Community context mediates effects of pollinator loss on seed production

K. C. Arrowsmith, V. A. Reynolds, H. M Briggs, and B. J. Brosi

1 Overview

This document contains all code necessary to replicate the analyses for the above article. Any questions about this project can be directed to Kaysee Arrowsmith at kcarrows@uw.edu.

1.1 Particulars

The data for this analysis come from an experimental bumble bee removal experiment that took place over three years (2011, 2013, and 2014; in 2012, there was a severe drought and we were not able to conduct the experiment) across 14 replicates. Each site had two experimental states (control and manipulation). At each site, we selected focal *Delphinium barbeyi* individuals that were identified with ID numbers (`delph.plant.num`). We further selected specific flowers on that focal individual (`delph.flower.num`) for which we collected and counted seeds. *D. barbeyi* flowers typically produce three carpels, so each selected *D. barbeyi* flower may appear in the dataset up to three times, with each row presenting a separate count of viable seeds produced by that carpel. For each site/state combination, we conducted a pollinator survey, which is summarized in this dataset with *Bombus* richness (`bombus.rich`), abundance (`bombus.abund`, scaled to `bomabund.scale`), and mean fidelity (`mean.fidelity`). The identities of these bees were also considered to determine the relative abundance of long-tongued bees (`prop.long`). For manipulation surveys only, we noted the species of *Bombus* removed (`species.removed`, always the most abundant species from that day's survey) and the number and relative abundance of that species (`num.removed`, `prop.removed`). Once per site (during the control period), we performed a floral survey in which is summarized in these data with the relative abundance of *D. barbeyi* (`prop.delph`) and the similarity between the rest of the floral community and *D. barbeyi* on the morphological axes of color (`color.sim`) and corolla length (`corolla.sim`).

2 Load Packages, Functions, Data

```
library(tidyverse)
library(kableExtra)
library(broom.mixed)
library(glmmTMB)
library(MuMIn)
library(xtable)
library(DHARMa)
library(performance)
```

```
# Function to test for overdispersion (credit Ben Bolker)
overdisp_fun <- function(model) {
    rdf <- df.residual(model)
    rp <- residuals(model,type="pearson")
    Pearson.chisq <- sum(rp^2)
    prat <- Pearson.chisq/rdf
```

```
    pval <- pchisq(Pearson.chisq, df=rdf, lower.tail=FALSE)
    c(chisq=Pearson.chisq,ratio=prat,rdf=rdf,p=pval)
}


# Function to calculate confidence intervals
## https://stackoverflow.com/questions/48612153/how-to-calculate-confidence-intervals-for-a-vector
confidence_interval <- function(vector, interval) {
  # Standard deviation of sample
  vec_sd <- sd(vector)
  # Sample size
  n <- length(vector)
  # Mean of sample
  vec_mean <- mean(vector)
  # Error according to t distribution
  error <- qt((interval + 1)/2, df = n - 1) * vec_sd / sqrt(n)
  # Confidence interval as a vector
  result <- c("lower" = vec_mean - error, "upper" = vec_mean + error)
  return(result)
}
```

```
# Clean data
dat <- read.csv("RMBLseeds-cleaned.csv", stringsAsFactors = T)
```

3 Check Model Assumptions

We start with a global model using a Poisson distribution.

```
global.mod <- glmmTMB(viable ~
                        prop.removed +
                        bomabund.scale +
                        prop.long +
                        mean.fidelity +
                        prop.delph +
                        color.sim +
                        corolla.sim +
                        state:bomabund.scale +
                        state:prop.long +
                        state:mean.fidelity +
                        (1|site/delph.plant.num/delph.flower.num) +
                        (1|year),
                      data = dat,
                      family = poisson)
```

3.1 Overdispersion

We use a function from Ben Bolker to test for overdispersion in our model.

```
# Test for overdispersion
overdisp_fun(global.mod)
```

```
##         chisq         ratio           rdf             p
##  3.118717e+03  1.997897e+00  1.561000e+03 3.146209e-106
```

Significant overdispersion detected (p « 0.05). To account for overdispersion, we switch from a Poisson

distribution to a negative binomial.

```
nbinom.mod <- glmmTMB(viable ~
                        prop.removed +
                        bomabund.scale +
                        prop.long +
                        mean.fidelity +
                        prop.delph +
                        color.sim +
                        corolla.sim +
                        state:bomabund.scale +
                        state:prop.long +
                        state:mean.fidelity +
                        (1|site/delph.plant.num/delph.flower.num) +
                        (1|year),
                      data = dat,
                      family = nbinom2)

# Zero-inflated model
nbinom.zi <- glmmTMB(viable ~
                        prop.removed +
                        bomabund.scale +
                        prop.long +
                        mean.fidelity +
                        prop.delph +
                        color.sim +
                        corolla.sim +
                        state:bomabund.scale +
                        state:prop.long +
                        state:mean.fidelity +
                        (1|site/delph.plant.num/delph.flower.num) +
                        (1|year),
                      data = dat,
                      family = nbinom2,
                     zi = ~.)
```

3.2 Zero-inflation

To test for zero-inflation in our data, we use two different methods. First, we perform a chi-squared test to see if there are more zeros than expected. Then, we use an ANOVA to compare global models with and without controlling for zero-inflation.

```
# Chi-square test for zero-inflation
contingency <- dat %>%
  mutate(value.cut = cut(viable, breaks = c(-1, 0, 50), labels = c("0", ">0"))) %>%
  with(table(value.cut, state, useNA = "ifany"))
chisq.test(contingency, simulate.p.value = T)
```

```
##
##  Pearson's Chi-squared test with simulated p-value (based on 2000
##  replicates)
##
## data:  contingency
## X-squared = 5.8251, df = NA, p-value = 0.06247
```

```
# Likelihood Ratio
anova(nbinom.mod, nbinom.zi)
```

```
## Data: dat
## Models:
## nbinom.mod: viable ~ prop.removed + bomabund.scale + prop.long + mean.fidelity + , zi=~0, disp=~1
## nbinom.mod:     prop.delph + color.sim + corolla.sim + state:bomabund.scale + , zi=~., disp=~1
## nbinom.mod:     state:prop.long + state:mean.fidelity + (1 | site/delph.plant.num/delph.flower.num) +
## nbinom.mod:     (1 | year), zi=~., disp=~1
## nbinom.zi: viable ~ prop.removed + bomabund.scale + prop.long + mean.fidelity + , zi=~0, disp=~1
## nbinom.zi:     prop.delph + color.sim + corolla.sim + state:bomabund.scale + , zi=~., disp=~1
## nbinom.zi:     state:prop.long + state:mean.fidelity + (1 | site/delph.plant.num/delph.flower.num) +
## nbinom.zi:     (1 | year), zi=~., disp=~1
##            Df    AIC    BIC  logLik deviance Chisq Chi Df Pr(>Chisq)
## nbinom.mod 16 6076.4 6162.2 -3022.2   6044.4
## nbinom.zi  31 5448.6 5614.8 -2693.3   5386.6 657.8     15  < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Chi-squared test is marginally significant ($p \sim 0.05$). AIC is lower for zero-inflated model, and ANOVA is statistically significant ($p \ll 0.05$). All of this tells us that our data are likely to be zero-inflated and we will therefore use a zero-inflated GLMM.

3.3 Collinearity

Next, we check for collinearity between these variables to ensure that they can all be used in model selection without problems.

