Maximum Entropy and Generalized Linear Models

EES 5891-03

Bayesian Statistical Methods

Jonathan Gilligan

Class #13: Tuesday, October 11 2022

Beyond Normal Likelihoods

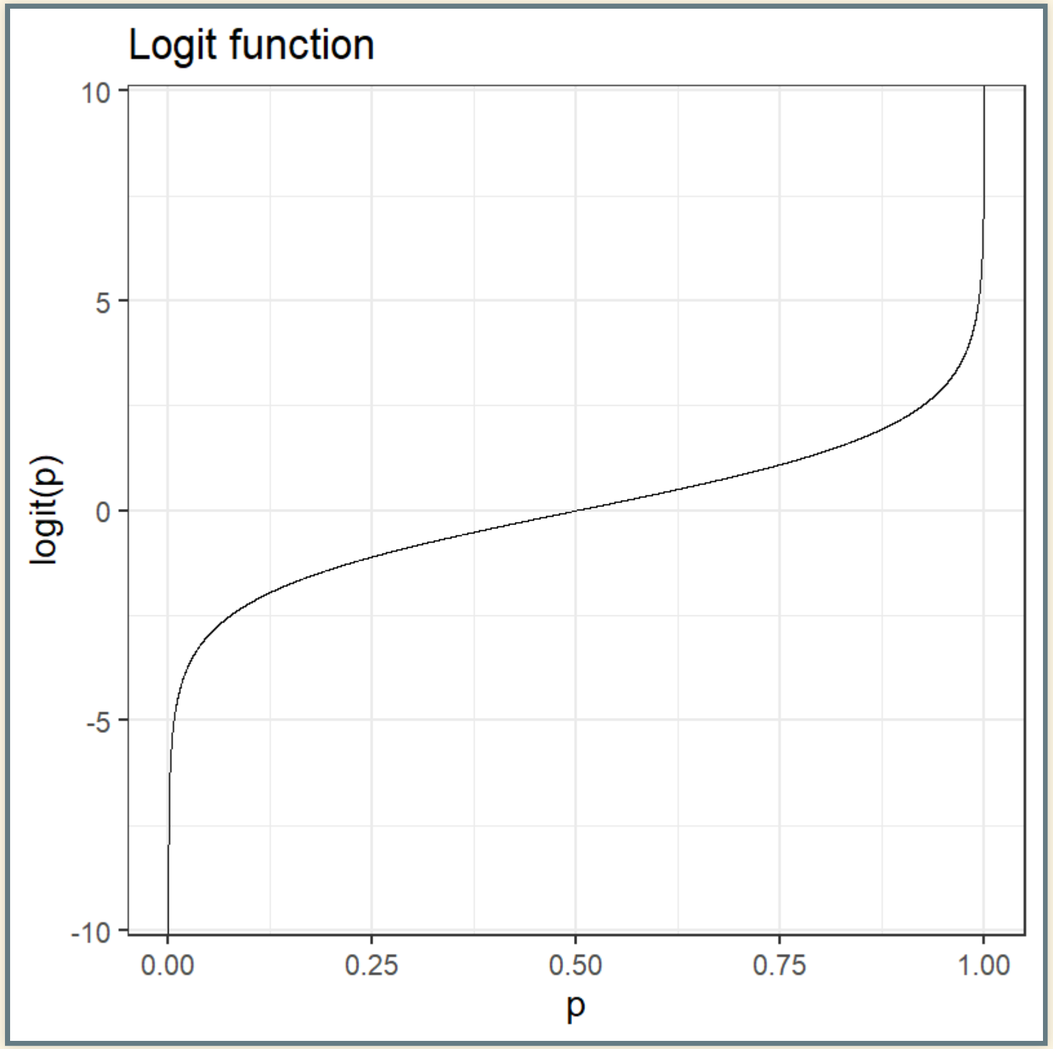
Linear Models

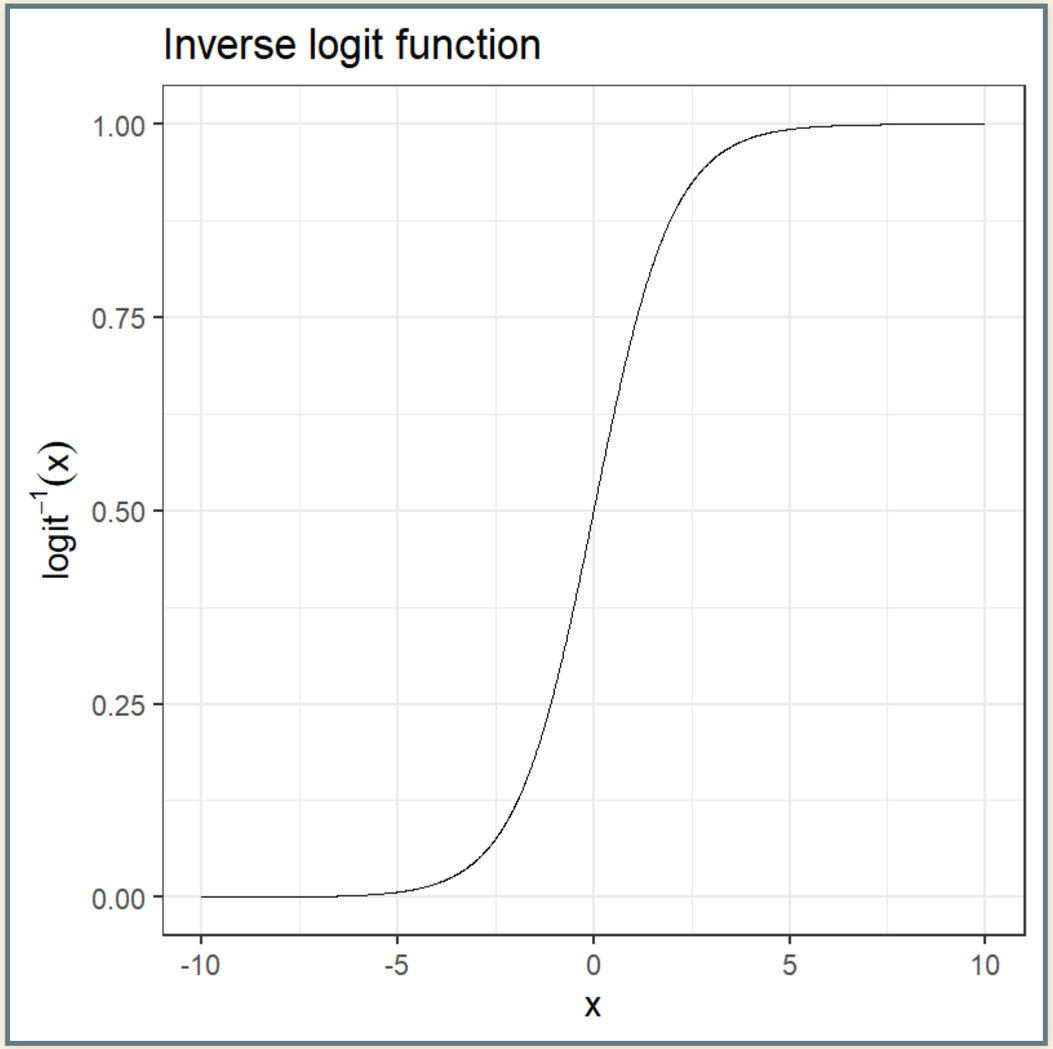
- Models we've considered so far have the form $$ \begin{align} y &\sim \text{Normal}(\mu, \sigma) \\ \mu &= \alpha + \beta x \end{align} $$
- This model is a *linear regression* because μ is a linear function of x.
- We can have multiple x variables (e.g., water and shade), so the equation for μ becomes $$ \mu = \alpha + \sum_i \beta_i x_i $$
- We can treat polynomial and interaction terms as linear terms:
  - Instead of $$ \begin{align} \mu &= \alpha + \beta_x x + \beta_{x^2} x^2 + \beta_{x^3} x^3 + \beta_z z + \beta_{xz} x z \end{align} $$ we can write $$ \begin{align} x_2 &= x^2 \\ x_3 &= x^3 \\ x_4 &= z \\ x_5 &= x z \\ \mu &= \alpha + \beta_1 x + \beta_2 x_2 + \beta_3 x_3 + \beta_4 x_4 + \beta_5 x_5 \end{align} $$

Generalized Linear Models

- What if y is not normally distributed for a given x?
  - The offspring of an animal have a probability p of being female and $1 - p$ of being male
  - We want to find out how the animal's diet affects the fraction of female offspring.
