BumblebeeRobbing

Daniel

2025-06-09

Behavioral assays done on bumblebees at Rocky Mountains Biology Laboratory (RMBL) field season 2023. Observations were carried out by Nick Dabagia and Daniel Souto for Corydalis and Mertensia species. Y-Tube assays carried out by Oriana Gutierrez in 2024.

Methods:

Refer to Souto-Vilarós et al. 'Yeast volatiles promote larceny in bumble bee behavior.'

Data analysis:

The dataset is arranged as date, plant species, bee individual, bee species, whether or not a choice was made, stalk number, treatment, whether or not the bee tried to legitimately visit the flower, whether it robbed the flower, time to rob, time feeding, flower number (total for that bout) and additional notes.

Note that not all packages are used in the current analysis

We need to wrangle the data a bit. Namely: Filter the data to include only cases where a flower was robbed (column CHOICE, "Yes"), Remove an outlier which was inactive on a flwoer over 3 minutes. Make sure all data is numeric and there are no missing data.

This final data set includes: 57 total successful robber trials (out of 116 bee trials transcribed) 42 Corydalis 2° robbers (22 bifarius, 16 flavifrons, 4 mixtus) and 15 Mertensia 2° robbers (all flavifrons).

Bombus mixtus removed from final analysis due to low sample size.

```
robbed_flowers <- visitation %>%
  filter(robbed == "yes") %>% #note that this will reduce total number of bees which made a choice sinc
  filter(species != "mixtus")%>%
  filter(timetorob !=154.7)

robbed_flowers <- robbed_flowers %>%
  mutate(
    timetorob = as.numeric(timetorob),
    timefeeding = as.numeric(timefeeding)
  ) %>%
  filter(!is.na(timetorob) & !is.na(timefeeding))

robbed_flowers <- robbed_flowers %>%
  mutate(trytolegit_binary = ifelse(trytolegit == "yes", 1, 0))

#How many bees per bee species per flower species in the original and the filtered datasets?

visitation %>% group_by(sample, plantsp, species) %>% count()
```

```
## # A tibble: 120 x 4
```

```
## # Groups:   sample, plantsp, species [120]
##    sample     plantsp   species      n
##    <chr>      <chr>     <chr>    <int>
##  1 CORYBIF001 corydalis bifarius     5
##  2 CORYBIF002 corydalis bifarius     1
##  3 CORYBIF003 corydalis bifarius     2
##  4 CORYBIF004 corydalis bifarius    16
##  5 CORYBIF005 corydalis bifarius     1
##  6 CORYBIF006 corydalis bifarius    11
##  7 CORYBIF007 corydalis bifarius    10
##  8 CORYBIF008 corydalis bifarius    10
##  9 CORYBIF009 corydalis bifarius     1
## 10 CORYBIF010 corydalis bifarius     1
## # i 110 more rows
```

```
robbed_flowers %>% group_by(sample, plantsp, species, bee) %>% count()
```

```
## # A tibble: 49 x 5
## # Groups:   sample, plantsp, species, bee [49]
##    sample     plantsp   species    bee     n
##    <chr>      <chr>     <chr>    <int> <int>
##  1 CORYBIF001 corydalis bifarius     3     3
##  2 CORYBIF004 corydalis bifarius     3    15
##  3 CORYBIF006 corydalis bifarius     7    11
##  4 CORYBIF007 corydalis bifarius     8    10
##  5 CORYBIF008 corydalis bifarius     2     9
##  6 CORYBIF011 corydalis bifarius     7    12
##  7 CORYBIF012 corydalis bifarius     8    19
##  8 CORYBIF014 corydalis bifarius     5    20
##  9 CORYBIF017 corydalis bifarius    11     9
## 10 CORYBIF019 corydalis bifarius     2     6
## # i 39 more rows
```