```
check_collinearity(nbinom.zi)
```

```
## # Check for Multicollinearity
##
## * conditional component:
##
## Low Correlation
##
##            Term  VIF Increased SE Tolerance
##  bomabund.scale 1.54         1.24      0.65
##       prop.long 1.69         1.30      0.59
##   mean.fidelity 3.50         1.87      0.29
##      prop.delph 1.45         1.20      0.69
##       color.sim 1.67         1.29      0.60
##     corolla.sim 1.39         1.18      0.72
##
## Moderate Correlation
##
##                  Term  VIF Increased SE Tolerance
##  bomabund.scale:state 5.77         2.40      0.17
##
## High Correlation
##
##                 Term   VIF Increased SE Tolerance
##         prop.removed 40.85         6.39      0.02
##      prop.long:state 75.90         8.71      0.01
##  mean.fidelity:state 92.10         9.60      0.01
```

```
##
## * zero inflated component:
##
## Low Correlation
##
##            Term  VIF Increased SE Tolerance
##  bomabund.scale 1.49         1.22      0.67
##       prop.long 1.75         1.32      0.57
##   mean.fidelity 2.50         1.58      0.40
##      prop.delph 1.33         1.15      0.75
##       color.sim 1.73         1.32      0.58
##     corolla.sim 1.54         1.24      0.65
##
## Moderate Correlation
##
##                   Term  VIF Increased SE Tolerance
##  bomabund.scale:state 5.32         2.31      0.19
##
## High Correlation
##
##                 Term    VIF Increased SE Tolerance
##          prop.removed  37.42         6.12      0.03
##      prop.long:state  68.60         8.28      0.01
##  mean.fidelity:state 105.03        10.25      0.01
```

```
# check_collinearity(no.int)
```

We see moderate to high correlation between all of the variables involving the manipulation (`prop.removed` and all interaction terms), which is expected. Importantly, we see low correlation with all of the additive variables that do not involve the strength of the manipulation.

4 Model Selection

We use a model selection approach to determine which fixed effects are most important in predicting the number of viable seeds produced by *D. barbeyi*. Because the data was found to be overdispersed, we use the `glmmTMB` package with `family = nbinom2`.

```
# Creating all combinations of fixed effects and pasting them into a formula
vars <- c("viable",
          "prop.removed",
          "bomabund.scale",
          "prop.delph",
          "prop.long",
          "color.sim",
          "corolla.sim",
          "mean.fidelity",
          "state:bomabund.scale",
          "state:prop.long",
          "state:mean.fidelity"
          )
N <- as.list(seq(1:(length(vars)-1)))
COMB <- sapply(N, function(m) combn(x=vars[2:length(vars)], m))
COMB2 <- list()
k=0
for(i in seq(COMB)){
```

```
    tmp <- COMB[[i]]
    for(j in seq(ncol(tmp))){
        k <- k + 1
        COMB2[[k]] <- formula(paste(
          "viable",
          "~",
          paste(tmp[,j], collapse=" + "),
          "+ (1|site/delph.plant.num/delph.flower.num) + (1|year)"))
    }
}

# Running glmmTMB for each formula and using glance to isolate AIC values
res <- vector(mode = "list", length(COMB2))
suppressWarnings(for(i in seq(COMB2)){
    res[[i]] <- glance(glmmTMB(COMB2[[i]], data=dat, family = nbinom2, zi = ~.))
})

# Add Model ID column to each tibble
ID <- c(1:length(res))
res2 <- mapply(cbind, res, "Model" = ID, SIMPLIFY = F)

# Removing models that failed to converge)
filt <- Filter(function(x) length(x) > 4, res2)

# Extracting AIC
filt.df <- data.frame(matrix(unlist(filt), nrow = length(filt), byrow = T))
filt.df[,c('X1', 'X2', 'X4', 'X5')] <- list(NULL)
names(filt.df)[names(filt.df) == 'X3'] <- 'AIC'
names(filt.df)[names(filt.df) == 'X6'] <- 'Model'

# Arranging AICs in increasing order
filt.df <- arrange(filt.df, AIC)
filt.df$delta <- filt.df$AIC - filt.df[1,1]

# Filter out only model outputs with delta < 2
filt.2 <- filt.df[filt.df$delta < 2,]

# Taking the Model IDs from the "best" models (delta < 2) and connecting them with the actual formulas
forms <- data.frame(matrix(unlist(COMB2), nrow = length(COMB2), byrow = T))
colnames(forms) <- "Formula"

forms$Model <- c(1:length(res))
forms$Model <- as.numeric(forms$Model)

mods <- merge(filt.2, forms, by = "Model")
mods <- arrange(mods, AIC)
mods$Formula <- as.character(mods$Formula)

mods %>%
  kable %>%
  kable_styling("striped", full_width = T)
```

Model	AIC	delta	Formula
971	5448.165	0.0000000	viable ~ prop.removed + bomabund.scale + prop.delph + prop.long + color.sim + corolla.sim + state:bomabund.scale + state:prop.long + (1 | site/delph.plant.num/delph.flower.num) + (1 | year)
731	5448.165	0.0000000	viable ~ prop.removed + prop.delph + color.sim + corolla.sim + state:bomabund.scale + state:prop.long + (1 | site/delph.plant.num/delph.flower.num) + (1 | year)
907	5448.165	0.0000000	viable ~ prop.removed + prop.delph + prop.long + color.sim + corolla.sim + state:bomabund.scale + state:prop.long + (1 | site/delph.plant.num/delph.flower.num) + (1 | year)
871	5448.165	0.0000001	viable ~ prop.removed + bomabund.scale + prop.delph + color.sim + corolla.sim + state:bomabund.scale + state:prop.long + (1 | site/delph.plant.num/delph.flower.num) + (1 | year)
983	5448.266	0.1011210	viable ~ prop.removed + bomabund.scale + prop.delph + color.sim + corolla.sim + mean.fidelity + state:bomabund.scale + state:prop.long + (1 | site/delph.plant.num/delph.flower.num) + (1 | year)
1013	5448.266	0.1011211	viable ~ prop.removed + bomabund.scale + prop.delph + prop.long + color.sim + corolla.sim + mean.fidelity + state:bomabund.scale + state:prop.long + (1 | site/delph.plant.num/delph.flower.num) + (1 | year)
919	5448.266	0.1011211	viable ~ prop.removed + prop.delph + color.sim + corolla.sim + mean.fidelity + state:bomabund.scale + state:prop.long + (1 | site/delph.plant.num/delph.flower.num) + (1 | year)
996	5448.266	0.1011212	viable ~ prop.removed +

5 Model Averaging

Because we do not have an obvious "best" model after model selection, we average the top four models ($\Delta AIC < 10^{-7}$).

