```r
# No-choice predation test ----------------------------------------------------
# Packages --------------------------------------------------------------------
pkgs <- c(
  "tidyverse",
  "ARTool",
  "rstatix",
  "multcompView",
  "pheatmap"
)

to_install <- pkgs[!pkgs %in% installed.packages()[, "Package"]]
if (length(to_install) > 0) install.packages(to_install)
invisible(lapply(pkgs, library, character.only = TRUE))

# Read data -------------------------------------------------------------------
df <- read.csv("Book1.csv", check.names = FALSE)

# Data cleaning ---------------------------------------------------------------
df <- df %>%
  mutate(
    mite  = factor(mite),
    thrip = factor(thrip, levels = c("I", "II")),
    `No.` = as.integer(`No.`),
    add   = as.numeric(add),
    con   = as.numeric(con),
    prop  = con / add
  )

# Data check ------------------------------------------------------------------
print(dplyr::glimpse(df))
print(table(df$mite, df$thrip))

# Summary statistics ----------------------------------------------------------
sum_df <- df %>%
  group_by(mite, thrip) %>%
  summarise(
    n = n(),
    mean_con = mean(con, na.rm = TRUE),
    se_con   = sd(con, na.rm = TRUE) / sqrt(n),
    .groups = "drop"
  )

# Model testing ---------------------------------------------------------------
m_art <- art(con ~ mite * thrip, data = df)

cat("\n===== ART ANOVA (Type III-like tests via ARTool) =====\n")
print(anova(m_art))

# Pairwise comparisons --------------------------------------------------------
make_letters_within_thrip <- function(data, thrip_level, p_adjust = "holm") {
  d <- data %>% filter(thrip == thrip_level)

  if (n_distinct(d$mite) < 2) {
    return(tibble(mite = levels(d$mite), thrip = thrip_level, letter = "a"))
  }

  pw <- pairwise.wilcox.test(
    x = d$con,
    g = d$mite,
    p.adjust.method = p_adjust,
    exact = FALSE
  )
```

```
  pmat <- pw$p.value
  pairs <- as.data.frame(as.table(pmat)) %>%
    filter(!is.na(Freq)) %>%
    mutate(
      pair = paste(Var1, Var2, sep = "-"),
      p = Freq
    )

  p_named <- pairs$p
  names(p_named) <- pairs$pair

  letters <- multcompView::multcompLetters(p_named, threshold = 0.05)$Letters

  tibble(
    mite = names(letters),
    thrip = thrip_level,
    letter = unname(letters)
  )
}

letters_I  <- make_letters_within_thrip(df, "I",  p_adjust = "holm")
letters_II <- make_letters_within_thrip(df, "II", p_adjust = "holm")
letters_df <- bind_rows(letters_I, letters_II)

# Compact letter display ----------------------------------------------------
sum_df2 <- sum_df %>%
  left_join(letters_df, by = c("mite", "thrip")) %>%
  mutate(letter = ifelse(is.na(letter), "", letter))

cat("\n===== Mean ± SE + letters =====\n")
print(sum_df2)

# Stage comparison within each mite species ---------------------------------
within_mite_test <- df %>%
  group_by(mite) %>%
  wilcox_test(con ~ thrip, exact = FALSE) %>%
  adjust_pvalue(method = "holm") %>%
  add_significance("p.adj") %>%
  ungroup()

cat("\n===== Wilcoxon (within each mite: I vs II) =====\n")
print(within_mite_test)

# Spearman correlation ------------------------------------------------------
cor_list <- list()

for (tt in levels(df$thrip)) {
  dtt <- df %>%
    filter(thrip == tt) %>%
    select(`No.`, mite, con) %>%
    pivot_wider(names_from = mite, values_from = con)

  cm <- suppressWarnings(
    cor(dtt %>% select(-`No.`), method = "spearman", use = "pairwise.complete.obs")
  )
  cor_list[[tt]] <- cm