Simple summary statistics for time to rob (mean, SE) broken down by bumblebee species and plant sp. Note that time to rob is longer for control flowers, but also SE higher for controls. Consistent response for Mreu plants. Tactic switch, I just converted into a binary column yes = 1, no = 0. Feeding time is higher in both treatments, with high SE but consistent throughout.

```
summary_stats <- robbed_flowers %>%
  group_by(species, treatment, plantsp) %>%
  summarize(
    Mean_TimeToRob = mean(timetorob, na.rm = TRUE),
    SE_TimeToRob = sd(timetorob, na.rm = TRUE) / sqrt(n()),
    Mean_TimeFeeding = mean(timefeeding, na.rm = TRUE),
    SE_TimeFeeding = sd(timetorob, na.rm = TRUE) / sqrt(n()),
    Mean_tacticswitch = mean(trytolegit_binary, na.rm = TRUE),
    SE_tacticswitch = sd(trytolegit_binary, na.rm = TRUE) / sqrt(n())
  ) %>%
  arrange(plantsp, treatment)
```

```
## `summarise()` has grouped output by 'species', 'treatment'. You can override
## using the `.groups` argument.
```

```r
print(summary_stats)
```

```
## # A tibble: 6 x 9
## # Groups:   species, treatment [4]
##   species    treatment plantsp   Mean_TimeToRob SE_TimeToRob Mean_TimeFeeding
##   <chr>      <chr>     <chr>              <dbl>        <dbl>            <dbl>
## 1 bifarius   control   corydalis           5.42        0.637             7.76
## 2 flavifrons control   corydalis           5.24        0.672             8.04
## 3 bifarius   mreu      corydalis           4.16        0.343             7.09
## 4 flavifrons mreu      corydalis           3.20        0.272             8.04
## 5 flavifrons control   mertensia           7.12        1.08              9.25
## 6 flavifrons mreu      mertensia           3.97        0.426             8.06
## # i 3 more variables: SE_TimeFeeding <dbl>, Mean_tacticswitch <dbl>,
## #   SE_tacticswitch <dbl>
```

```r
write.table(summary_stats, "results/summary_feeding_stats.csv", row.names = FALSE)
```

```r
# Summarize the data to calculate the number of visits
summary_data <- robbed_flowers %>%
  group_by(treatment, species, plantsp) %>%
  summarize(
    number_of_visits = n(),  # Count the number of visits
    .groups = "drop"
  )

# Calculate the mean, stdev, and se for each treatment group (control vs mreu)
visit_summary <- summary_data %>%
  group_by(species, plantsp) %>%
  summarize(
    mean_visits_control = mean(number_of_visits[treatment == "control"], na.rm = TRUE),
    mean_visits_treated = mean(number_of_visits[treatment == "mreu"], na.rm = TRUE),
    stdev_diff = sd(number_of_visits[treatment == "mreu"] - number_of_visits[treatment == "control"], na
    se_diff = stdev_diff / sqrt(n()),
    .groups = "drop"
  )

# View the resulting summary table
print(visit_summary)
```

```
## # A tibble: 3 x 6
##   species    plantsp  mean_visits_control mean_visits_treated stdev_diff se_diff
##   <chr>      <chr>                  <dbl>               <dbl>      <dbl>   <dbl>
## 1 bifarius   corydal~                 114                 141         NA      NA
## 2 flavifrons corydal~                  47                 128         NA      NA
## 3 flavifrons mertens~                  39                  78         NA      NA
```

```r
# Perform t-tests for each species and plantsp combination
t_test_results <- robbed_flowers %>%
  group_by(species, plantsp) %>%
  summarize(
    t_test = list(
      t.test(
```

```
        x = treatment == "mreu",     # Logical vector for mreu
        y = treatment == "control" # Logical vector for control
      )
    ),
    .groups = "drop"
  )

# Extract p-values and t-statistics from the t-test results
t_test_summary <- t_test_results %>%
  mutate(
    p_value = sapply(t_test, function(x) x$p.value),        # Extract p-value
    statistic = sapply(t_test, function(x) x$statistic)    # Extract t-statistic
  )

# Select only necessary columns
t_test_summary <- t_test_summary %>%
  dplyr::select(species, plantsp, p_value, statistic)

# View the summarized results
print(t_test_summary)
```

```
## # A tibble: 3 x 4
##   species    plantsp    p_value statistic
##   <chr>      <chr>        <dbl>     <dbl>
## 1 bifarius   corydalis 1.68e- 2      2.40
## 2 flavifrons corydalis 5.52e-20      9.74
## 3 flavifrons mertensia 1.77e- 7      5.39
```

```
# Extract p-values, t-statistics, and degrees of freedom from the t-test results
t_test_summary <- t_test_results %>%
  mutate(
    p_value = sapply(t_test, function(x) x$p.value),        # Extract p-value
    statistic = sapply(t_test, function(x) x$statistic),    # Extract t-statistic
    df = sapply(t_test, function(x) x$parameter)            # Extract degrees of freedom
  )

# Explicitly use dplyr::select()
t_test_summary <- t_test_summary %>%
  dplyr::select(species, plantsp, p_value, statistic, df)

# View the summarized results
print(t_test_summary)
```

```
## # A tibble: 3 x 5
##   species    plantsp    p_value statistic    df
##   <chr>      <chr>        <dbl>     <dbl> <dbl>
## 1 bifarius   corydalis 1.68e- 2      2.40   508
## 2 flavifrons corydalis 5.52e-20      9.74   348
## 3 flavifrons mertensia 1.77e- 7      5.39   232
```

```
visit_choices <- robbed_flowers %>%
  group_by(sample, species, plantsp) %>%
```

```
  summarise(
    mreu_visits = sum(treatment == "mreu"),
    control_visits = sum(treatment == "control"),
    .groups = "drop"
  )

m_choice <- glmer(
  cbind(mreu_visits, control_visits) ~ plantsp * species + (1 | sample),
  data = visit_choices,
  family = binomial
)
```

```
## fixed-effect model matrix is rank deficient so dropping 1 column / coefficient
```

```
summary(m_choice)
```

```
## Generalized linear mixed model fit by maximum likelihood (Laplace
##   Approximation) [glmerMod]
##  Family: binomial  ( logit )
## Formula: cbind(mreu_visits, control_visits) ~ plantsp * species + (1 |
##     sample)
##    Data: visit_choices
##
##       AIC       BIC    logLik -2*log(L)  df.resid
##     195.8     203.4     -93.9     187.8        45
##
## Scaled residuals:
##      Min       1Q   Median       3Q      Max
## -0.65593 -0.09027 -0.01831  0.30905  0.40899
##
## Random effects:
##  Groups Name        Variance Std.Dev.
##  sample (Intercept) 17.32    4.161
## Number of obs: 49, groups:  sample, 49
##
## Fixed effects:
##                   Estimate Std. Error z value Pr(>|z|)
## (Intercept)          1.861      1.247   1.492    0.136
## plantspmertensia    -1.752      1.914  -0.915    0.360
## speciesflavifrons    1.552      1.784   0.870    0.385
##
## Correlation of Fixed Effects:
##             (Intr) plntsp
## plntspmrtns -0.107
## spcsflvfrns -0.514 -0.572
## fit warnings:
## fixed-effect model matrix is rank deficient so dropping 1 column / coefficient
```