```
# Run all of the models that the model selection spit out
## Create a list with all of the model estimates extracted using tidy
## Filter only the zero-inflated estimates
top.mods <- vector(mode = "list", length(mods$Formula))
for(i in seq(mods$Formula)){
    top.mods[[i]] <-
      tidy(glmmTMB(as.formula(mods$Formula[[i]]), data=dat, family = nbinom2, zi = ~.)) %>%
      filter(component == "zi") %>%
      mutate(var = ifelse(term == "sd__(Intercept)", group, term)) %>%
      select(var, estimate) %>%
      pivot_wider(names_from = var, values_from = estimate)
}

# Turn the list of estimates for each model into a single dataframe
modselect.results <- data.frame()
temp.df <- data.frame()
for(i in seq(top.mods)){
  temp.df <- as.data.frame(unlist(top.mods[[i]]))
  colnames(temp.df) <- "estimate"
  temp.df$var <- rownames(temp.df)
  temp.wide <- pivot_wider(temp.df, names_from = var, values_from = estimate)
  modselect.results <- bind_rows(modselect.results, temp.wide)
}

# Add model number to this dataframe because otherwise these are all useless
modselect.results$Model <- mods$Model

# Reorganize this table and change variable naming convention
modselect.table <- modselect.results %>%
  dplyr::select(Model,
         `(Intercept)`,
         prop.removed,
         bomabund.scale,
         prop.long,
         mean.fidelity,
         prop.delph,
         color.sim,
         corolla.sim,
         `statecontrol:bomabund.scale`,
         `statemanipulation:bomabund.scale`,
         `bomabund.scale:statemanipulation`,
         `statecontrol:prop.long`,
         `statemanipulation:prop.long`,
         `prop.long:statemanipulation`,
         `statecontrol:mean.fidelity`,
         `statemanipulation:mean.fidelity`,
         `mean.fidelity:statemanipulation`,
         `delph.flower.num:delph.plant.num:site`,
         `delph.plant.num:site`,
         site,
```

```
        year) %>%
  mutate(`state:bomabund.scale` =
           ifelse(!is.na(`statemanipulation:bomabund.scale`) |
                    !is.na(`bomabund.scale:statemanipulation`), "+", ""),
         `state:prop.long` =
           ifelse(!is.na(`statemanipulation:prop.long`) |
                  !is.na(`prop.long:statemanipulation`), "+", ""),
         `state:mean.fidelity` =
           ifelse(!is.na(`statemanipulation:mean.fidelity`) |
                    !is.na(`mean.fidelity:statemanipulation`), "+", "")) %>%
  dplyr::select(Model,
         `(Intercept)`,
         prop.removed,
         bomabund.scale,
         prop.long,
         mean.fidelity,
         prop.delph,
         color.sim,
         corolla.sim,
         `state:bomabund.scale`,
         `state:prop.long`,
         `state:mean.fidelity`,
         `delph.flower.num:delph.plant.num:site`,
         `delph.plant.num:site`,
         site,
         year)

# Filter top models
tiptop <- mods %>%
  filter(delta < 10^-6)

form.list <- as.list(rep(NA, times = nrow(tiptop)))
for(i in 1:nrow(tiptop)) {
  form.list[[i]] = glmmTMB(as.formula(tiptop$Formula[i]), data = dat, family = poisson, zi = ~.)
}

# Model average
top.avg <- model.avg(form.list)

avg.table <- as.data.frame(summary(top.avg)[9]) %>%
  rownames_to_column(var = "variable") %>%
  filter(str_detect(variable, "zi")) %>%
  column_to_rownames(var = "variable")
avg.table %>%
  kable %>%
  kable_styling("striped", full_width = T)
```

	coefmat.full.Estimate	coefmat.full.Std..Error	coefmat.full.Adjusted.SE	coefmat.full.z.value	coefmat.full.Pr...z..
zi..Int..	3.1512996	1.2801815	1.2811819	2.4596817	0.0139060
zi.prop.removed.	-1.3574372	1.0677219	1.0685563	1.2703470	0.2039611
zi.bomabund.scale.	-0.0510678	0.1909921	0.1911307	0.2671877	0.7893247
zi.prop.delph.	0.4853748	0.8168358	0.8174742	0.5937494	0.5526797
zi.prop.long.	-2.0265971	2.3055270	2.3059367	0.8788607	0.3794768
zi.color.sim.	1.4807137	1.2997860	1.3008018	1.1383085	0.2549917
zi.corolla.sim.	1.8645933	0.5041129	0.5045069	3.6958728	0.0002191
zi.bomabund.scale.statemanipulation.	2.2588293	0.5226228	0.5230273	4.3187596	0.0000157
zi.prop.long.statemanipulation.	1.0244162	2.4209290	2.4214953	0.4230511	0.6722580
zi.bomabund.scale.statecontrol	-0.0510683	0.1909923	0.1911308	0.2671904	0.7893226
zi.bomabund.scale.statemanipulation..1	2.2588293	0.5226228	0.5230273	4.3187596	0.0000157
zi.prop.long.statecontrol	-2.0265983	2.3055280	2.3059377	0.8788608	0.3794767
zi.prop.long.statemanipulation..1	1.0244162	2.4209290	2.4214953	0.4230511	0.6722580

6 Effect of Abundance

We test whether our manipulation has a significant impact on *Bombus* abundance that could affect interpretation of results.

```
abund.test <- glmmTMB(bombus.abund ~ state + (1|site/delph.plant.num/delph.flower.num) + (1|year), data
summary(abund.test)
```

```
##  Family: poisson  ( log )
## Formula:
## bombus.abund ~ state + (1 | site/delph.plant.num/delph.flower.num) +
##     (1 | year)
## Data: dat
##
##      AIC      BIC   logLik deviance df.resid
##  12224.7  12257.7  -6106.4  12212.7     1794
##
## Random effects:
##
## Conditional model:
##  Groups                                Name        Variance  Std.Dev.
##  delph.flower.num:delph.plant.num:site (Intercept) 9.964e-09 9.982e-05
##  delph.plant.num:site                  (Intercept) 3.557e-02 1.886e-01
##  site                                  (Intercept) 2.560e-01 5.060e-01
##  year                                  (Intercept) 2.263e-02 1.504e-01
## Number of obs: 1800, groups:
## delph.flower.num:delph.plant.num:site, 641; delph.plant.num:site, 413; site, 14; year, 3
##
## Conditional model:
##                    Estimate Std. Error z value Pr(>|z|)
## (Intercept)        3.918388   0.161777  24.221   <2e-16 ***
## statemanipulation -0.001427   0.013731  -0.104    0.917
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The manipulation does not have a statistically significant effect on *Bombus* abundance ($p = 0.917$).