  cat("\n===== Spearman correlation among mites (Thrip", tt, ") =====\n")
  print(round(cm, 3))

  if (ncol(dtt) > 2) {
    pheatmap::pheatmap(
      cm,
      main = paste("Spearman correlation (Thrip", tt, ")"),
      border_color = NA
```

```
    )
  }
}

# Plot ------------------------------------------------------------------
thrip_cols <- c("I" = "#0072B2", "II" = "#E69F00")

# Remove all-zero combinations ------------------------------------------
zero_flag <- df %>%
  group_by(mite, thrip) %>%
  summarise(all_zero = all(con == 0), .groups = "drop")

plot_df <- sum_df2 %>%
  left_join(zero_flag, by = c("mite", "thrip")) %>%
  mutate(all_zero = ifelse(is.na(all_zero), FALSE, all_zero)) %>%
  filter(!all_zero) %>%  # remove all-zero bars
  mutate(
    mite  = factor(mite),
    thrip = factor(thrip, levels = c("I", "II"))
  ) %>%
  droplevels()

# Manual x positions ----------------------------------------------------
dodge_width <- 0.75
bar_width   <- 0.325   # half width
half_step   <- (dodge_width / 2) * 0.9

plot_df2 <- plot_df %>%
  group_by(mite) %>%
  mutate(
    n_bars = n(),                 # 1 or 2 existing thrip levels for this mite
    x_base = as.numeric(mite),
    x_bar = case_when(
      n_bars == 1 ~ x_base,
      n_bars == 2 & thrip == "I"  ~ x_base - half_step,
      n_bars == 2 & thrip == "II" ~ x_base + half_step,
      TRUE ~ x_base
    )
  ) %>%
  ungroup()

# Remove selected compact letter ----------------------------------------
plot_df2 <- plot_df2 %>%
  mutate(
    letter = ifelse(mite == "A. limonicus" & thrip == "II", "", letter)
  )

# Italic x-axis labels --------------------------------------------------
mite_levels <- levels(plot_df2$mite)
mite_labels_italic <- lapply(mite_levels, function(x) bquote(italic(.(x))))

# Figure 1 --------------------------------------------------------------
p <- ggplot(plot_df2, aes(x = x_bar, y = mean_con, fill = thrip)) +
  geom_col(width = bar_width) +
  geom_errorbar(
    aes(ymin = mean_con - se_con, ymax = mean_con + se_con),
    width = 0.18
  ) +
  geom_text(
    aes(label = letter, y = mean_con + se_con + 0.15),
    size = 7
  ) +
  scale_fill_manual(values = thrip_cols, drop = TRUE) +
  scale_x_continuous(
    breaks = sort(unique(plot_df2$x_base)),
```

```
    labels = mite_labels_italic,
    expand = expansion(mult = c(0.02, 0.02))
  ) +
  scale_y_continuous(
    limits = c(0, NA),
    expand = expansion(mult = c(0, 0.06))
  ) +
  labs(
    x = "Predatory mite species",
    y = "Thrips consumed",
    fill = "Thrips instar"
  ) +
  theme_classic(base_size = 28) +
  theme(
    axis.text.x = element_text(angle = 25, hjust = 1),
    legend.position = "right"
  )

print(p)




# Choice test ------------------------------------------------------------------
# Packages ---------------------------------------------------------------------
pkgs <- c("tidyverse")
to_install <- pkgs[!pkgs %in% installed.packages()[, "Package"]]
if (length(to_install) > 0) install.packages(to_install, dependencies = TRUE)
library(tidyverse)

# Read data --------------------------------------------------------------------
dat_raw <- read.csv("book2.1.csv", stringsAsFactors = FALSE)

# Data cleaning ----------------------------------------------------------------
dat <- dat_raw %>% rename_with(tolower)

# Column check -----------------------------------------------------------------
need_cols <- c("no", "stage", "add", "con")
miss_cols <- setdiff(need_cols, colnames(dat))
if (length(miss_cols) > 0) {
  stop("缺少必要列：", paste(miss_cols, collapse = ", "))
