# EEEB UN3005/GR5005
# Lab - Week 12 - 15 and 17 April 2019

*USE YOUR NAME HERE*

## Binomial Regression

### Exercise 1: Fitting a Binomial Generalized Linear Model

On either the class CourseWorks or GitHub page, you'll find a dataset called `eagles.csv`. This dataset summarizes observations of salmon foraging interactions by bald eagles in the western United States. More specifically, when one eagle (the victim) is trying to feed, another eagle (the pirate) may try to steal the catch. The `eagles.csv` data records the number of successful thieving events by the pirate eagle (`successes`) given a total number of thieving attempts (`total_attempts`). In addition, there is information on the specifics of the eagle interaction including size of the pirating eagle, age of the pirating eagle, and size of the victim eagle.

Import the `eagles.csv` data, and fit a binomial generalized linear with `successes` as the outcome variable and `pirate_size` and `victim_size` as predictor variables. Note, you'll need to create dummy predictor variables for use in your model. Fit your model using `map2stan()` with four MCMC chains and priors of `dnorm(0, 10)` for the intercept parameter and `dnorm(0, 5)` for the beta coefficients. After fitting your model, report the 97% HPDI for all model parameters.

```
d = read.csv('../Data/eagles.csv')
d$pirate_size = ifelse(d$pirate_size == 'large', 1, 0)
d$victim_size = ifelse(d$victim_size == 'large', 1, 0)
model.1 = map2stan(
    alist(
        successes ~ dbinom(size = total_attempts, prob = p),
        logit(p) <- a + bP * pirate_size + bV * victim_size,
        a ~ dnorm(0, 10),
        bP ~ dnorm(0, 5),
        bV ~ dnorm(0, 5)),
    data = d,
    chains = 2)
```

```
##
## SAMPLING FOR MODEL 'successes ~ dbinom(size = total_attempts, prob = p)' 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 / 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: 0.087 seconds (Warm-up)
## Chain 1:                0.089 seconds (Sampling)
## Chain 1:                0.176 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL 'successes ~ dbinom(size = total_attempts, prob = p)' 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 / 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: 0.081 seconds (Warm-up)
## Chain 2:                0.079 seconds (Sampling)
## Chain 2:                0.16 seconds (Total)
## Chain 2:
```

```
##
## SAMPLING FOR MODEL 'successes ~ dbinom(size = total_attempts, prob = p)' 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: WARNING: No variance estimation is
## Chain 1:          performed for num_warmup < 20
## Chain 1:
## Chain 1: Iteration: 1 / 1 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 0 seconds (Warm-up)
## Chain 1:                0 seconds (Sampling)
## Chain 1:                0 seconds (Total)
## Chain 1:
```

```
## Computing WAIC
```

```
## Constructing posterior predictions
```

```
## [ 200 / 2000 ]
[ 400 / 2000 ]
[ 600 / 2000 ]
[ 800 / 2000 ]
[ 1000 / 2000 ]
[ 1200 / 2000 ]
[ 1400 / 2000 ]
[ 1600 / 2000 ]
[ 1800 / 2000 ]
[ 2000 / 2000 ]
```

```
## Aggregated binomial counts detected. Splitting to 0/1 outcome for WAIC calculation.
```

```
precis(model.1, 0.97)
```