7 Visits to D. barbeyi

We explore how our manipulation influenced the relative abundance of pollinator visits to *D. barbeyi* compared to other flowers.

```
dat.site <- dat %>%
  group_by(site, state, year) %>%
  summarise(prop.removed = unique(prop.removed),
            bombus.abund = unique(bombus.abund),
            prop.long = unique(prop.long))

forage <- read.csv("../Data/forage-cleaned.csv", stringsAsFactors = F)
delph.dat <- read.csv("../Data/dbarbcounts.csv", stringsAsFactors = F)

forage2 <- forage %>%
  mutate(plant.simple = ifelse(plant.species == "Delphinium barbeyi", "Dbarb", "NotDbarb"),
         year = as.numeric(year)) %>%
  group_by(site, state, year, plant.simple) %>%
  summarise(total.visits = sum(num.indiv.visited)) %>%
  pivot_wider(names_from = "plant.simple",
              values_from = "total.visits") %>%
  left_join(dat.site, by = c("site", "state", "year")) %>%
  left_join(delph.dat, by = "site") %>%
    mutate(prop.dbvisit = Dbarb/(Dbarb + NotDbarb))

dbvisit.test <- glmmTMB(cbind(Dbarb, NotDbarb) ~
                          prop.removed +
                          bombus.abund +
                          prop.long +
                          num.delph +
                          prop.delph +
                          state:bombus.abund +
                          state:prop.long +
                          (1|site) +
                          (1|year),
                        data = forage2,
                        family = binomial)
summary(dbvisit.test)
```

```
##  Family: binomial  ( logit )
## Formula:
## cbind(Dbarb, NotDbarb) ~ prop.removed + bombus.abund + prop.long +
##     num.delph + prop.delph + state:bombus.abund + state:prop.long +
##     (1 | site) + (1 | year)
## Data: forage2
##
##      AIC      BIC   logLik deviance df.resid
##    779.7    792.2   -379.8    759.7       16
##
## Random effects:
##
## Conditional model:
##  Groups Name        Variance  Std.Dev.
##  site   (Intercept) 9.439e-01 0.9715308
##  year   (Intercept) 5.848e-08 0.0002418
```

```
## Number of obs: 26, groups:  site, 14; year, 3
##
## Conditional model:
##                                  Estimate Std. Error z value Pr(>|z|)
## (Intercept)                     0.5159050  0.5039611   1.024 0.305977
## prop.removed                    0.2837885  0.2234731   1.270 0.204120
## bombus.abund                   -0.0006522  0.0012438  -0.524 0.600053
## prop.long                       0.5533475  0.2889985   1.915 0.055530 .
## num.delph                      -0.0001215  0.0001263  -0.962 0.335988
## prop.delph                      0.9799356  1.5183157   0.645 0.518662
## bombus.abund:statemanipulation  0.0029489  0.0013551   2.176 0.029538 *
## prop.long:statemanipulation    -0.8813025  0.2269138  -3.884 0.000103 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

We find that our manipulation interacted with *Bombus* abundance and distribution of tongue lengths to significantly influence the proportion of *Bombus* foraging visits to *D. barbeyi*.

8 Figures

Code to produce all figures included in the manuscript.

8.1 Model Prediction

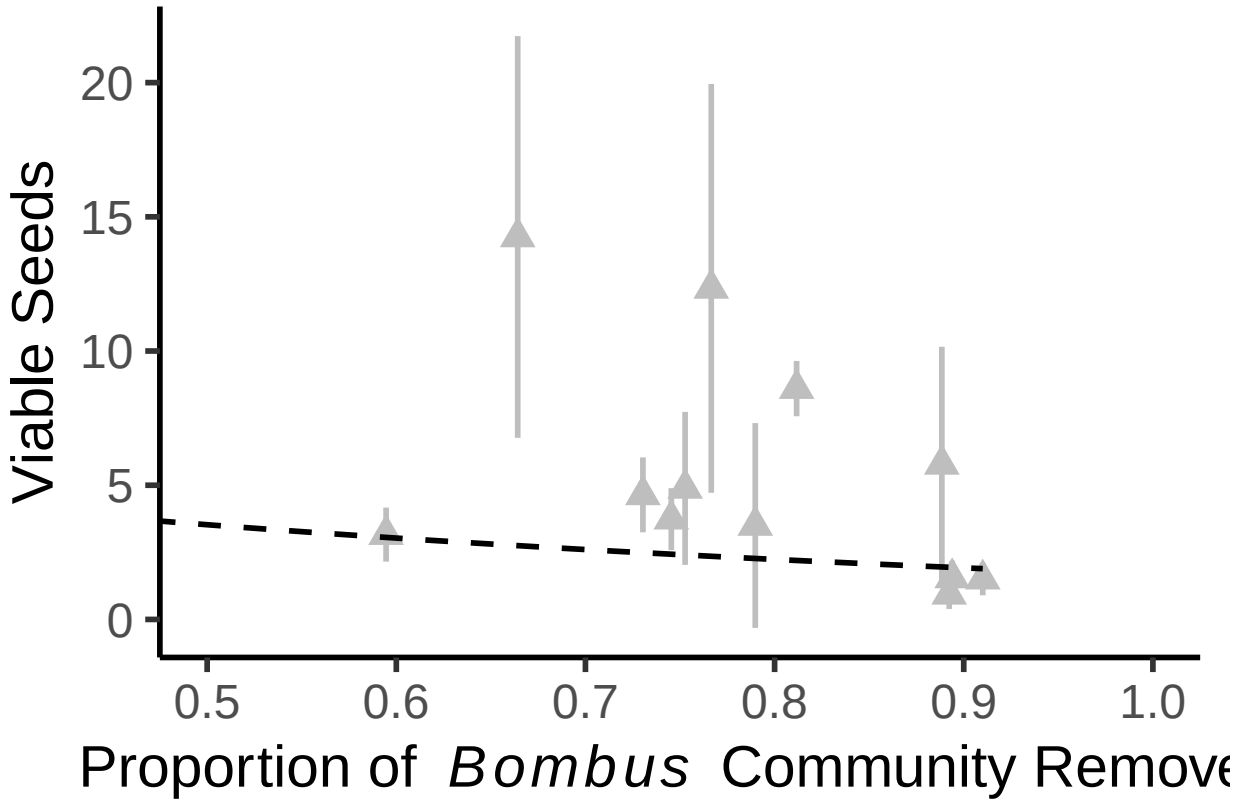
Generate model predictions so that our plots will show the trends found in our model selection and averaging.

```
pred.dat <- expand.grid(
    prop.removed = seq(from = min(dat$prop.removed, na.rm = T), to = max(dat$prop.removed, na.rm = T),
    bomabund.scale = seq(from = min(dat$bomabund.scale), to = max(dat$bomabund.scale), length.out = 4),
    viable = 0,
    state = c("control", "manipulation"),
    prop.delph = seq(from = min(dat$prop.delph), to = max(dat$prop.delph), length.out = 4),
    prop.long = seq(from = min(dat$prop.long), to = max(dat$prop.long), length.out = 4),
    mean.fidelity = seq(from = min(dat$mean.fidelity, na.rm = T), to = max(dat$mean.fidelity, na.rm = T
    color.sim = seq(from = min(dat$color.sim), to = max(dat$color.sim), length.out = 4),
    corolla.sim = seq(from = min(dat$corolla.sim), to = max(dat$corolla.sim), length.out = 4),
    site = c("site1", "site2"),
    delph.plant.num = rep(1:2),
    delph.flower.num = rep(1:2),
    year = c("year1", "year2")
)

