Fitness of *Penstemon digitalis* increases with floral scent emission

BIOS15 Exam 2025-26

Oliver E. Todreas

2026-01-14

Introduction

Floral scent is often thought to be an important part of floral phenotypes, but is rarely studied (TODO: cite). Our hypothesis is that higher floral scent emission results in higher fitness. This hypothesis is formed by the assumption that floral scent can help guide pollinators to the flowers, thus increasing the likelihood that offspring are successfully produced. To test this hypothesis, we analyze data collected in a garden study on a variety of phenotypic variables of *Penstemon digitalis*. Specifically, we analyze the effect of floral scent emission on a fitness variable computed by the data collectors. We include the population of origin from which a given individual came and account for the effect of experimental block.

Methods

The data used contain a variety of phenotypic data for *P. digitalis* individuals. Each data point represents one plant. The floral scent emission was measured per plant in $ng/L/h$, while fitness was defined as number of fruits times fruit mass in mg. Both floral scent emission and fitness were natural log-transformed to remove right skew in the data, shown in the model description below. As a result, one individual, whose log of floral scent emission was undefined, was dropped. Furthermore, each plant was sourced from one of three populations in the north-eastern USA, abbreviated NR, WF, and TH. Plants were arranged in a randomized complete block design, where 35 blocks each contained one plant from each of the three populations. To determine the effect of floral scent emission on fitness for individuals from each of the three source populations while accounting for the random effect of experimental block, the following generalized linear mixed model with random intercepts was fitted to the data, shown here in R syntax.

```
log(fitness) ~ log(floral scent emission) + population + (1|block)
```

Results

The whole model (both fixed and random effects) fitted to the data explained 26.0% of the variance in fitness across the dataset (pseudo $r^2 = 0.260$). 18.6% of the variance in fitness was explained by the experimental block (Table 1). Fitness was predicted to increased by 0.155% for each 1% increase in total scent emission ($elasticity = 0.155 \pm 0.051\ mg/ng/L/h$, Table 2, Figure 1). At any given total scent emission, population WF had the highest predicted fitness, followed by population NR and TH, in descending order (Table 2, Figure 1). On the data scale, this meant that at a floral scent emission of 1.72 $ng/L/h$ (the bounds of the first quartile), the predicted fitness was 68.5, 70.6, and 73.5 mg for populations TH, NR, and WF respectively. At a floral scent emission of 7.80 $ng/L/h$ (the bounds of the third quartile), the predicted fitness was 86.7, 89.2, and 92.9 mg for populations TH, NR, and WF respectively. In other words, an increase in fitness of 18.1, 18.7, and 19.4 mg was predicted by the model across the interquartile range of total floral scent.

	mpg	cyl	disp	hp	drat
Mazda RX4	21.0	6	160	110	3.90
Mazda RX4 Wag	21.0	6	160	110	3.90
Datsun 710	22.8	4	108	93	3.85
Hornet 4 Drive	21.4	6	258	110	3.08
Hornet Sportabout	18.7	8	360	175	3.15

TODO: Table 1, random effects TODO: Table 2, model parameter estimates

Conclusions

The analysis presented in this report supports the hypothesis that increased floral scent emission increases fitness of *P. digitalis*. The increase was small but noticable at 0.15% increase in fitness per 1% increase in total scent emission. We found that the populations were relatively similar in how total scent emission affected fitness, but that fitness was generally the fighest for WF and the lowest for TH. We also found that a fair share of variance was represented by the experimental blocks, suggesting that the design captured the variance within groups to a large extent. In conclusion, total scent emission had a small but noticable effect on the fitness of plants, and populations were generally quite similar in regards to this relationship.

Appendix

The repository for this project can be found at github.com/otodreas/BIOS15_Exam/

The script, data, and the report itself can be found in the GitHub repo at the following locations

```
root/
└── Report/
    ├── Scripts/
    │   └── Analysis.R
    ├── Data/
    │   └── penstemon_copy.txt
    └── Todreas_BIOS15_Exam.pdf
```