  - We count the total number of offspring (N) and the number of females (F). $$\begin{align} F &\sim \text{Binomial}(N, p) \\ f(p) &= \alpha + \beta x \end{align}$$
    - $f()$ is a *link function*.
- This is a *generalized linear model* because $f(p)$ is a linear function of x.
- $0 \le p \le 1$, so we need a *link function* to map between the valid range of p and the unlimited range of $\alpha + \beta x$.
- A common link function for probabilities is the *logit* function $$\begin{align} \text{logit}(p) &= \ln \frac{p}{1 - p} \\ \text{logit}^{-1}(x) &= \frac{1}{1 + \exp(-x)} = \frac{\exp(x)}{\exp(x) + 1} \end{align}$$
  - *logit* maps a probability in the range $[0,1]$ to $[-\infty, \infty]$.
  - $logit^{-1}$ maps the real axis $[-\infty, \infty]$ to a probability in the range $[0,1]$

Logit link function





Choosing Likelihood Functions

Likelihood Functions

- In the model $$ \begin{align} F &\sim \text{Binomial}(N, p) \\ \text{logit}(p) &= \alpha + \beta x \end{align} $$ $\text{Binomial}(N, p)$ defines a *likelihood* $P(F \mid N, p)$ observing F female offspring, given that there are N total offspring and the probability of any offspring being female is p.
- Why do we choose a *Binomial* distribution here instead of some other distribution?
- In the model $$ \begin{align} F &\sim \text{Normal}(\mu, \sigma) \\ \mu &= \alpha + \beta x \end{align} $$ Why do we choose a *Normal* distribution?