# type = zlink should call the zero-inflated results
pred.dat$viable <- predict(top.avg, pred.dat, type =  "zlink", full = T, allow.new.levels = T)
```

8.2 Figure 1

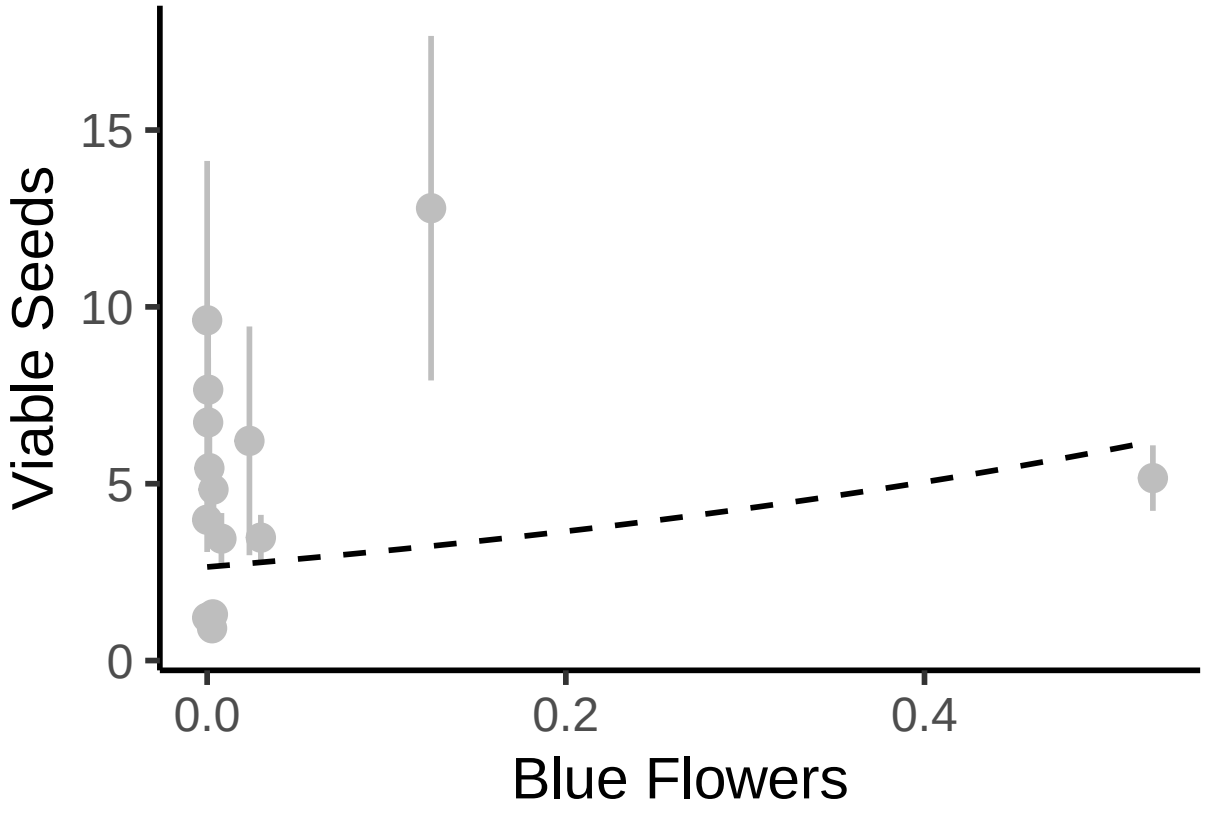
```
## Data for geom_pointrange
dat.manip <- dat[dat$state == "manipulation",]
manip.sum <- dat.manip %>%
  group_by(site, year, prop.removed, bombus.abund, mean.fidelity) %>%
  summarise(mean.viable = mean(viable, na.rm = T),
            lower.ci = confidence_interval(na.omit(viable), 0.95)[[1]],
            higher.ci = confidence_interval(na.omit(viable), 0.95)[[2]])
```

```r
## Data for geom_smooth
pred.manip <- pred.dat %>%
  filter(prop.removed > 0)

proprem.plot <- ggplot() +
  geom_pointrange(data = manip.sum,
                  aes(x = prop.removed,
                      y = mean.viable,
                      ymin = lower.ci,
                      ymax = higher.ci),
                  size = 1,
                  shape = 17,
                  color = "grey") +
  geom_smooth(data = pred.manip,
              aes(x = prop.removed,
                  y = as.integer(exp(viable))),
              method = "glm",
              method.args = list(family = poisson(link = "log")),
              col = "black",
              linetype = "dashed",
              se = F) +
  theme_classic(base_size = 22) +
  labs(x = "Proportion of "~italic(Bombus)~" Community Removed",
       y = "Viable Seeds") +
  coord_cartesian(xlim = c(0.5, 1))
proprem.plot
```

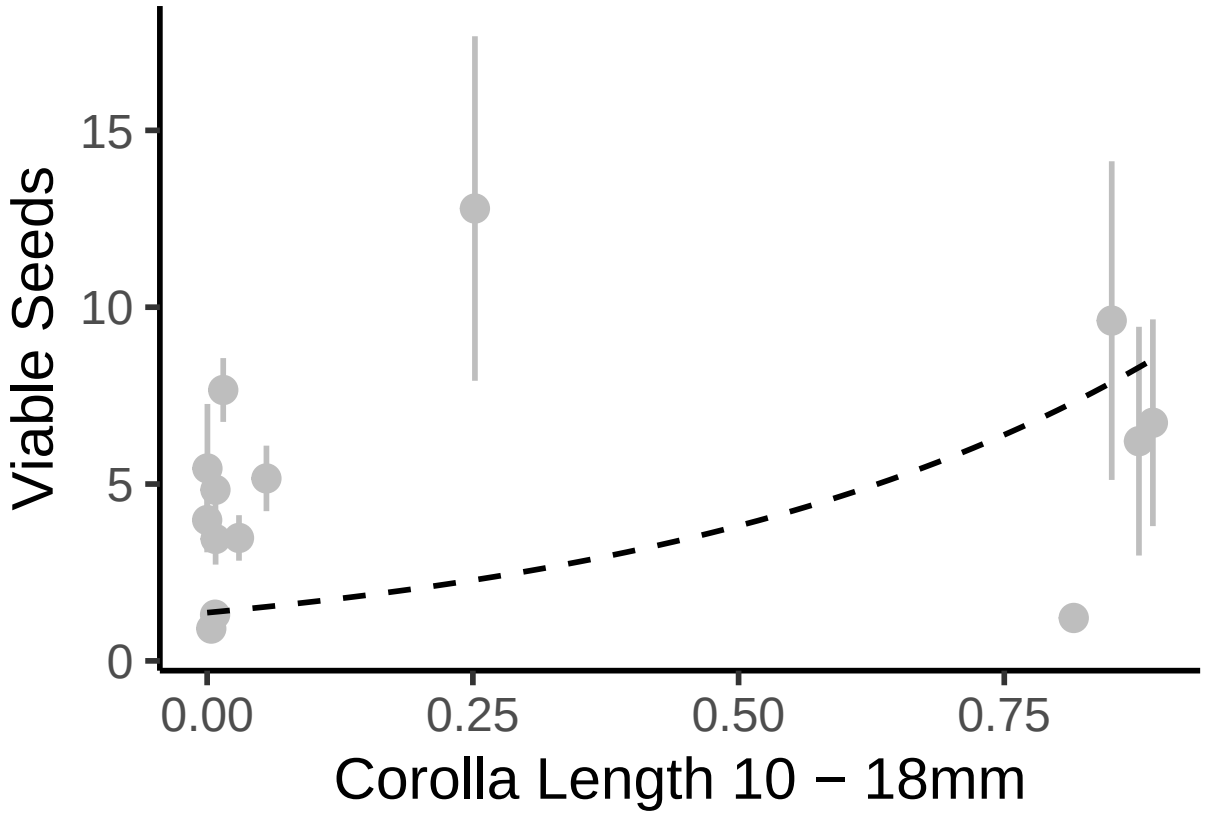


8.3 Figure 2