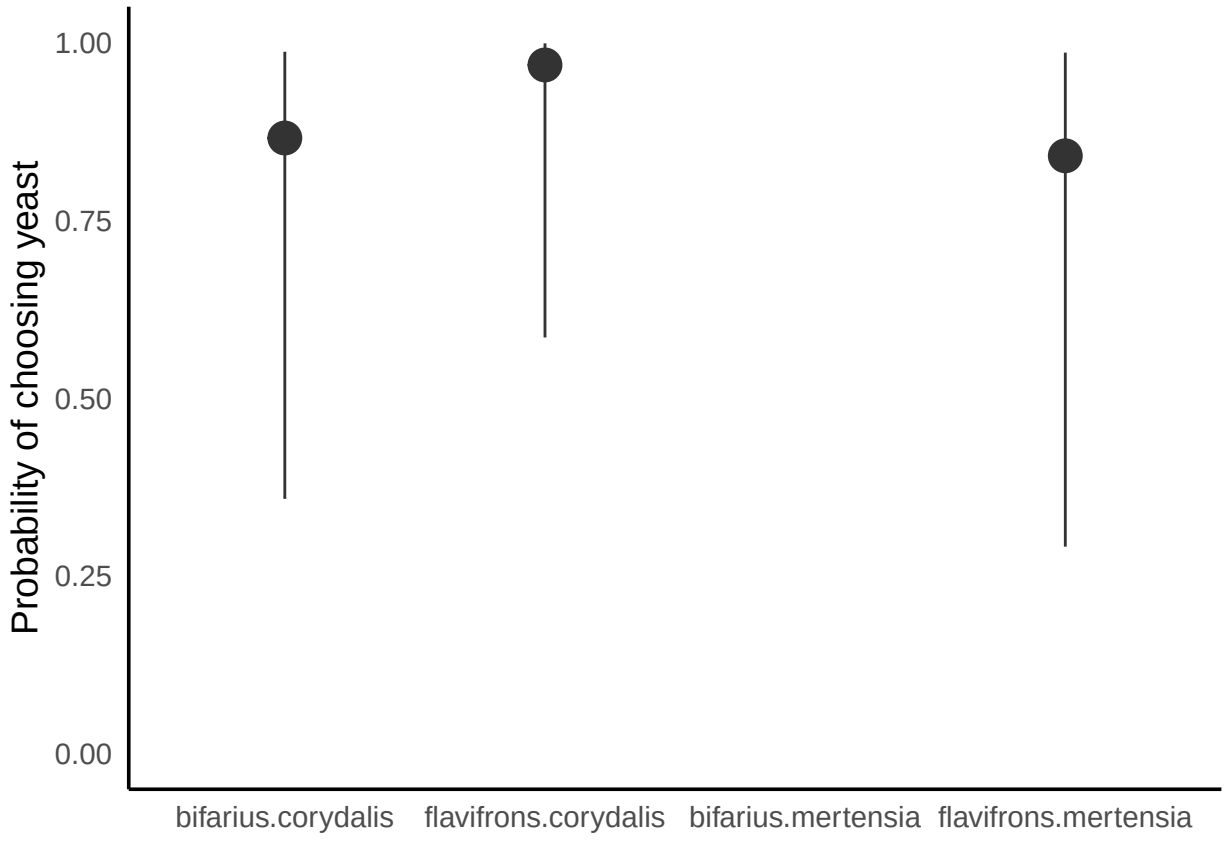
```
emm_choice <- emmeans(m_choice, ~ plantsp * species, type = "response")
emm_choice
```

```
##  plantsp   species      prob     SE  df asymp.LCL asymp.UCL
```

```
##  corydalis bifarius    0.865 0.1450 Inf     0.358     0.987
##  mertensia bifarius   nonEst     NA  NA        NA        NA
##  corydalis flavifrons  0.968 0.0484 Inf     0.585     0.998
##  mertensia flavifrons  0.840 0.1750 Inf     0.291     0.985
##
## Confidence level used: 0.95
## Intervals are back-transformed from the logit scale
```

```r
emm_choice_df <- as.data.frame(emm_choice)

ggplot(emm_choice_df, aes(x = interaction(species, plantsp),
                          y = prob, ymin = asymp.LCL, ymax = asymp.UCL)) +
  geom_pointrange(color = "gray20", size = 1.2) +
  coord_cartesian(ylim = c(0,1)) +
  labs(x = NULL, y = "Probability of choosing yeast") +
  theme_minimal(base_size = 14) +
  theme(panel.grid.major = element_blank(),
        panel.grid.minor = element_blank(),
        axis.line = element_line(color = "black"))
```

```
## Warning: Removed 1 row containing missing values or values outside the scale range
## (`geom_pointrange()`).
```



```r
test(emm_choice, null = 0.5)
```

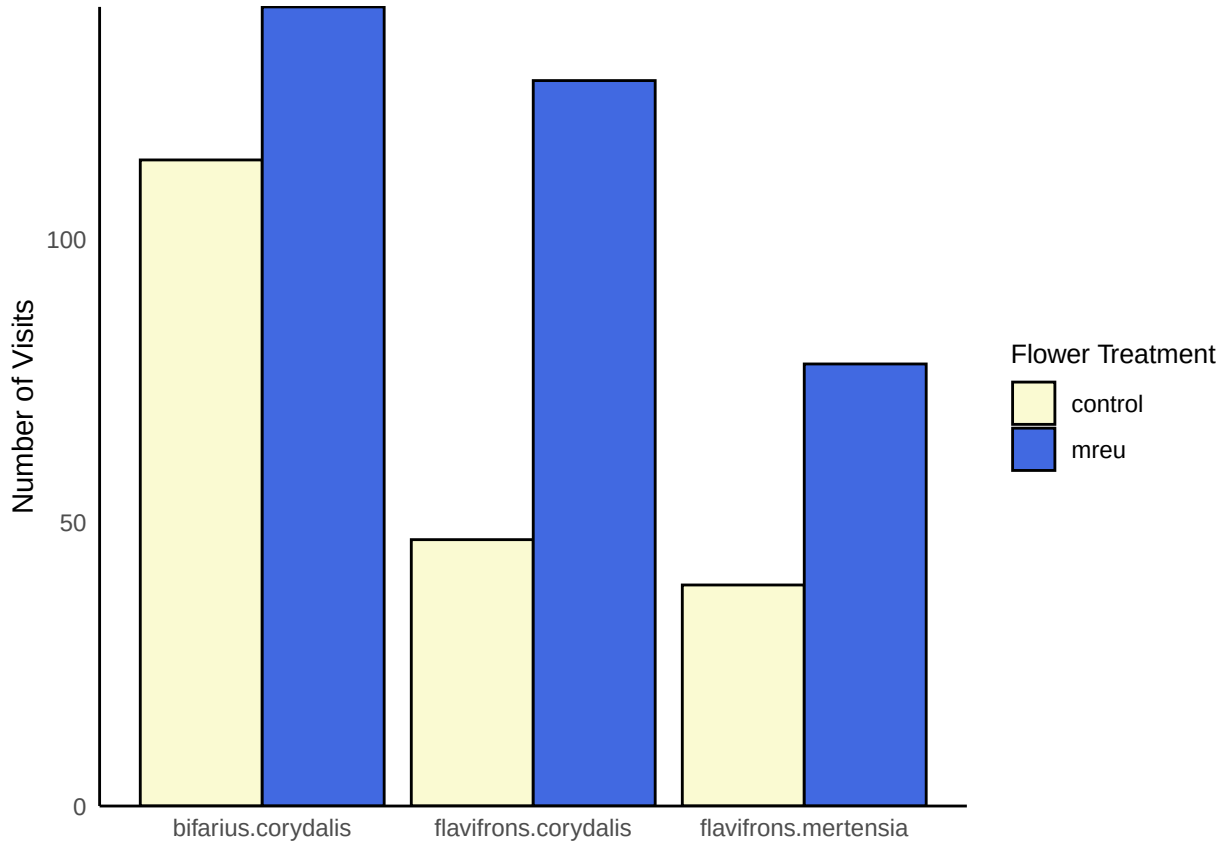
```
##  plantsp   species      prob     SE  df  null z.ratio p.value
##  corydalis bifarius    0.865 0.1450 Inf 0.622   1.091  0.2751
##  mertensia bifarius   nonEst     NA  NA 0.622      NA      NA
##  corydalis flavifrons  0.968 0.0484 Inf 0.622   1.860  0.0629
##  mertensia flavifrons  0.840 0.1750 Inf 0.622   0.892  0.3724
##
## Tests are performed on the logit scale
```

```r
# visits per bout
visits_per_bout <- robbed_flowers %>%
  group_by(sample) %>%
  summarise(total_visits = n(), .groups = "drop")

# mean ± SE across all bouts
mean_visits <- mean(visits_per_bout$total_visits)
se_visits <- sd(visits_per_bout$total_visits) / sqrt(nrow(visits_per_bout))

# visits per bout
visits_per_bout <- robbed_flowers %>%
  group_by(sample) %>%
  summarise(total_visits = n(), .groups = "drop")

# mean ± SE across all bouts
mean_visits <- mean(visits_per_bout$total_visits)
se_visits <- sd(visits_per_bout$total_visits) / sqrt(nrow(visits_per_bout))

cat("Mean number of flower visits per bout =",
    round(mean_visits, 2), "±", round(se_visits, 2), "SE\n")
```

```
## Mean number of flower visits per bout = 11.16 ± 1.03 SE
```

```r
# Create a histogram
ggplot(robbed_flowers, aes(x = interaction(species, plantsp), fill = treatment)) +
  geom_bar(position = "dodge", color = "black") +
  labs(x = NULL,
    y = "Number of Visits",
    fill = "Flower Treatment"
  ) +
  scale_fill_manual(
    values = c("mreu" = "#4169E1",
               "control" = "#FAFAD2")
  )+
  scale_y_continuous(expand = c(0, 0)) +  # Ensures bars rest on the x-axis line
  theme_minimal() +
  theme(
    panel.grid.major = element_blank(),
    panel.grid.minor = element_blank(),
    axis.line = element_line(color = "black"),
    axis.line.x = element_line(),
    axis.line.y = element_line(),
```

```
    legend.title = element_text(size = 10),
    legend.text = element_text(size = 9)
  ) -> plot

plot
```