Below is a copy of the script for reference

```
# This script fits a GLM to the dataset provided. The goal of the analysis is
# to determine how total floral scent emission effects a plant's fitness. Users
# who do not clone the entire repo but instead just download the script will
# need to provide custom filepaths. The data file can be found at
# root/Report/Data/penstemon_copy.txt


# =====================
# CONFIGURE ENVIRONMENT
# =====================

# Clear variables
rm(list = ls())

# Load packages
library(here)
library(tidyverse)
library(glmmTMB)
library(MuMIn)

# Supress update warning from MuMIn because all the variables used in the
# original model call have the same values as when the model was fitted
options(MuMIn.noUpdateWarning = TRUE)
```

```r
# Set filepaths
data_path <- here("Report", "Data", "penstemon_copy.txt")
summary_path <- here("Report", "Output", "summary.txt")
params_path <- here("Report", "Output", "params.csv")


# =========
# LOAD DATA
# =========

# Load raw data
df <- as_tibble(
  read.table(data_path, header = TRUE)
) |>
  select(Pop, Block, tscent, fitness) |>  # Select relevant columns
  mutate(across(c(Pop, Block), as.factor)) |>  # Make grouped variables factors
  filter(tscent != 0 & fitness != 0) |>  # Drop values whose log is undefined
  mutate(log_tscent = log(tscent)) |>  # Create log_tscent column
  mutate(log_fitness = log(fitness))  # Create log_fitness column

# NOTE: one row is dropped for 0 values
# NOTE: Elasticity = for a 1% change in x (tscent), a x% change in y (fitness)


# ==================================
# FIT MODEL AND GENERATE PREDICTIONS
# ==================================

# Fit glmm on a log-log scale
m <- glmmTMB(log_fitness ~ log_tscent + Pop + (1|Block), data = df)

# Create data frames with sequences to generate new predictions on for each pop
new_data <- list()

for (i in seq_along(levels(df$Pop))) {
  new_data[[i]] <- tibble(
    log_tscent = seq(min(df$log_tscent), max(df$log_tscent), length.out = 15),
    Pop = factor(rep(levels(df$Pop)[i], 15))
  )
}

names(new_data) <- levels(df$Pop)

# Generate predictions and standard error estimates
preds <- list()

for (i in seq_along(new_data)) {
  preds[[i]] <- predict(
    m,
    newdata = new_data[[i]],
    type = "response",  # Calculate predictions on response scale
    se.fit = TRUE,
    re.form = NA  # Do not include random effects in predictions
  )
}

names(preds) <- levels(df$Pop)
```

```r
# =========
# DRAW PLOT
# =========

# Create plot on data scale
plot(
  df$tscent,
  df$fitness,
  xlab = "Total floral scent emission (ng/L/h)",
  ylab = "Fitness (mg)",
  col = adjustcolor(seq_along(levels(df$Pop)), alpha.f = 0.4),
  pch = 19,
)

# Draw legend
legend(
  "topleft",
  legend = levels(df$Pop),
  col = adjustcolor(seq_along(levels(df$Pop)), alpha.f = 0.4),
  pch = 19,
  bty = "n",
  horiz = TRUE,
  title = "Population",
  inset = 0.02
)

# Draw 95% CI ribbons on data scale
for (i in seq_along(preds)) {
  polygon(
    c(exp(new_data[[i]]$log_tscent), rev(exp(new_data[[i]]$log_tscent))),
    c(
      exp(preds[[i]]$fit + 1.96 * preds[[i]]$se.fit),
      rev(exp(preds[[i]]$fit - 1.96 * preds[[i]]$se.fit))
    ),
    col = adjustcolor(i, alpha.f = 0.15),
    border = FALSE
  )
}

# Draw regression lines on data scale
for (i in seq_along(preds)) {
  lines(exp(new_data[[i]]$log_tscent), exp(preds[[i]]$fit), col = i)
}

# NOTE: Plots were saved using RStudio/Positron interface


# ==================================
# BUILD SUMMARY STATISTICS TEXT FILES
# ==================================

# Assign variances to a vector
v_part <- c(attr(VarCorr(m)$cond$Block, "stddev")^2, attr(VarCorr(m)$cond, "sc")^2)

# Build summary file
cat(
```

```
  paste0(
    # R^2
    "R^2",
    "\nMarginal (represents variance explained only by the fixed effects): ",
    r.squaredGLMM(m)[1],
    "\nConditional (represents variance explained by the whole model): ",
    r.squaredGLMM(m)[2],

    # Variance partitioning
    "\n\nVariance partitioning",
    "\nVariance among blocks: ", v_part[1],
    "\nVariance within groups: ", v_part[2],
    "\n% variance explained by block: ", v_part[1] / sum(v_part) * 100
  ),
  file = summary_path
)

# Build parameters tibble
params <- as_tibble(summary(m)$coefficients$cond[, 1:2]) |>  # Get params & SE
  mutate(
    Parameter = c(
      "PopNR intercept (ln(mg))",
      "Slope (ln(mg))/ln(ng/L/h))",
      "PopTH intercept (ln(mg))",
      "PopWF intercept (ln(mg))"
    )
  ) |>  # Create parameter names that make sense and move the column left
  relocate(Parameter)

# Make every population's intercept absolute rather than relative
params[3, 2] <- params[1, 2] + params[3, 2]
params[4, 2] <- params[1, 2] + params[4, 2]

# Write parameters to file
write_csv(params, params_path)

# Define function to return preditions on data scale for a value of tscent
make_pred <- function(x_in) {
  # Ensure that each population is paired with the position of the parameter in
  # the params tibble
  pops <- list(1, 3, 4)
  names(pops) <- levels(df$Pop)
  estimates <- matrix(nrow = 3, ncol = 2)
  row <- 0
  for (i in levels(df$Pop)) {
    row <- row + 1
    estimate <- as.numeric(exp(log(x_in) * params[2, 2] + params[pops[[i]], 2]))
    estimates[row, ] <- c(x_in, estimate)
  }
  colnames(estimates) <- c("Input", "Prediction")
  rownames(estimates) <- levels(df$Pop)
  estimates
}
```