```
## Data for geom_pointrange
trait.sum <- dat %>%
  group_by(site, color.sim, corolla.sim, year) %>%
  summarise(mean.viable = mean(viable, na.rm = T),
            lower.ci = confidence_interval(na.omit(viable), 0.95)[[1]],
            higher.ci = confidence_interval(na.omit(viable), 0.95)[[2]])

color.plot <- ggplot() +
  geom_pointrange(data = trait.sum, aes(x = color.sim,
                                        y = mean.viable,
                                        ymax = higher.ci,
                                        ymin = lower.ci),
                  size = 1,
                  col = "grey") +
  geom_smooth(data = pred.dat,
              aes(x = color.sim, y = as.integer(exp(viable))),
              method = "glm",
              method.args = list(family = poisson(link = "log")),
              col = "black",
              linetype = "dashed",
              se = F) +
  theme_classic(base_size = 22) +
  labs(x = "Blue Flowers",
       y = "Viable Seeds")
color.plot
```



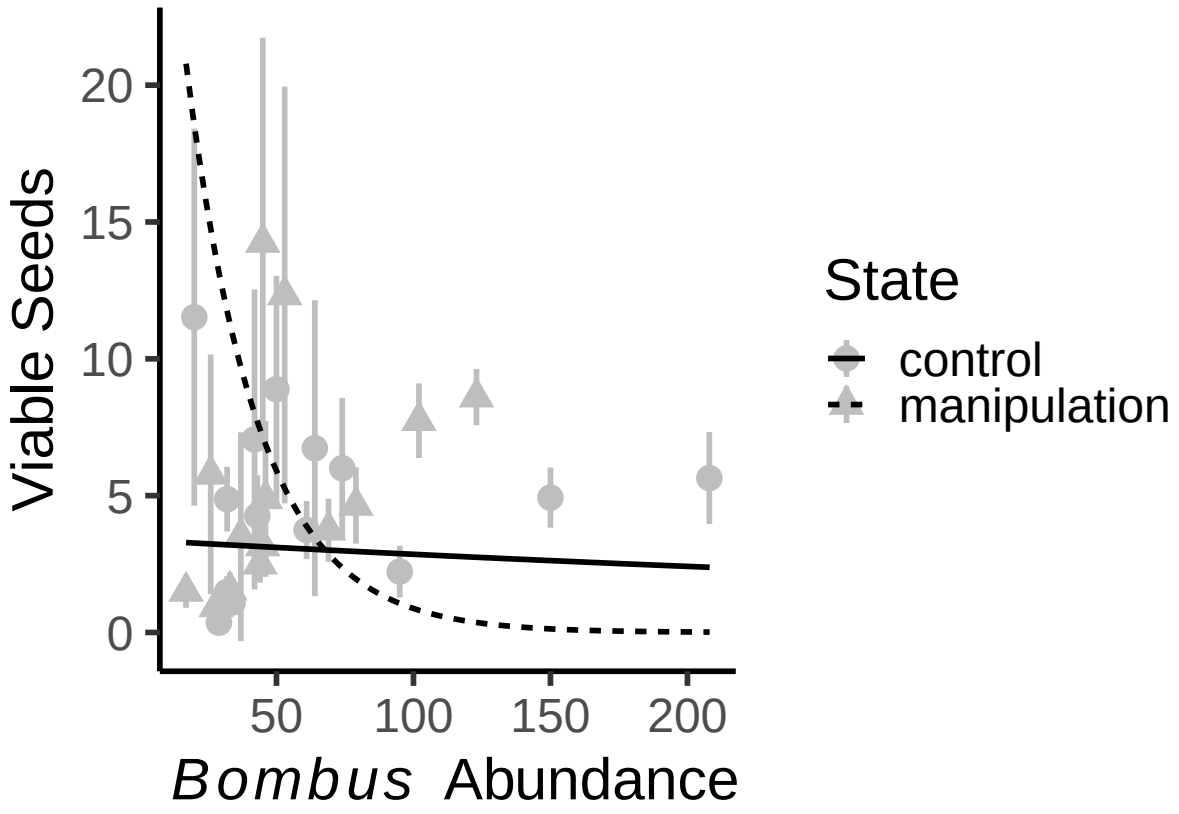
```
corolla.plot <- ggplot() +
  geom_pointrange(data = trait.sum, aes(x = corolla.sim,
                                        y = mean.viable,
                                        ymax = higher.ci,
                                        ymin = lower.ci),
                  size = 1,
                  col = "grey") +
  geom_smooth(data = pred.dat,
              aes(x = corolla.sim, y = as.integer(exp(viable))),
              method = "glm",
              method.args = list(family = poisson(link = "log")),
              col = "black",
              linetype = "dashed",
              se = F) +
  theme_classic(base_size = 22) +
  labs(x = "Corolla Length 10 - 18mm",
       y = "Viable Seeds")
corolla.plot
```



8.4 Figure 3

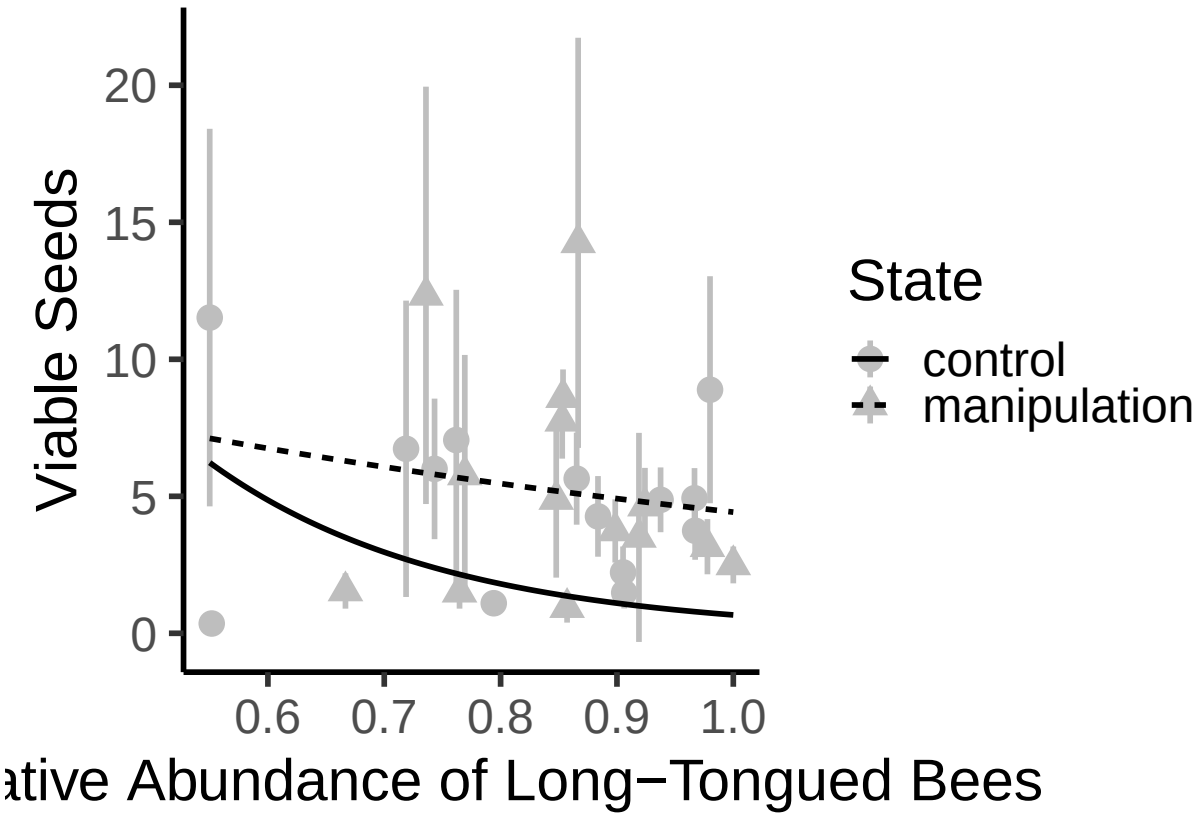
```
# to unscale bomabund.scale, multiply by scaling factor and then add center
# look at str(dat) to find that
# scale = 70.5, center = 3
pred.dat <- pred.dat %>%
  mutate(bombus.abund = (bomabund.scale * 70.5) + 3)
```