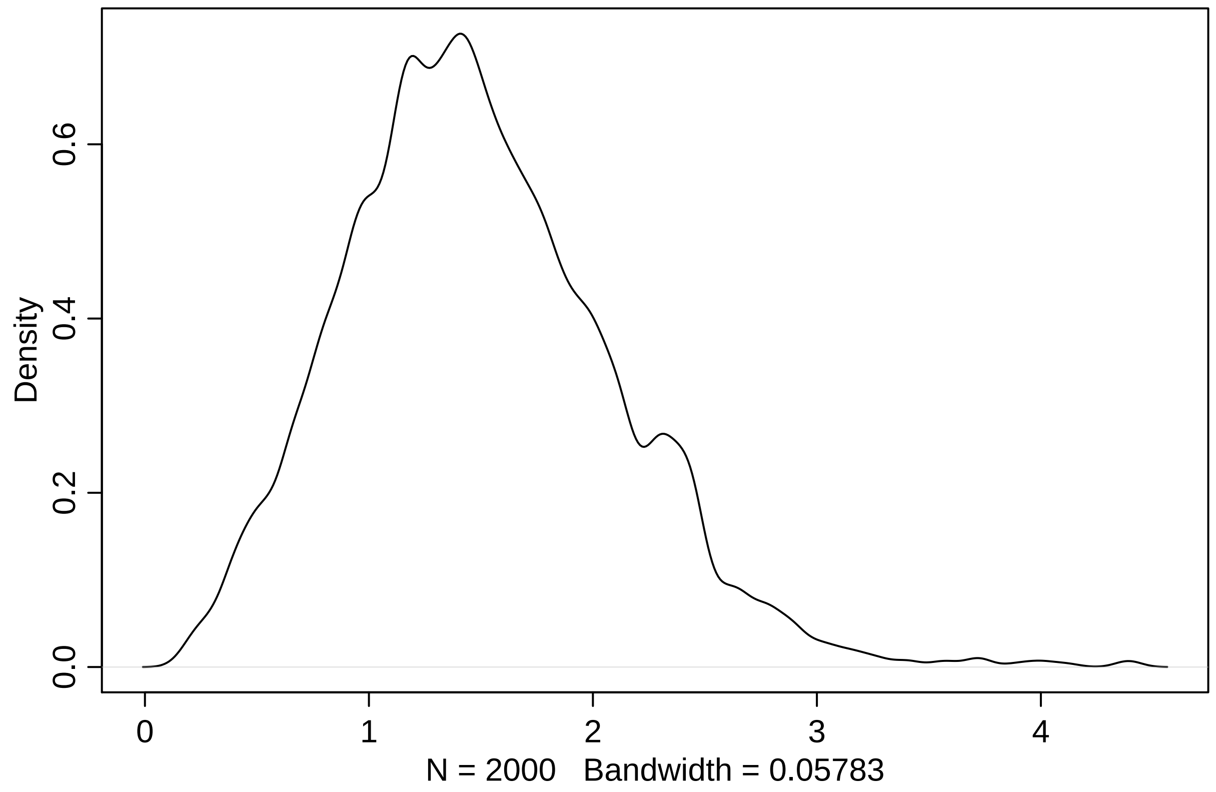
```
##     Mean StdDev lower 0.89 upper 0.89 n_eff Rhat
## a   1.49   0.60       0.63       2.46   773 1.00
## bP  4.54   0.97       3.07       6.02   632 1.00
## bV -5.32   1.09      -7.00      -3.60   548 1.01
```

## Exercise 2: Visualizing the Model Intercept and the Implied Probability of Success Values

Extract posterior samples from your fit model and visualize the intercept parameter samples using `dens()`. What type of eagle interaction (i.e., combination of large/small pirate and victim) does the intercept value represent?

The 97% HPDI for the fit model parameter that you just visualized should already be represented in your answers for Exercise 1. But now can you report the 97% HPDI for the probability of success (i.e., probability of thieving) that this raw parameter estimate implies? As a hint, this will involve reversing the link function that is used in fitting the model. Similarly, visualize the implied probability of success values using `dens()`.

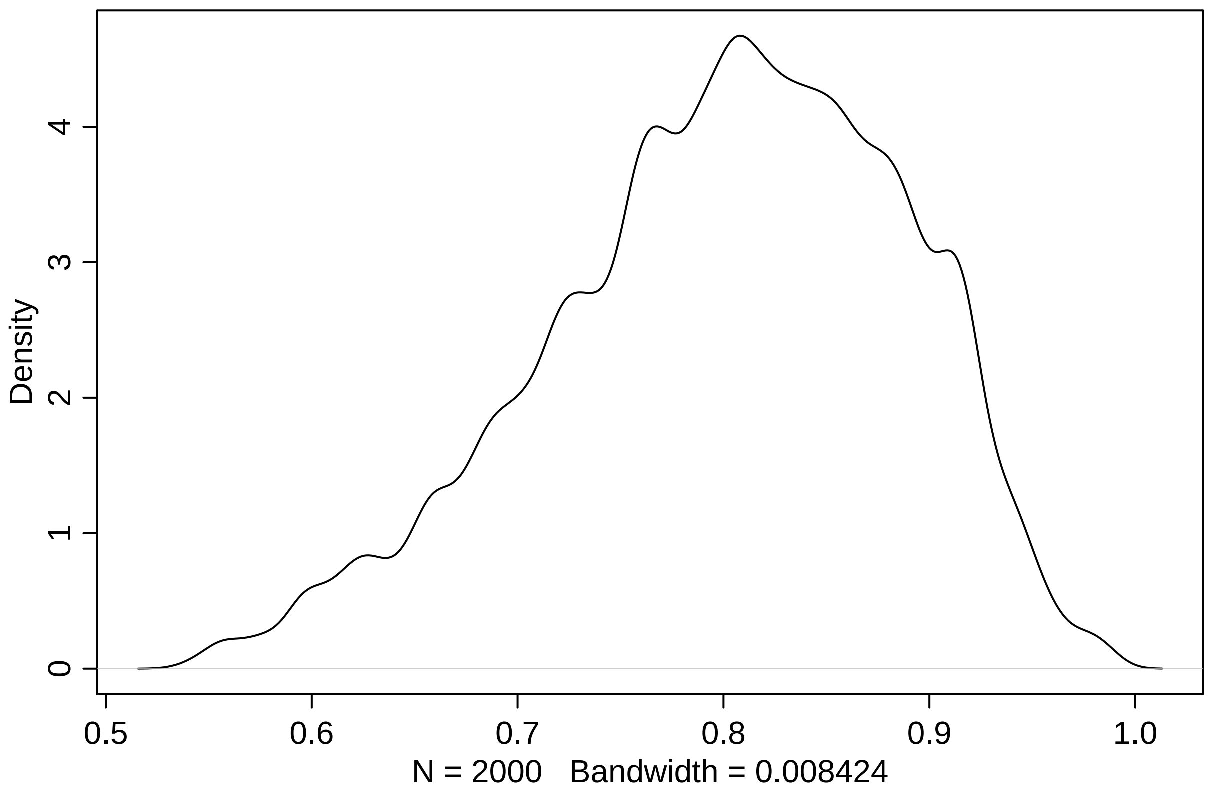
```
sample = extract.samples(model.1, n = 10000)
dens(sample$a)
```



```
HPDI(logistic(sample$a), prob = 0.97)
```

```
##     |0.97     0.97|
## 0.6103031 0.9617546
```

```
dens(logistic(sample$a))
```



