Fitness of *Penstemon digitalis*

BIOS15 Exam 2025-26

Oliver E. Todreas

2026-01-14

Introduction

Methods

Results

Conclusions

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

```r
# 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)
```

```r
# 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)

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


# =========
# 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: Elasticity = for a 1% change in x (tscent), a x% change in y (fitness)
# is expected ("Percent change in y per percent change in x")
# TODO: double check with course literature


# ==================================
# 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]],

    # TODO: check if theres a way to generate predictions on data scale
    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)


# ===================================
# 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(Fitness))",
      "Slope (ln(Fitness)/ln(Total floral scent emission (ng/L/h)))",
      "PopTH intercept (ln(Fitness))",
      "PopWF intercept (ln(Fitness))"
    )
  ) |>  # 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 given group
make_pred <- function(x_in, pop) {
  # 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)
  exp(log(x_in) * params[2, 2] + params[pops[pop][[1]], 2])
}

# Write example calculations into a file
cat("", file = example_path)  # Clear the file

# Loop through populations
for (i in levels(df$Pop)) {
  # Get smallest and largest observed values for tscent
  ranges <- df |>
    filter(Pop == i) |>
    pull(tscent) |>
    range()

  # Populate file with predictions on data scale
  cat(paste0(i, "\n"), file = example_path, append = TRUE)
  for (j in ranges) {
    cat(
      paste0("tscent: ", j, "\tprediction: ", make_pred(j, i), "\n"),
      file = example_path,
      append = TRUE
    )
  }
}


# =========
# DRAW PLOT
# =========

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

# Draw legend
legend(
  "topleft",
  legend = levels(df$Pop),
  col = seq_along(levels(df$Pop)),
  lty = 1,
  bty = "n",
  title = "Population"
)

# 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]]$fit) + 1.96 * exp(preds[[i]]$se.fit),
      rev(exp(preds[[i]]$fit) - 1.96 * exp(preds[[i]]$se.fit))
    ),
    col = adjustcolor(i, alpha.f = 0.25),
    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
```