- Information theory: The likelihood function that makes the fewest assumptions about the data will be the one that has the largest *information entropy* under a set of constraints.
- *Information entropy* is related to counting the different ways you can get a result, so *maxiumum entropy* corresponds to the result that has the greatest number of ways that it can happen.

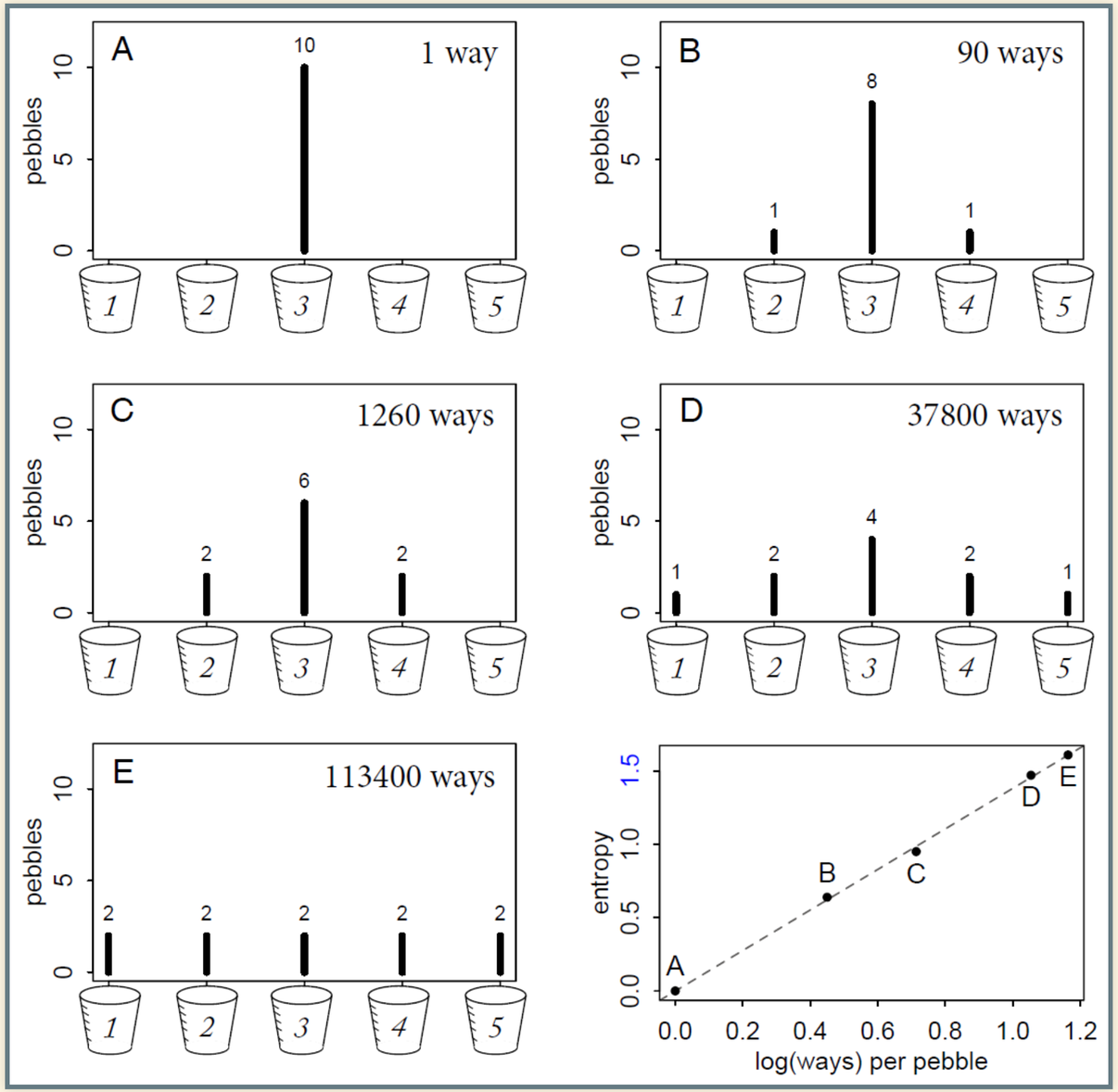
Information Entropy

- Find a measure of uncertainty where:
  1. It is continuous
  2. It increases as the number of possible events increases
  3. It is additive: $f(a + b) = f(a) + f(b)$
- Information Entropy: For a set of events i, with probabilities p_i, $$ H(p) = - \sum_i p_i \log p_i $$