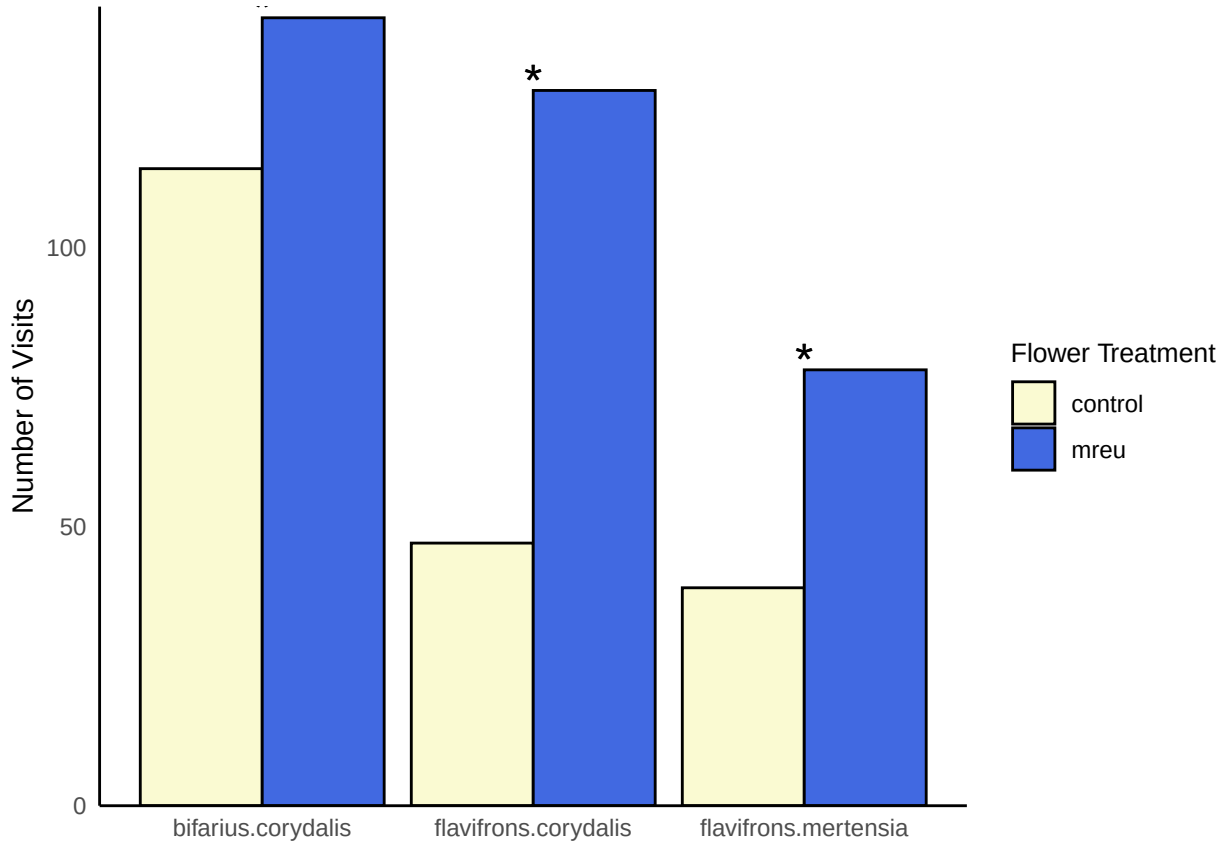

```
ggsave("results/bee_visits_histogram.pdf", plot = plot, width = 8, height = 6)
```

```
summary_df <- robbed_flowers %>%
  group_by(group = interaction(species, plantsp), treatment) %>%
  summarise(count = n(), .groups = "drop") %>%
  group_by(group) %>%
  summarise(
    max_y = max(count),
    x_pos = as.numeric(factor(group)),  # for proper placement on x axis
    .groups = "drop"
  ) %>%
  mutate(label = "*")
```

```
## Warning: Returning more (or less) than 1 row per `summarise()` group was deprecated in
## dplyr 1.1.0.
## i Please use `reframe()` instead.
## i When switching from `summarise()` to `reframe()`, remember that `reframe()`
##   always returns an ungrouped data frame and adjust accordingly.
```

```
## Call ‘lifecycle::last_lifecycle_warnings()‘ to see where this warning was
## generated.
```

```
ggplot(robbed_flowers, aes(x = interaction(species, plantsp), fill = treatment)) +
  geom_bar(position = position_dodge(width = 0.9), color = "black") +
  geom_text(
    data = summary_df,
    aes(x = group, y = max_y + 2, label = label),
    inherit.aes = FALSE,
    size = 6
  ) +
  labs(x = NULL,
       y = "Number of Visits",
       fill = "Flower Treatment") +
  scale_fill_manual(
    values = c("mreu" = "#4169E1", "control" = "#FAFAD2")
  ) +
  scale_y_continuous(expand = c(0, 0)) +  # Ensures bars rest on the x-axis line
  theme_minimal() +
  theme(
    panel.grid.major = element_blank(),
    panel.grid.minor = element_blank(),
    axis.line = element_line(color = "black"),
    axis.line.x = element_line(),
    axis.line.y = element_line(),
    legend.title = element_text(size = 10),
    legend.text = element_text(size = 9)
  ) -> plot2

plot2
```




```
ggsave("results/bee_visits_histogram.pdf", plot = plot2, width = 8, height = 6)
```