```r
# Data for geom_pointrange
dat.sum <- dat %>%
  group_by(site, state, bombus.abund, prop.long) %>%
  summarise(mean.viable = mean(viable, na.rm = T),
            lower.ci = confidence_interval(na.omit(viable), 0.95)[[1]],
            higher.ci = confidence_interval(na.omit(viable), 0.95)[[2]])

state.abund <- ggplot() +
  geom_pointrange(data = dat.sum, aes(x = bombus.abund,
                                      y = mean.viable,
                                      ymax = higher.ci,
                                      ymin = lower.ci,
                                      shape = state),
                  size = 1,
                  color = "grey") +
  geom_smooth(data = pred.dat,
              aes(x = bombus.abund,
                  y = as.integer(exp(viable)),
                  linetype = state),
              method = "glm",
              method.args = list(family = poisson(link = "log")),
              col = "black",
              se = F) +
  labs(x = ~italic(Bombus)~" Abundance",
       y = "Viable Seeds",
       shape = "State",
       linetype = "State") +
  theme_classic(base_size = 22)
state.abund
```



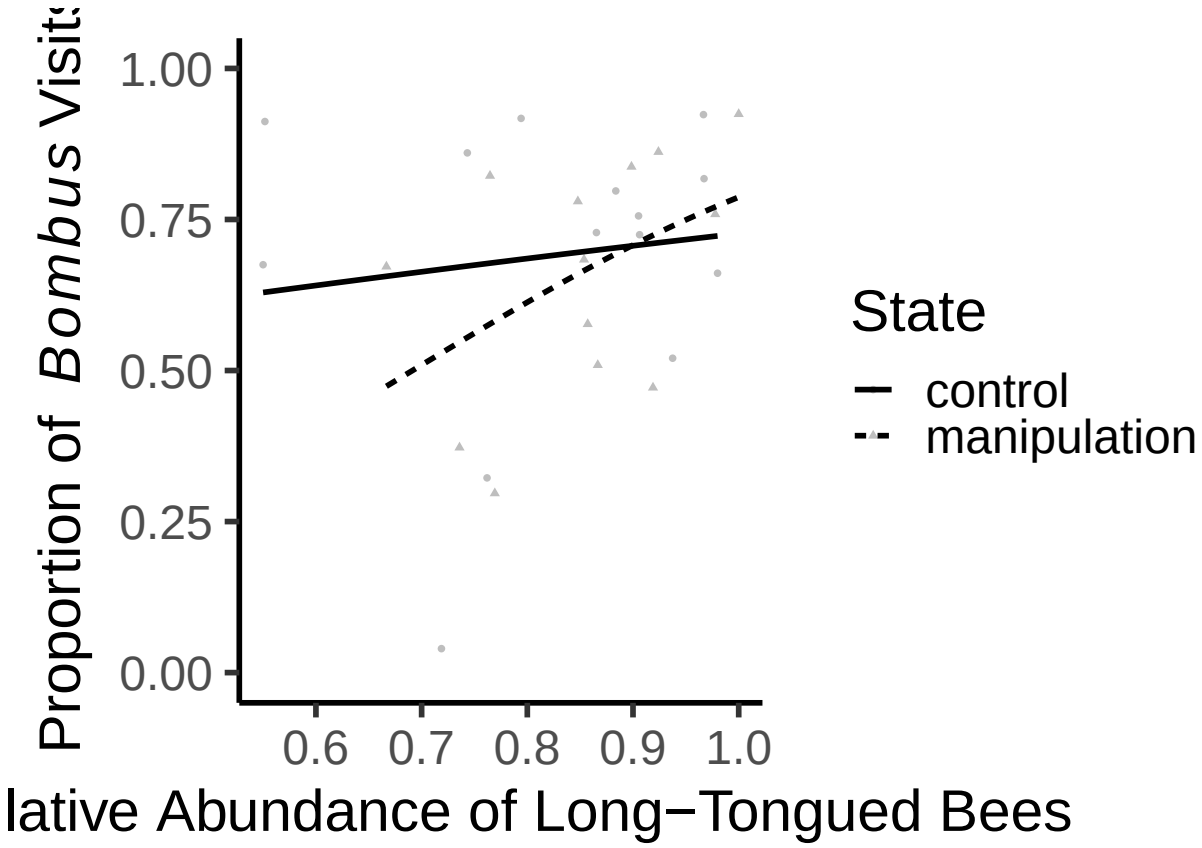
8.5 Figure 4