}

# Data formatting --------------------------------------------------------------
dat <- dat %>%
  mutate(
    no    = as.factor(no),
    add   = as.numeric(add),
    con   = as.numeric(con),
    stage = as.character(stage)
  ) %>%
  mutate(
    stage = case_when(
      stage %in% c("I","i","1")  ~ "I",
      stage %in% c("II","ii","2") ~ "II",
      TRUE ~ stage
    ),
    stage = factor(stage, levels = c("I","II"))
  )

# Summary statistics -----------------------------------------------------------
sum_tab <- dat %>%
  group_by(stage) %>%
  summarise(
```

```
    mean = mean(con, na.rm = TRUE),
    se   = sd(con, na.rm = TRUE) / sqrt(n()),
    .groups = "drop"
  )

print(sum_tab)

# Figure 2 ----------------------------------------------------------------
p_bar <- ggplot(sum_tab, aes(x = stage, y = mean, fill = stage)) +
  geom_col(width = 0.6, color = "black") +
  geom_errorbar(
    aes(ymin = mean - se, ymax = mean + se),
    width = 0.15,
    linewidth = 1
  ) +
  scale_fill_manual(values = c("I" = "#0072B2", "II" = "#E69F00")) +
  scale_y_continuous(
    limits = c(0, 4),
    breaks = seq(0, 4, by = 1),
    expand = c(0, 0)
  ) +
  scale_x_discrete(expand = expansion(mult = c(0.25, 0.25))) +
  labs(
    x = "Thrips instar",
    y = "Number consumed (mean ± SE)",
    fill = "Thrips instar",

  ) +
  theme_classic(base_family = "Times New Roman") +
  theme(
    text = element_text(size = 38),
    plot.title = element_text(size = 38, face = "bold", hjust = 0.5),
    axis.title = element_text(size = 38, face = "bold"),
    axis.text  = element_text(size = 38),
    legend.position = "right",
    legend.title = element_text(size = 38, face = "bold"),
    legend.text  = element_text(size = 38),
    legend.key.size = unit(1.2, "lines"),
    axis.ticks.length = unit(0.3, "cm")
  )

print(p_bar)



# Functional response analysis --------------------------------------------
rm(list = ls())

# Packages ----------------------------------------------------------------
req_pkgs <- c("dplyr", "ggplot2", "frair", "bbmle", "tibble")
to_install <- req_pkgs[!sapply(req_pkgs, requireNamespace, quietly = TRUE)]
if (length(to_install) > 0) install.packages(to_install)
invisible(lapply(req_pkgs, library, character.only = TRUE))

# Read data ---------------------------------------------------------------
dat <- read.csv("book3.1.csv", stringsAsFactors = FALSE)

# Data check --------------------------------------------------------------
stopifnot(all(c("rep", "add", "eat") %in% names(dat)))

dat <- dat %>%
  mutate(
    rep = as.factor(rep),
    add = as.numeric(add),
    eat = as.numeric(eat)
```

```
  ) %>%
  filter(!is.na(add), !is.na(eat), add > 0) %>%
  mutate(
    eat = pmax(0, pmin(eat, add))
  )

# Juliano logistic test ------------------------------------------------------
fr_test <- frair_test(eat ~ add, data = dat)

cat("\n========== frair_test (Juliano logistic test) ==========\n")
print(fr_test)

# Binomial logistic regression -----------------------------------------------
glm_juliano <- glm(
  cbind(eat, add - eat) ~ add + I(add^2),
  family = binomial,
  data = dat
)

cat("\n========== Explicit binomial logistic regression ==========\n")
print(summary(glm_juliano))

coef_tab <- summary(glm_juliano)$coefficients
write.csv(as.data.frame(coef_tab), "Juliano_logistic_coefficients.csv")

# Rogers Type II functional response -----------------------------------------
rog_fit <- frair_fit(
  eat ~ add,
  data = dat,
  response = "rogersII",