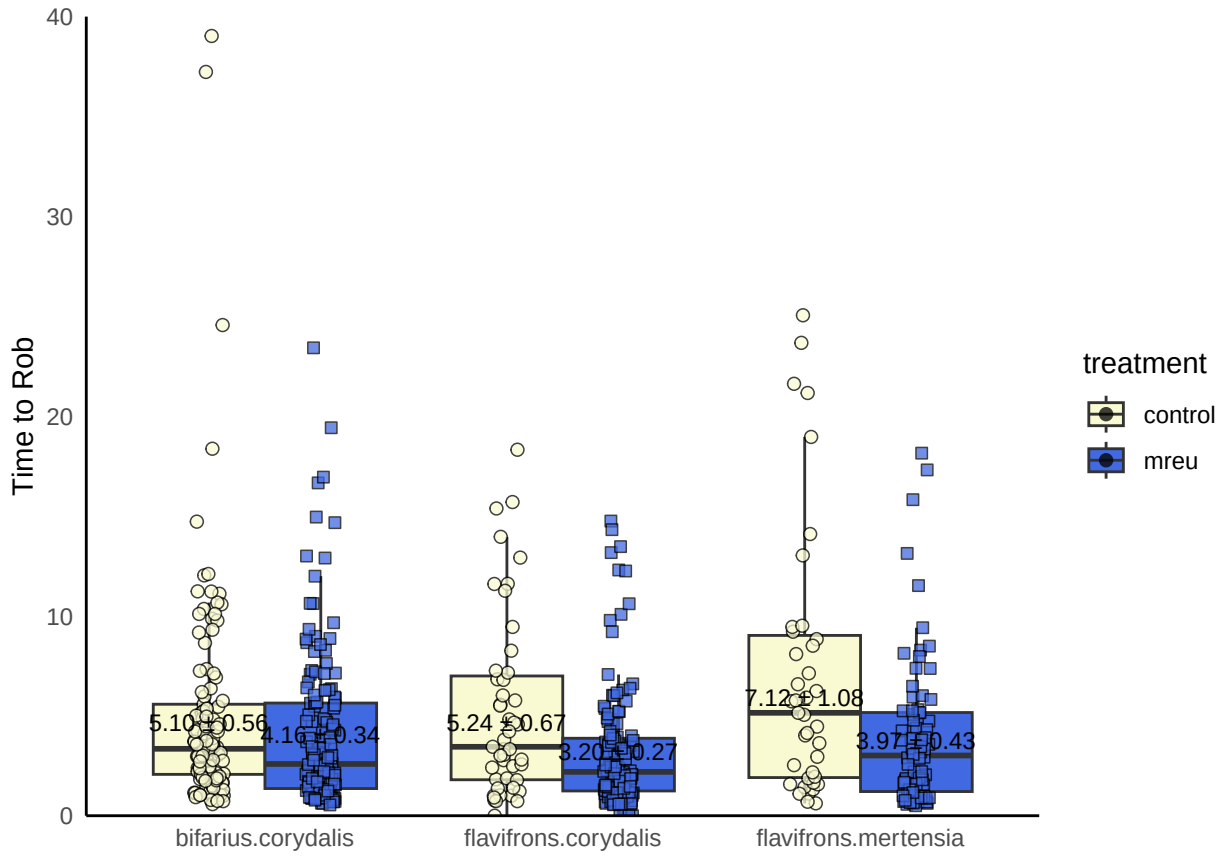
```
mean_se <- function(x) {
  m <- mean(x, na.rm = TRUE)
  se <- sd(x, na.rm = TRUE) / sqrt(length(na.omit(x)))
  return(paste0(round(m, 2), " ± ", round(se, 2)))
}

timetorob_box_species <- ggplot(
  robbed_flowers,
  aes(x = interaction(species, plantsp), y = timetorob, fill = treatment)
) +
  geom_boxplot(outlier.shape = NA, position = position_dodge(width = 0.75)) +
  geom_point(
    position = position_jitterdodge(jitter.width = 0.1, dodge.width = 0.75),
    aes(shape = treatment, fill = treatment),   # use fill, not color
    size = 2,
    stroke = 0.4,       # thin outline
    color = "black",    # outline color
    alpha = 0.75
  ) +
  stat_summary(
    fun.data = function(x) {
      m <- mean(x)
      se <- sd(x) / sqrt(length(x))
```

```
      data.frame(y = m + se * -1.5, label = sprintf("%.2f ± %.2f", m, se))
    },
    geom = "text",
    position = position_dodge(width = 0.75),
    size = 3,
    vjust = 0,
    color = "black"
  ) +
  scale_fill_manual(
    values = c("mreu" = "#4169E1",
               "control" = "#FAFAD2")
  ) +
  labs(x = NULL, y = "Time to Rob") +
  scale_shape_manual(name = "Treatment", values = c(21, 22), guide = "none") +
  scale_y_continuous(expand = c(0, 0), limits = c(0, 40)) +
  theme_minimal() +
  theme(
    panel.grid.major = element_blank(),
    panel.grid.minor = element_blank(),
    axis.line = element_line(color = "black")
  )

timetorob_box_species
```

```
## Warning: Removed 1 row containing non-finite outside the scale range
## (`stat_boxplot()`).
```

```
## Warning: Removed 1 row containing non-finite outside the scale range
## (`stat_summary()`).
```

```
## Warning: Removed 1 row containing missing values or values outside the scale range
## (`geom_point()`).
```