Example

- Distribute 10 pebbles in 5 buckets, where each pebble has an equal probability to land in any bucket.
  - 5^{10} (9,765,625) ways to do this, if we pay attention to which pebble is in which bucket.
  - If we just care about the number of pebbles in each bucket, the number of ways to get \(n_1, n_2, n_3, n_4, n_5\) pebbles in buckets 1–5 is \[p(n_1, n_2, n_3, n_4, n_5) = \frac{N!}{n_1! n_2! n_3! n_4! n_5!},\]
- The number of ways to get all the pebbles in bucket 3 is \[\frac{10}{0! \times 0! \times 10! \times 0! \times 0!} = \frac{10!}{10!} = 1 \]
- The number of ways to get 1 pebble in bucket 2, 8 in bucket 3, and 1 in bucket 4 is \[\frac{10}{0! \times 1! \times 8! \times 1! \times 0!} = \frac{10!}{8!} = 90 \]
- The number of ways to get 2 pebbles in bucket 2, 6 in bucket 3, and 2 in bucket 4 is \[\frac{10}{0! \times 2! \times 6! \times 2! \times 0!} = 1,260 \]

Example


A
1 way
10
B
90 ways
1
8
1
C
1260 ways
2
6
2
D
37800 ways
1
2
4
2
1
E
113400 ways
2
2
2
2
2
pebbles
0
5
10
1
2
3
4
5
entropy
0.0
0.5
1.0
1.5
A
B
C
D
E
0.0
0.2
0.4
0.6
0.8
1.0
1.2
log(ways) per pebble

Application to Bayesian Statistics

- Continuously distributed variable:
  - If all we know is the mean and the standard deviation, then the *normal distribution* is the distribution with the greatest information entropy.
  - If we know something else, beside the mean and standard deviation, then that imposes a new constraint, and the normal distribution may not have the maximum entropy.
- Binomial data:
  - Two possible outcomes (e.g., Heads and Tails) Count the number of heads in N trials, if the probability of getting heads in any trial is p.
  - If we know p is the same for each trial, then the *binomial distribution* has the greatest information entropy.
    - If p can vary from one trial to another, then other distributions will maximuze entropy (e.g., *beta binomial distribution*).

Homework

Exercise 8M4

Repeat the tulips analysis, but this time use priors that constrain the effect of water to be positive and the effect of shade to be negative. Use prior predictive simulation. What do these prior assumptions mean for the interaction prior, if anything?

- Set the problem up

```
library(tidyverse)
library(rethinking)

data(tulips)
d <- tulips %>%
  mutate(
    blooms_std = blooms / max(blooms),
    water_cent = water - mean(water),
    shade_cent = shade - mean(shade)
  )
```

- Define a model with no interaction

```
 mdl <- quap(
alist(
  blooms_std ~ dnorm(mu, sigma),
  mu <- a + bw * water_cent - bs * shade_cent,
  a ~ dnorm(0.5, 0.25),
  bw ~ dlnorm(0, 0.25),
  bs ~ dlnorm(0, 0.25),
  sigma ~ dexp(1)
  ), data = d)
```