```
state.tongue <- ggplot() +
  geom_pointrange(data = dat.sum, aes(x = prop.long,
                                      y = mean.viable,
                                      ymax = higher.ci,
                                      ymin = lower.ci,
                                      shape = state),
                  size = 1,
                  color = "grey") +
  geom_smooth(data = pred.dat,
              aes(x = prop.long,
                  y = as.integer(exp(viable)),
                  linetype = state),
              method = "glm",
              method.args = list(family = poisson(link = "log")),
              col = "black",
              se = F) +
  labs(x = "Relative Abundance of Long-Tongued Bees",
       y = "Viable Seeds",
       shape = "State",
       linetype = "State") +
  theme_classic(base_size = 22)
state.tongue
```

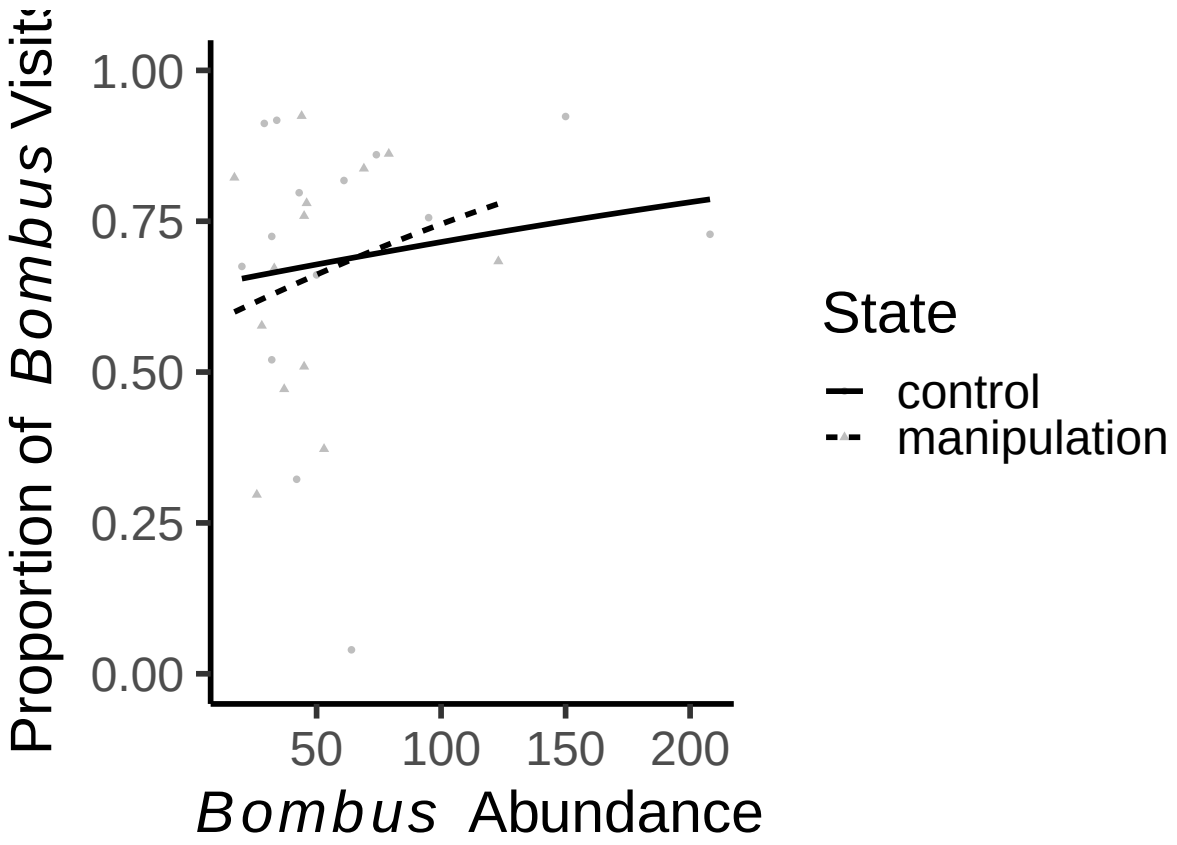


8.6 Figure 5

```
propdbtongue.plot <- ggplot(forage2, aes(x = prop.long,
                                         y = prop.dbvisit,
                                         shape = state,
                                         linetype = state)) +
  geom_point(size = 1,
             color = "grey") +
  geom_smooth(method = "glm",
              method.args = list(family = binomial()),
              col = "black",
              se = F) +
  coord_cartesian(ylim = c(0, 1)) +
  labs(x = "Relative Abundance of Long-Tongued Bees",
       y = "Proportion of "~italic(Bombus)~"Visits",
       shape = "State",
       linetype = "State") +
  theme_classic(base_size = 22)
propdbtongue.plot
```




```
propdbabund.plot2 <- ggplot(forage2, aes(x = bombus.abund,
                                         y = prop.dbvisit,
                                         shape = state,
                                         linetype = state)) +
  geom_point(size = 1,
             color = "grey") +
  geom_smooth(method = "glm",
              method.args = list(family = binomial()),
              col = "black",
              se = F) +
  coord_cartesian(ylim = c(0, 1)) +
  labs(x = ~italic(Bombus)~" Abundance",
       y = "Proportion of "~italic(Bombus)~"Visits",
       shape = "State",
       linetype = "State") +
  theme_classic(base_size = 22)
propdbabund.plot2
```



9 Session Info

```
sessionInfo()
```

```
## R version 4.1.2 (2021-11-01)
## Platform: x86_64-apple-darwin17.0 (64-bit)
## Running under: macOS Big Sur 10.16
##
## Matrix products: default
## BLAS:   /Library/Frameworks/R.framework/Versions/4.1/Resources/lib/libRblas.0.dylib
## LAPACK: /Library/Frameworks/R.framework/Versions/4.1/Resources/lib/libRlapack.dylib
##
## locale:
## [1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
##  [1] performance_0.8.0 DHARMa_0.4.4      xtable_1.8-4      MuMIn_1.43.17
##  [5] glmmTMB_1.1.3     broom.mixed_0.2.7 kableExtra_1.3.4  forcats_0.5.1
##  [9] stringr_1.4.0     dplyr_1.0.7       purrr_0.3.4       readr_2.1.2
## [13] tidyr_1.1.4       tibble_3.1.5      ggplot2_3.3.5     tidyverse_1.3.1
##
## loaded via a namespace (and not attached):
##  [1] nlme_3.1-153        fs_1.5.0            lubridate_1.8.0
```

```
##  [4] insight_0.14.5      webshot_0.5.2       httr_1.4.2
##  [7] numDeriv_2016.8-1.1 tools_4.1.2         TMB_1.7.22
## [10] backports_1.3.0     utf8_1.2.2          R6_2.5.1
## [13] mgcv_1.8-38         DBI_1.1.1           colorspace_2.0-2
## [16] withr_2.5.0         tidyselect_1.1.1    emmeans_1.7.4-1
## [19] compiler_4.1.2      cli_3.2.0           rvest_1.0.2
## [22] xml2_1.3.2          sandwich_3.0-1      labeling_0.4.2
## [25] scales_1.1.1        mvtnorm_1.1-3       systemfonts_1.0.3
## [28] digest_0.6.28       minqa_1.2.4         rmarkdown_2.11
## [31] svglite_2.0.0       pkgconfig_2.0.3     htmltools_0.5.2
## [34] lme4_1.1-27.1       highr_0.9           dbplyr_2.1.1
## [37] fastmap_1.1.0       rlang_1.0.2         readxl_1.3.1
## [40] rstudioapi_0.13     farver_2.1.0        generics_0.1.1
## [43] zoo_1.8-9           jsonlite_1.7.2      magrittr_2.0.1
## [46] Matrix_1.3-4        Rcpp_1.0.7          munsell_0.5.0
## [49] fansi_0.5.0         lifecycle_1.0.1     stringi_1.7.5
## [52] multcomp_1.4-19     yaml_2.2.1          MASS_7.3-54
## [55] grid_4.1.2          crayon_1.4.1        lattice_0.20-45
## [58] haven_2.4.3         splines_4.1.2       hms_1.1.1
## [61] knitr_1.36          pillar_1.6.4        boot_1.3-28
## [64] estimability_1.3    codetools_0.2-18    stats4_4.1.2
## [67] reprex_2.0.1        glue_1.6.2          evaluate_0.14
## [70] modelr_0.1.8        vctrs_0.4.1         nloptr_1.2.2.3
## [73] tzdb_0.2.0          cellranger_1.1.0    gtable_0.3.0
## [76] assertthat_0.2.1    xfun_0.27           broom_0.7.9
## [79] coda_0.19-4         survival_3.2-13     viridisLite_0.4.0
## [82] TH.data_1.1-1       ellipsis_0.3.2
```