
##             Median MAD_SD
## (Intercept) 1.1    0.1
## x           1.9    0.1
##
## Auxiliary parameter(s):
##                       Median MAD_SD
## reciprocal_dispersion 7.0    2.2
##
## ------
## * For help interpreting the printed output see ?print.stanreg
## * For info on the priors used see ?prior_summary.stanreg
```

finalmente, se grafica cada dataset a lo largo de las 3 curvas

```
for (k in 1:K) {
plot(x, y_nb[[k]])
curve(exp(coef(fit_nb[[k]])[1] + coef(fit_nb[[k]])[2]*x), add=TRUE)
}
```




y_nb[[k]]
0
20
40
60
80
100
−2
−1
0
1
2
x



Interpretación de coeficientes de regresión binomial negativa o poisson

El coeficiente beta en un modelo de regresión logarítmico puede ser exponenciado y tratado como efecto multiplicativo. Por ejemplo, supongo que el modelo de tráfico de accidentes de tráfico es:

$$y_i \sim bonimial - negativo(exp(2.8 + 0.012X_{ii} - 0.20X_{i2}), \phi)$$

,

Donde X_{i1} es la velocidad promedio en millas por hora en las calles cercanas y $X_{i2} = 1$ representa si la intersección de calles tiene una señal de tráfico y cero en caso contrario. Podemos interpretar cada coeficiente como sigue.

El término constante da el intercepto de regresión, esto es, la predicción cuando $x_{i2} = 0$ y $X_{i2} = 0$. Debido a que no es posible (ya que ninguna calle tendrá velocidad promedio del tráfico igual a cero) no trataremos de interpretar el término constante.

El coeficiente de x_{i1} es la diferencia esperada en Y (en escala logarítmica) por cada milla por hora (mph) de velocidad adicional. Así que el incremento multiplicativo esperado es 1.012 o 1.2% de diferencia positiva en la tasa de accidentes de tráfico por millas por hora. Debido a que la velocidad del tráfico varía por decenas de millas por hora, podría tener sentido definir X_{i1} como la velocidad en decenas de millas por hora. En el caso en el que el coeficiente fuese 0.12 correspondiente al 12% en la tasa de accidentes por millas por hora. (más precisamente... $e^{0.12} = 1.127$: incremento de 12.7%).

El coeficiente de X_{i2} nos dice que la diferencia predictiva de tener señales de tráfico puede ser encontrada al multiplicar la tasa de accidente por $e^{-0.20} = 0.82$, produciendo una reducción del 18%.

Exposición

En muchas aplicaciones de regresión de conteo de datos, existe una línea base o exposición, que puede ser el promedio del flujo de vehículos que circulan al rededor de una intersección de accidentes de tráfico. Se puede modelar y_i como el número de casos en un proceso con tasa θ_i y exposición μ_i.

$$y_i \sim binomial - negativo(u_i\theta_i, \phi),$$

donde $\theta_i = e^{X_i B}$.

El logaritmo de la exposición, $log(u_i)$, es llamado *offset* en la terminología de modelos lineales generalizados. El coeficiente de regresión β refleja ahora la asociación entre los predictores θ_i (en nuestro ejemplo, la tasa de accidentes de tráfico por vehículo).

Diferencia entre modelo binomial y poisson o binomial negativo.

El modelo poisson o binomial negativo son similares al modelo binomial para conteo de datos, pero son aplicados en situaciones ligeramente distintas.

- Si cada punto y_i puede ser interpretado como el número de éxitos de n_i ensayos, entonces es conveniente usar el modelo binomial logístico o su generalización sobredispersa.
- Si cada punto de datos y_i no tiene un límite natural (no está basado en un número de ensayos independientes) , entonces es natural usar el modelo poisson o binomial negativo con liga logarítmica.
- Si cada punto y_i tiene un límite natural que es mucho más grande del número esperado de conteos, entonces la regresión Poisson/logarítmica puede ser usada para aproximar el modelo binomial.

Modelo binomial logístico

El modelo logístico puede ser usado también para contar datos, usando las distribución binomial para modelar el número de éxitos de un número específico de posibilidades, con la probabilidad de éxito siendo ajustada a una regresión logística.

Lo demostramos con un modelo simple de tiros de basketball. Primero simulamos $N = 100$ jugadores y cada uno tira $n = 20$ lanzamientos, donde la probabilidad de éxito de tiro es una función lineal de altura (30% para jugadores de altura 5'9'', 40% para jugadores de 6' de altura, y así):