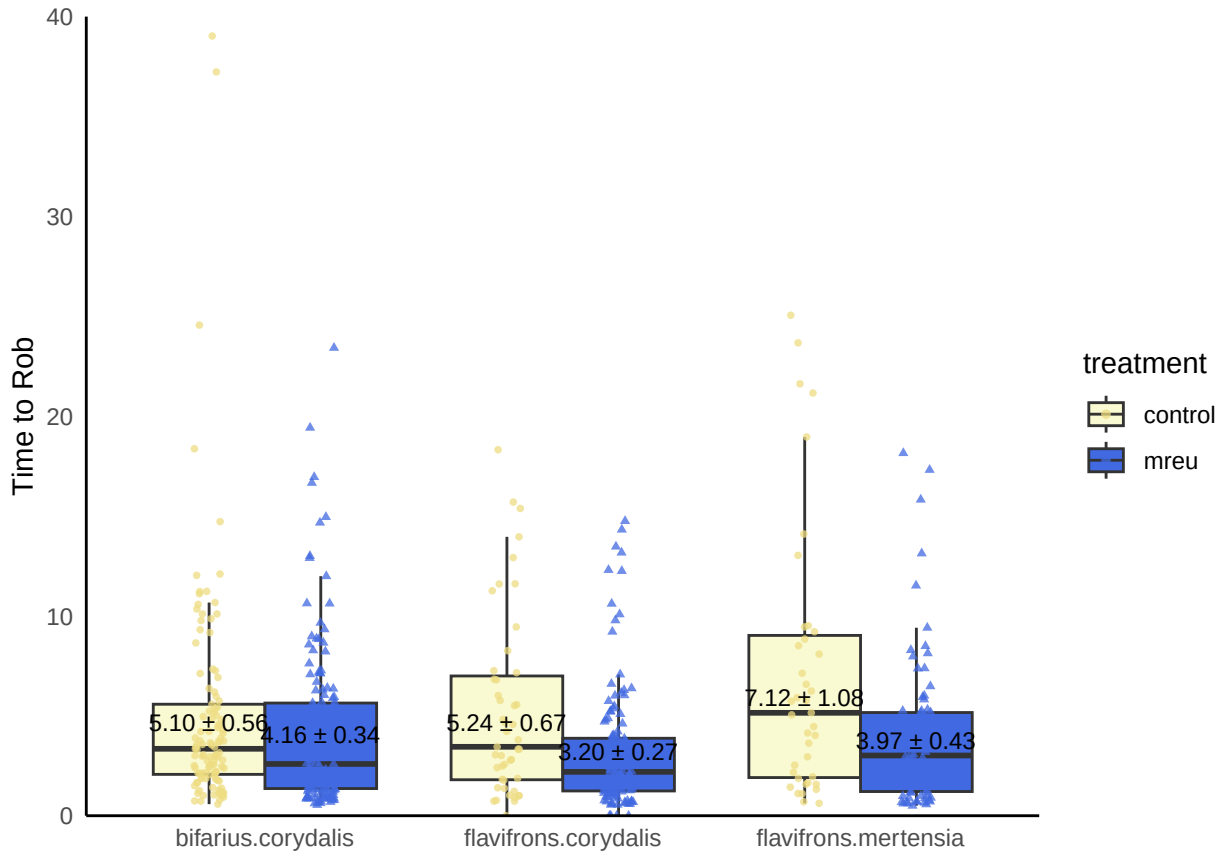

```
timetorob_box_species <- ggplot(robbed_flowers, aes(x = interaction(species, plantsp), y = timetorob, f
  geom_boxplot(outlier.shape = NA, position = position_dodge(width = 0.75)) +
  geom_point(position = position_jitterdodge(jitter.width = 0.1, dodge.width = 0.75),
             aes(shape = treatment, color = treatment),
             size = 1, alpha = 0.75) +
  stat_summary(
    fun.data = function(x) {
      m <- mean(x)
      se <- sd(x) / sqrt(length(x))
      data.frame(y = m + se * -1.5, label = sprintf("%.2f ± %.2f", m, se))
    },
    geom = "text",
    position = position_dodge(width = 0.75),
    size = 3,
    vjust = 0,
    color = "black"
  ) +
  scale_fill_manual(
    values = c("mreu" = "#4169E1",
               "control" = "#FAFAD2")
  )+
  scale_color_manual(
    values = c("mreu" = "#4169E1",
               "control" = "#EEDD82")
  )+
  labs(
```

```
    #title = "Time to rob by bee species and treatment",
    x = NULL,
    y = "Time to Rob"
  ) +
  #scale_fill_discrete(name = "Treatment") +
  #scale_color_discrete(name = "Treatment") +
  scale_shape_manual(name = "Treatment", values = c(16, 17), guide = "none") +
  scale_y_continuous(expand = c(0, 0), limits = c(0, 40)) +
  labs(
    #title = "Time to rob by bee species and treatment",
    x = NULL,
    y = "Time to Rob"
  ) +
  theme_minimal() +
  theme(
    panel.grid.major = element_blank(),
    panel.grid.minor = element_blank(),
    axis.line = element_line(color = "black"),
    axis.line.x = element_line(),
    axis.line.y = element_line(),
    )

timetorob <- timetorob_box_species +
  coord_cartesian(ylim = c(0, 40))

timetorob
```

```
## Warning: Removed 1 row containing non-finite outside the scale range
## (`stat_boxplot()`).
```

```
## Warning: Removed 1 row containing non-finite outside the scale range
## (`stat_summary()`).
```

```
## Warning: Removed 1 row containing missing values or values outside the scale range
## (`geom_point()`).
```




```
ggsave("results/timetorob_pub.pdf", plot = timetorob, width = 8, height = 6)
```

```
## Warning: Removed 1 row containing non-finite outside the scale range
## (‘stat_boxplot()‘).
```

```
## Warning: Removed 1 row containing non-finite outside the scale range
## (‘stat_summary()‘).
```

```
## Warning: Removed 1 row containing missing values or values outside the scale range
## (‘geom_point()‘).
```

ANOVA time to rob on bee species (1) and plant species (2) - in both cases treatment is highly significant, but interaction between either plant or bee species and treatment is not. Suggests instead that only the nectar treatment significantly different

```
anova_result <- aov(timetorob ~ species + treatment, data = robbed_flowers)
summary(anova_result)
```

```
##              Df Sum Sq Mean Sq F value   Pr(>F)
## species       1     29    29.2   1.249    0.264
## treatment     1    455   455.1  19.479 1.23e-05 ***
## Residuals   544  12711    23.4
## ---
## Signif. codes:  0 ’***’ 0.001 ’**’ 0.01 ’*’ 0.05 ’.’ 0.1 ’ ’ 1
```