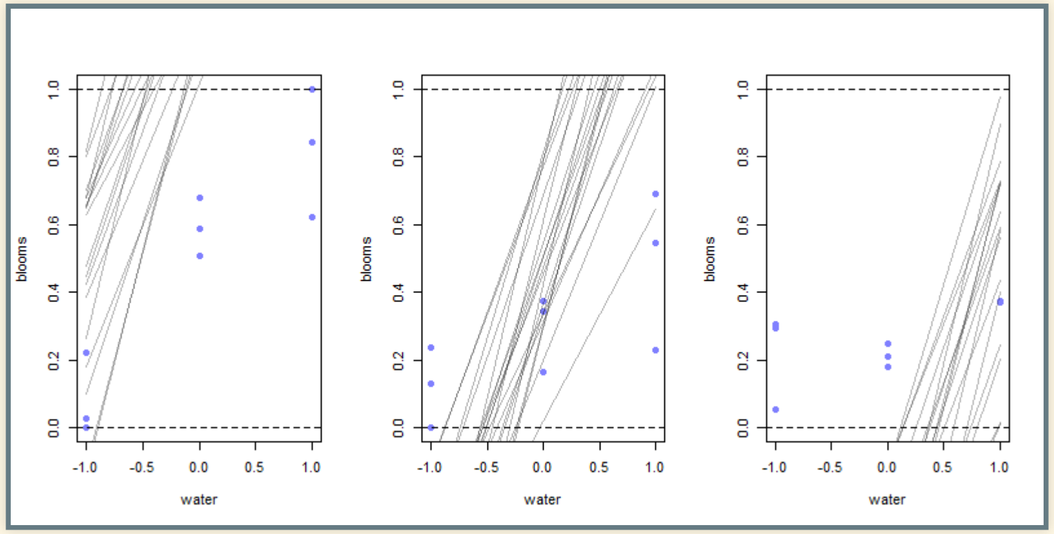
Prior predictive simulation

```
mdl <- quap(
  alist(
    blooms_std ~ dnorm(mu, sigma),
    mu <- a + bw * water_cent - bs * shade_cent,
    a ~ dnorm(0.5, 0.25),
    bw ~ dlnorm(0, 0.25),
    bs ~ dlnorm(0, 0.25),
    sigma ~ dexp(1)
    ), data = d)

prior <- extract.prior(mdl)

par(mfrow=c(1,3)) # 3 plots in 1 row

for (s in -1:1){
  idx <- which(d$shade_cent == s)
  plot( d$water_cent[idx], d$blooms_std[idx],
        xlim=c(-1,1), ylim=c(0,1),
        xlab = "water", ylab = "blooms",
        pch = 16, col = rangi2)
  abline( h=0,lty=2)
  abline( h=1,lty=2)
  mu <- link(mdl, data = data.frame(shade_cent = s,
                                    water_cent = -1:1),
             post = prior)
  for (i in 1:20)
    lines(-1:1, mu[i,], col = col.alpha("black", 0.3))
}
```



- Focus first on the center panel (shade = 0).
- Slope is too steep, range of slopes is too small:
  - Change the prior for \(\beta_w\)

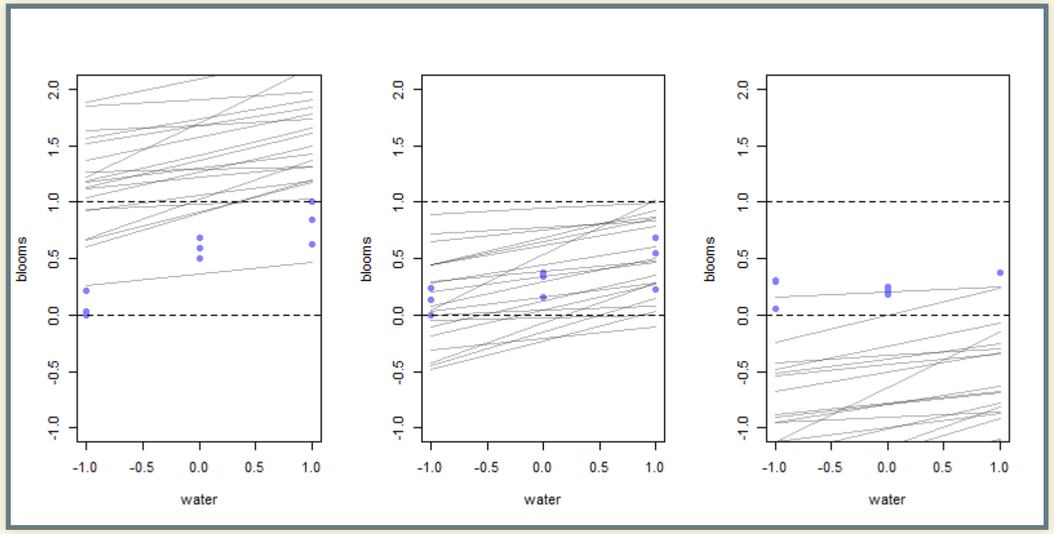
Prior predictive simulation

```
mdl.2 <- quap(
  alist(
    blooms_std ~ dnorm(mu, sigma),
    mu <- a + bw * water_cent - bs * shade_cent,
    a ~ dnorm(0.25, 0.35),
    bw ~ dlnorm(-2, 1),
    bs ~ dlnorm(0, 0.25),
    sigma ~ dexp(1)
    ), data = d)

prior <- extract.prior(mdl.2)

par(mfrow=c(1,3)) # 3 plots in 1 row

for (s in -1:1){
  idx <- which(d$shade_cent == s)
  plot( d$water_cent[idx], d$blooms_std[idx],
        xlim=c(-1,1), ylim=c(-1,2),
        xlab = "water", ylab = "blooms",
        pch = 16, col = rangi2)
  abline( h=0,lty=2)
  abline( h=1,lty=2)
  mu <- link(mdl.2, data = data.frame(shade_cent = s,
                                      water_cent = -1:1),
             post = prior)
  for (i in 1:20)
    lines(-1:1, mu[i,], col = col.alpha("black", 0.3))
}
```



- Look at left & right panels (shade)
- Slope is too steep:
  - Left is too high, right is too low.
  - Change the prior for \(\beta_s\)