```
N <- 100
height <- rnorm(N, 72, 3)
p <- 0.4 + 0.1*(height - 72)/3
n <- rep(20, N)
y <- rbinom(N, n, p)
data <- data.frame(n=n, y=y, height=height)
head(data)
```

```
##    n  y   height
## 1 20 12 73.79707
## 2 20  6 68.74454
## 3 20  8 76.25791
## 4 20  7 73.41490
## 5 20  9 72.86357
## 6 20  3 69.67042
```

Podemos ajustar y mostrar la probabilidad de éxito predicha por el modelo de regresión logística dada por la altura:

```
fit_1a <- stan_glm(cbind(y, n-y) ~ height,
                   family = binomial(link = "logit"),
                   data = data)
```

```
##
## SAMPLING FOR MODEL 'binomial' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 8.6e-05 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0.86 seconds.
## Chain 1: Adjust your expectations accordingly!
## Chain 1:
## Chain 1:
## Chain 1: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 1: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 1: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 1: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 1: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 1: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 1: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 1: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 1: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 1: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 1: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 1: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 0.174568 seconds (Warm-up)
## Chain 1:                0.182122 seconds (Sampling)
## Chain 1:                0.35669 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL 'binomial' NOW (CHAIN 2).
## Chain 2:
## Chain 2: Gradient evaluation took 3.6e-05 seconds
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0.36 seconds.
## Chain 2: Adjust your expectations accordingly!
## Chain 2:
## Chain 2:
## Chain 2: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 2: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 2: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 2: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 2: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 2: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 2: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 2: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 2: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 2: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 2: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 2: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 2:
## Chain 2:  Elapsed Time: 0.161361 seconds (Warm-up)
```

```
## Chain 2:                0.190735 seconds (Sampling)
## Chain 2:                0.352096 seconds (Total)
## Chain 2:
##
## SAMPLING FOR MODEL 'binomial' NOW (CHAIN 3).
## Chain 3:
## Chain 3: Gradient evaluation took 3.4e-05 seconds
## Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0.34 seconds.
## Chain 3: Adjust your expectations accordingly!
## Chain 3:
## Chain 3:
## Chain 3: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 3: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 3: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 3: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 3: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 3: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 3: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 3: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 3: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 3: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 3: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 3: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 3:
## Chain 3:  Elapsed Time: 0.190456 seconds (Warm-up)
## Chain 3:                0.181254 seconds (Sampling)
## Chain 3:                0.37171 seconds (Total)
## Chain 3:
##
## SAMPLING FOR MODEL 'binomial' NOW (CHAIN 4).
## Chain 4:
## Chain 4: Gradient evaluation took 3.5e-05 seconds
## Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 0.35 seconds.
## Chain 4: Adjust your expectations accordingly!
## Chain 4:
## Chain 4:
## Chain 4: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 4: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 4: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 4: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 4: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 4: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 4: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 4: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 4: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 4: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 4: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 4: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 4:
## Chain 4:  Elapsed Time: 0.169026 seconds (Warm-up)
## Chain 4:                0.16286 seconds (Sampling)
## Chain 4:                0.331886 seconds (Total)
## Chain 4:
```

```
print(fit_1a)
```

```
## stan_glm
##  family:       binomial [logit]
##  formula:      cbind(y, n - y) ~ height
##  observations: 100
##  predictors:   2
## ------
##             Median MAD_SD
## (Intercept) -11.8    1.1
## height        0.2    0.0
##
## ------
## * For help interpreting the printed output see ?print.stanreg
## * For info on the priors used see ?prior_summary.stanreg
```

El modelo binomial para conteo de datos (aplicado a sentencias de muerte)

Ilustramos la regresión binomial logística en el contexto de estudio de la proporción de veredictos de pena de muerte que fueron anuladas en 34 estados entre 1973 y 1995. Las unidades de este análisis son los 34*23=784 estados-años (aunque sólo hay n=450 estados-años en el análisis). Para cada estado-año* i*, etiquetamos como n_i al número de sentencias de muerte en el estado-año y como y_1 al número de veredictos que fueron anulados. El modelo es de la forma:

$$y_i \sim Binomial(n_i, p_i),$$
$$p_i = logit^{-1}(X_i B),$$

Donde X_i es la matriz de predictores. Para empezar, incluimos

- Un término constante
- 33 indicadores parta estados
- La tendencia en los años (esta variable es 1 para 1973, 2 para 1974, 3 para 1975 y así). El modelo puede escribirse como:

$$y_{st} \sim Binomial(n_{st}, p_{st})$$
$$p_{st} = logit^{-1}(\mu + \alpha_{st} + \beta_t)$$

con subíndice s para eñ estado y t para el tiempo