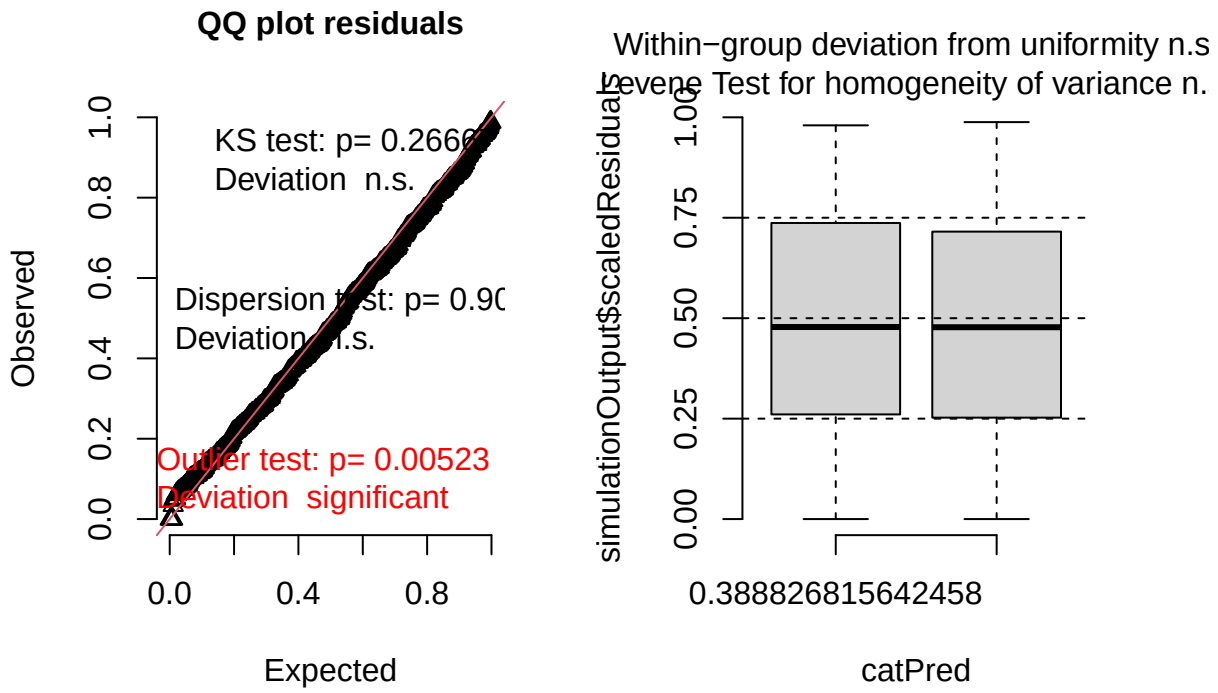
```r
anova_result_plant <- aov(timetorob ~ plantsp + treatment, data = robbed_flowers)
summary(anova_result_plant)
```

```
##              Df Sum Sq Mean Sq F value   Pr(>F)
## plantsp       1     45    44.6   1.918    0.167
## treatment     1    491   491.4  21.115 5.38e-06 ***
## Residuals   544  12659    23.3
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

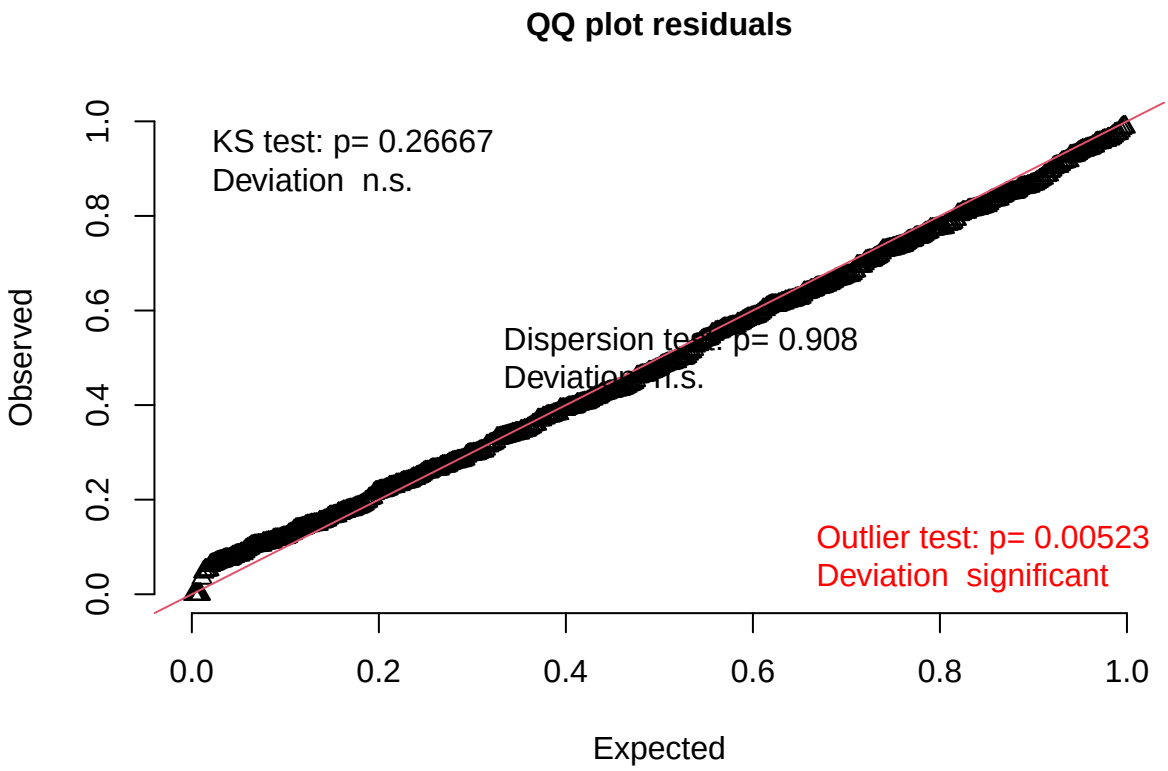
```r
eps <- 0.01
m_rob <- lmer (log(timetorob+eps) ~ treatment + (1| sample) + (1| plantsp) + (1| flower.) + (1| species)
summary(m_rob)
```

```
## Linear mixed model fit by REML. t-tests use Satterthwaite's method [
## lmerModLmerTest]
## Formula: log(timetorob + eps) ~ treatment + (1 | sample) + (1 | plantsp) +
##     (1 | flower.) + (1 | species)
##    Data: robbed_flowers
##
## REML criterion at convergence: 1575.3
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -5.4847 -0.5964 -0.0079  0.6355  2.3439
##
## Random effects:
##  Groups   Name        Variance Std.Dev.
##  sample   (Intercept) 0.074140 0.27229
##  flower.  (Intercept) 0.009685 0.09841
##  species  (Intercept) 0.010882 0.10431
##  plantsp  (Intercept) 0.021478 0.14655
##  Residual             0.971239 0.98551
## Number of obs: 547, groups:  sample, 49; flower., 31; species, 2; plantsp, 2
##
## Fixed effects:
##                Estimate Std. Error       df t value Pr(>|t|)
## (Intercept)     1.3433     0.1616   0.8769   8.314 0.097210 .
## treatmentmreu  -0.3941     0.1046 195.6156  -3.766 0.000219 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation of Fixed Effects:
##             (Intr)
## treatmentmr -0.413
```