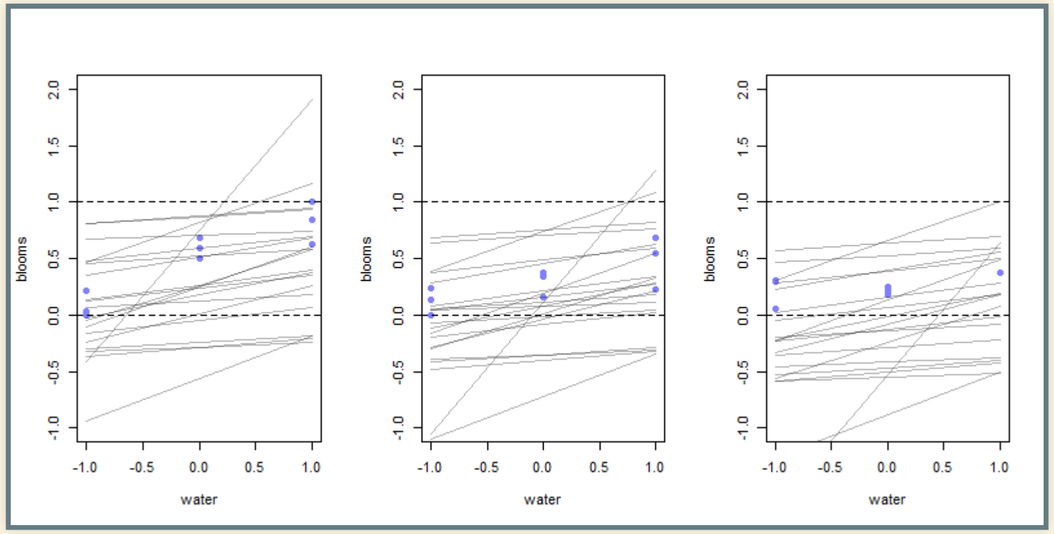
Prior predictive simulation

```
mdl.3 <- quap(
  alist(
    blooms_std ~ dnorm(mu, sigma),
    mu <- a + bw * water_cent - bs * shade_cent,
    a ~ dnorm(0.25, 0.35),
    bw ~ dlnorm(-2, 1),
    bs ~ dlnorm(-2, 1),
    sigma ~ dexp(1)
    ), data = d)

prior <- extract.prior(mdl.3)

par(mfrow=c(1,3)) # 3 plots in 1 row

for (s in -1:1){
  idx <- which(d$shade_cent == s)
  plot( d$water_cent[idx], d$blooms_std[idx],
        xlim=c(-1,1), ylim=c(-1,2),
        xlab = "water", ylab = "blooms",
        pch = 16, col = rangi2)
  abline( h=0,lty=2)
  abline( h=1,lty=2)
  mu <- link(mdl.3, data = data.frame(shade_cent = s,
                                      water_cent = -1:1),
             post = prior)
  for (i in 1:20)
    lines(-1:1, mu[i,], col = col.alpha("black", 0.3))
}
```

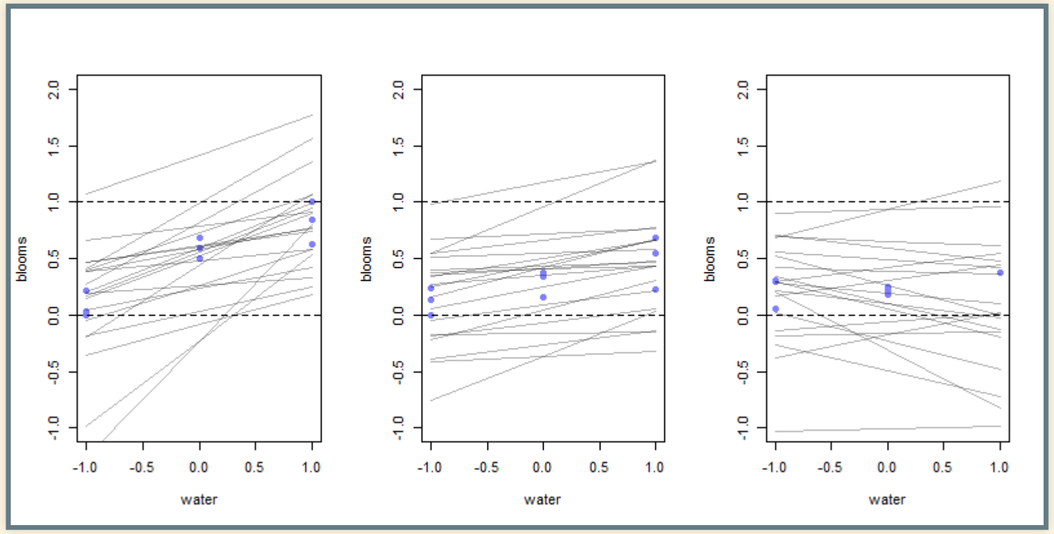


- This looks good.
  - Now make a model with an interaction ...