---

**Answer:**

It represents the small pirate and small victim.

---

## Exercise 3: Plotting Implied Probability of Success Values for All Pirate-Victim Combinations

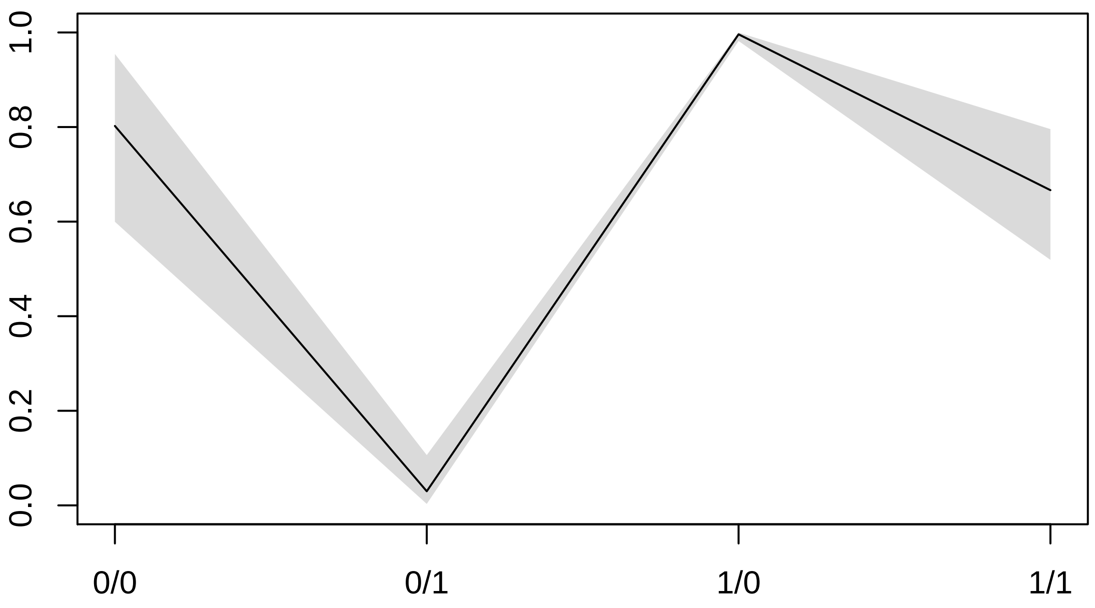
Now extend the ideas you just implemented in Exercise 2 to use `dens()` to plot implied probability of success values for all pirate-victim size combinations. Rather than generating these implied values yourself manually, you can create counterfactual datasets and let `link()` generate the implied values for you. After you've generated implied probability of success values for all pirate-victim combinations, the plotting should be straightforward.

```
df = data.frame(
    pirate_size = c(0, 0, 1, 1),
    victim_size = c(0, 1, 0, 1))
preds = link(model.1, data = df)
```

```
## [ 100 / 1000 ]
```

```
[ 200 / 1000 ]
[ 300 / 1000 ]
[ 400 / 1000 ]
[ 500 / 1000 ]
[ 600 / 1000 ]
[ 700 / 1000 ]
[ 800 / 1000 ]
[ 900 / 1000 ]
[ 1000 / 1000 ]
```

```
preds.mu = apply(preds, 2, mean)
preds.PI = apply(preds, 2, PI, prob = 0.97)
plot(0, 0, xlim = c(1, 4), ylim = c(0, 1), type = 'n', xaxt = 'n', xlab = '', ylab = '')
axis(1, at = 1:4, labels = c('0/0', '0/1', '1/0', '1/1'))
lines(1:4, preds.mu)
shade(preds.PI, 1:4)
```



0.0
0.2
0.4
0.6
0.8
1.0
0/0
0/1
1/0
1/1