  start = list(a = 1.2, h = 0.15),   # adjust if needed
  fixed = list(T = 1)
)

cat("\n========== Rogers Type II fit ==========\n")
print(summary(rog_fit$fit))
cat("\nCoefficients:\n")
print(coef(rog_fit))

# Generalised functional response model --------------------------------------
flex_fit <- frair_fit(
  eat ~ add,
  data = dat,
  response = "flexpnr",
  start = list(b = 1.2, q = 0.1, h = 0.15),  # q free
  fixed = list(T = 1)
)

cat("\n========== Generalized flexpnr fit (q free) ==========\n")
print(summary(flex_fit$fit))
cat("\nCoefficients:\n")
print(coef(flex_fit))

# Nested model with q fixed at zero ------------------------------------------
flex_q0_fit <- frair_fit(
  eat ~ add,
  data = dat,
  response = "flexpnr",
  start = list(b = 1.2, h = 0.15),
  fixed = list(T = 1, q = 0)
)

cat("\n========== Nested flexpnr fit (q = 0) ==========\n")
print(summary(flex_q0_fit$fit))
cat("\nCoefficients:\n")
```

```
print(coef(flex_q0_fit))

cat("\n========== AIC comparison ==========\n")
print(AIC(rog_fit$fit, flex_fit$fit, flex_q0_fit$fit))

aic_vals <- AIC(rog_fit$fit, flex_fit$fit, flex_q0_fit$fit)
aic_vals$deltaAIC_vs_q0 <- aic_vals$AIC - aic_vals$AIC[aic_vals$df == aic_vals$df[aic_vals$df ==
min(aic_vals$df)]][1]
print(aic_vals)

# Delta AIC ------------------------------------------------------------------
delta_q <- AIC(flex_fit$fit)[1, "AIC"] - AIC(flex_q0_fit$fit)[1, "AIC"]
cat("\nDelta AIC [q free - q = 0] =", delta_q, "\n")

cat("\n========== Rogers Type II confidence intervals ==========\n")
print(confint(rog_fit$fit))

cat("\n========== flexpnr confidence intervals ==========\n")
print(confint(flex_fit$fit))

sum_df <- dat %>%
  group_by(add) %>%
  summarise(
    n = n(),
    mean_eat = mean(eat, na.rm = TRUE),
    se_eat = sd(eat, na.rm = TRUE) / sqrt(n),
    .groups = "drop"
  )

# Prediction curve -----------------------------------------------------------
xseq <- seq(0, max(dat$add), length.out = 300)

pred_rog <- data.frame(
  add = xseq,
  pred = frair::rogersII(xseq,
                         a = coef(rog_fit)["a"],
                         h = coef(rog_fit)["h"],
                         T = 1)
)

pred_flex <- data.frame(
  add = xseq,
  pred = frair::flexpnr(xseq,
                        b = coef(flex_fit)["b"],
                        q = coef(flex_fit)["q"],
                        h = coef(flex_fit)["h"],
                        T = 1)
)

# Figure 3 -------------------------------------------------------------------
p <- ggplot() +
  geom_point(
    data = sum_df,
    aes(x = add, y = mean_eat),
    size = 5
  ) +
  geom_errorbar(
    data = sum_df,
    aes(x = add,
        ymin = pmax(0, mean_eat - se_eat),
        ymax = mean_eat + se_eat),
    width = 0.25,
    linewidth = 1.2
  ) +
  geom_line(
```

```
    data = pred_rog,
    aes(x = add, y = pred),
    linewidth = 2
  ) +
  scale_x_continuous(
    limits = c(0, max(dat$add)),
    breaks = sort(unique(dat$add))
  ) +
  scale_y_continuous(
    limits = c(0, NA),
    expand = expansion(mult = c(0, 0.08))
  ) +
  labs(
    x = "Prey density",
    y = "Number consumed"
  ) +
  theme_classic(base_size = 38, base_family = "Times New Roman") +
  theme(
    axis.title.x = element_text(size = 38),
    axis.title.y = element_text(size = 38),
    axis.text.x  = element_text(size = 38),
    axis.text.y  = element_text(size = 38)
  )

print(p)

# Save figure ----------------------------------------------------------------
ggsave(
  "Functional_response_frair.png",
  p,
  width = 12,
  height = 8,
  dpi = 600
)
```