Prior predictive simulation

```
mdl.4 <- quap(
  alist(
    blooms_std ~ dnorm(mu, sigma),
    mu <- a + bw * water_cent - bs * shade_cent -
      bws * water_cent * shade_cent,
    a ~ dnorm(0.25, 0.35),
    bw ~ dlnorm(-2, 1),
    bs ~ dlnorm(-2, 1),
    bws ~ dlnorm(-2, 1),
    sigma ~ dexp(1)
    ), data = d)
```

- Why do we use a minus sign for the interaction term?

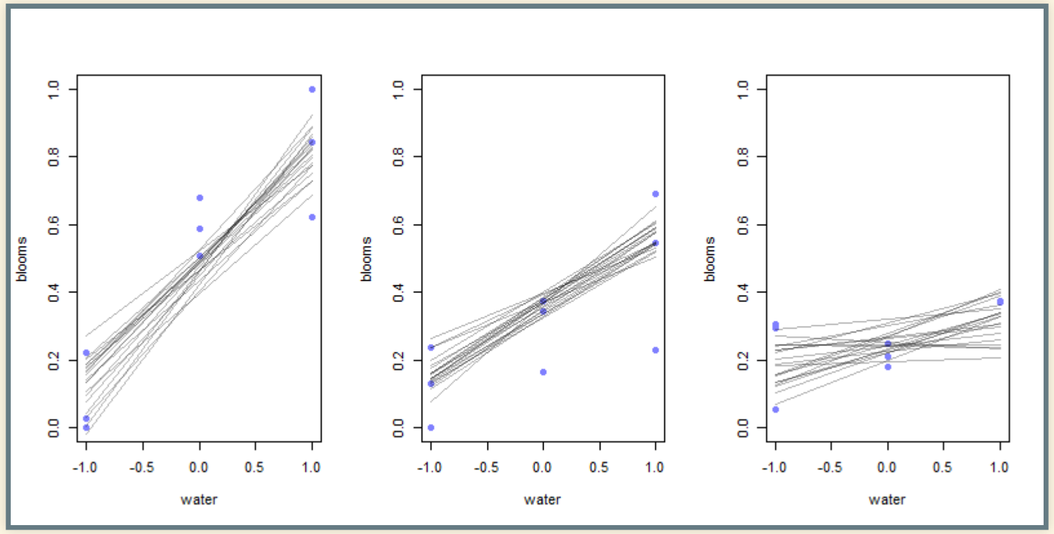


Fitted Model

```
mdl.4 <- quap(
  alist(
    blooms_std ~ dnorm(mu, sigma),
    mu <- a + bw * water_cent - bs * shade_cent -
      bws * water_cent * shade_cent,
    a ~ dnorm(0.25, 0.35),
    bw ~ dlnorm(-2, 1),
    bs ~ dlnorm(-2, 1),
    bws ~ dlnorm(-2, 1),
    sigma ~ dexp(1)
    ), data = d)

par(mfrow=c(1,3)) # 3 plots in 1 row

for (s in -1:1){
  idx <- which(d$shade_cent == s)
  plot( d$water_cent[idx], d$blooms_std[idx],
        xlim=c(-1,1), ylim=c(0,1),
        xlab = "water", ylab = "blooms",
        pch = 16, col = rangi2)
  mu <- link(mdl.4, data = data.frame(shade_cent = s,
                                      water_cent = -1:1))
  for (i in 1:20)
    lines(-1:1, mu[i,], col = col.alpha("black", 0.3))
}
```



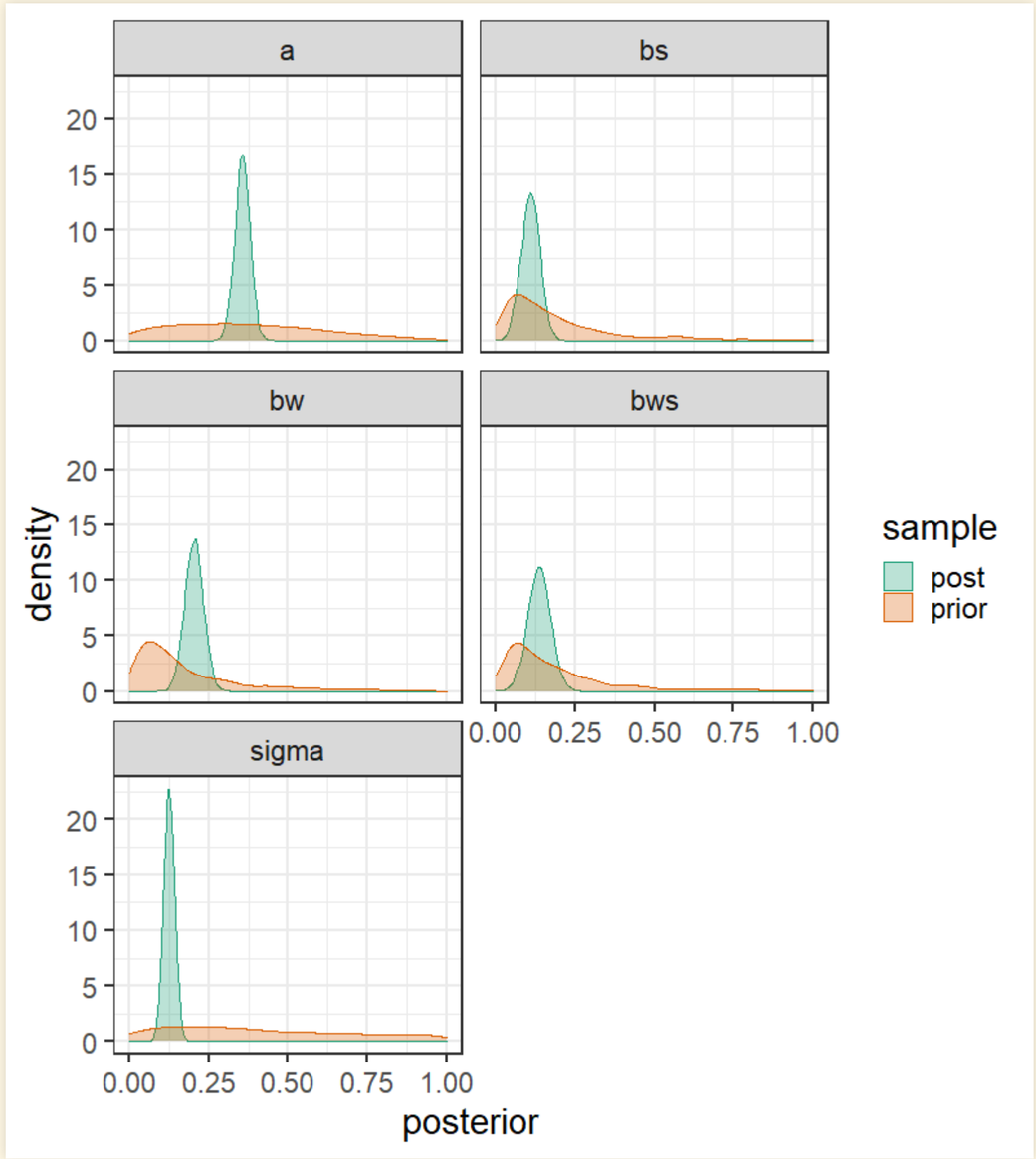
Inspect posteriors

```
library(tidybayes)
library(tidybayes.rethinking)

prior <- extract.prior(mdl.4) %>%
  imap_dfr(~tibble(.variable = .y, .value = .x,
                   .draw = seq_along(.x), sample =
        "prior"))

posterior <- tidy_draws(mdl.4) %>% gather_variables() %>%
  mutate(sample = "post") %>%
  bind_rows(prior)

ggplot(posterior, aes(x = .value, color = sample,
                      fill = sample)) +
  geom_density(alpha = 0.3) +
  xlim(0,1) +
  facet_wrap(~.variable, ncol = 2) +
  labs(x = "posterior", y = "density") +
  scale_color_brewer(palette = "Dark2") +
  scale_fill_brewer(palette = "Dark2")
```



Alternate Model Specification

```
mdl.4b <- quap(
  alist(
    blooms_std ~ dnorm(mu, sigma),
    mu <- a + bw * water_cent + bs * shade_cent +
      bws * water_cent * shade_cent,
    # New links
    bs <- -bs_m,
    bws <- -bws_m,
    a ~ dnorm(0.25, 0.35),
    bw ~ dlnorm(-2, 1),
    bs_m ~ dlnorm(-2, 1),
    bws_m ~ dlnorm(-2, 1),
    sigma ~ dexp(1)
    ), data = d)
par(mfrow=c(1,3)) # 3 plots in 1 row
for (s in -1:1){
  idx <- which(d$shade_cent == s)
  plot( d$water_cent[idx], d$blooms_std[idx],
        xlim=c(-1,1), ylim=c(0,1),
        xlab = "water", ylab = "blooms",
        pch = 16, col = rangi2)
  mu <- link(mdl.4b, data = data.frame(shade_cent = s,
                                       water_cent = -1:1))
  # There are 3 links, so link() returns a list with
  # mu, bs, and bws. We only want mu.
  mu <- mu$mu
  for (i in 1:20)
    lines(-1:1, mu[i,], col = col.alpha("black", 0.3))
}
```