```r
res <- simulateResiduals(m_rob, n = 1000)
plot(res)                # full diagnostic plot
```

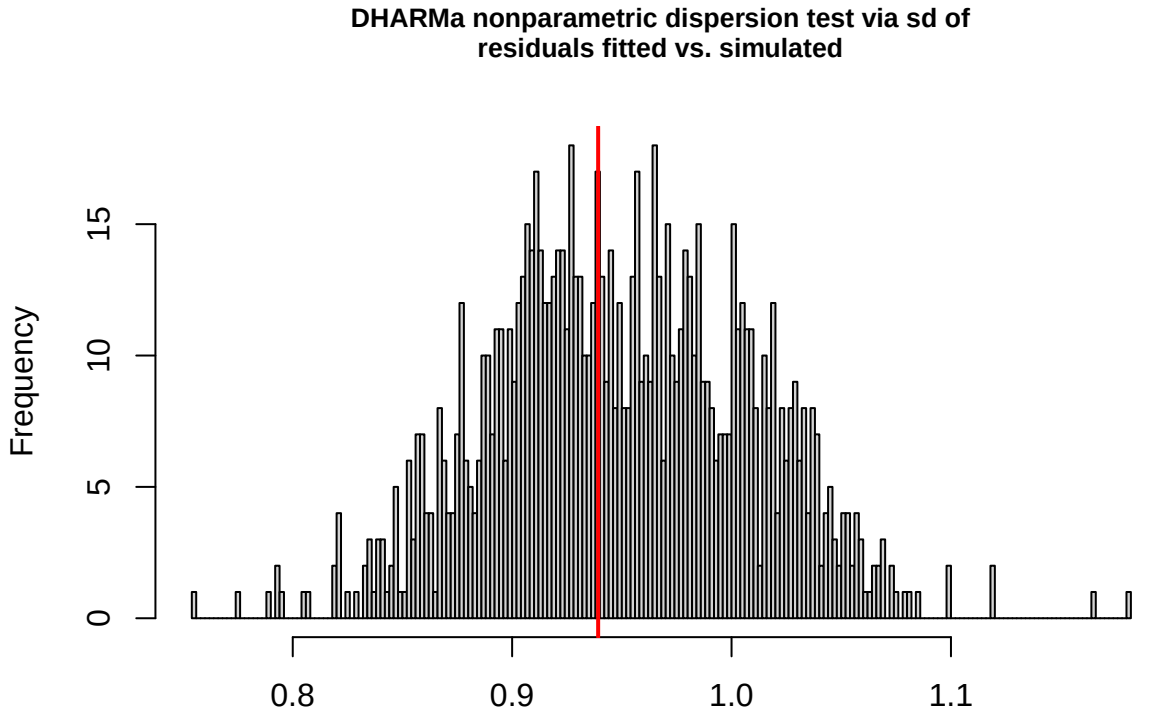



```r
testUniformity(res)       # residuals should be uniform
```



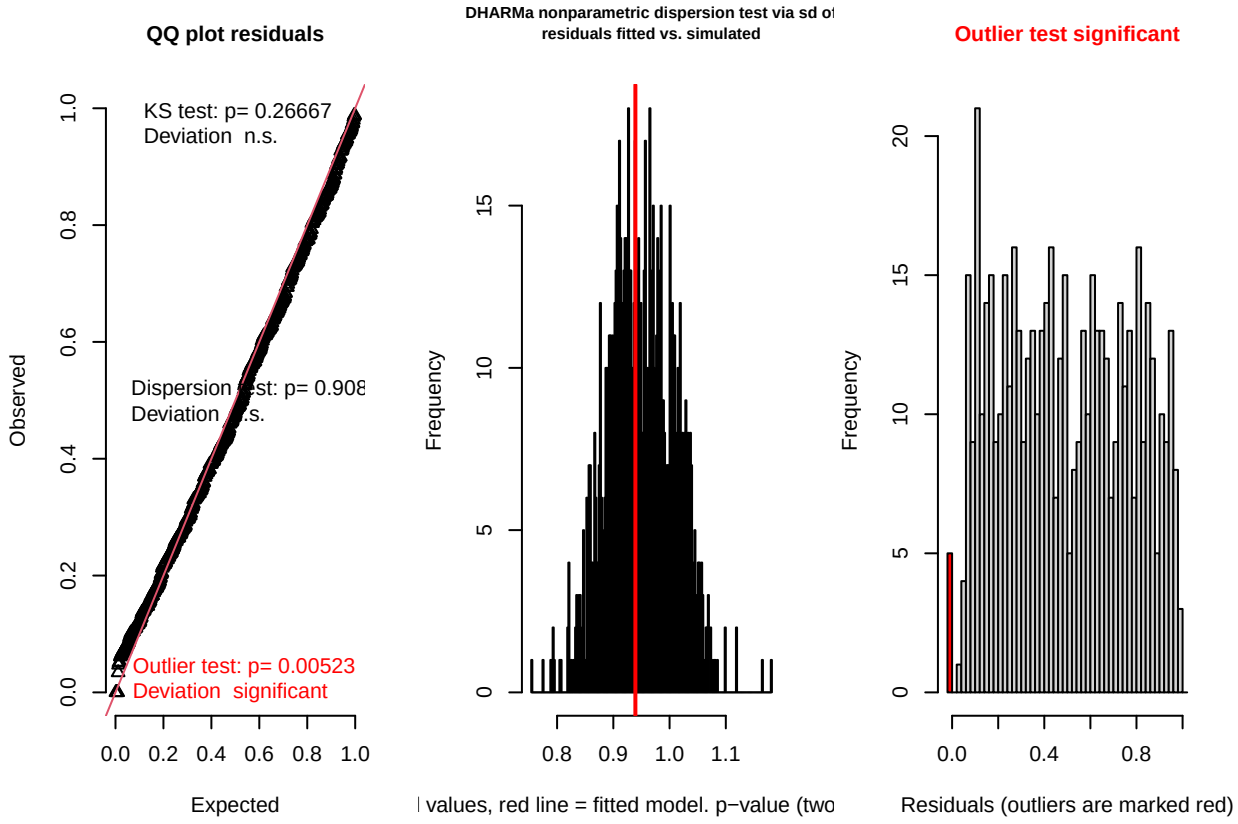

```
##
##  Asymptotic one-sample Kolmogorov-Smirnov test
##
## data:  simulationOutput$scaledResiduals
## D = 0.04289, p-value = 0.2667
## alternative hypothesis: two-sided
```

```
testDispersion(res)       # checks for over/under dispersion
```




```
##
##  DHARMa nonparametric dispersion test via sd of residuals fitted vs.
##  simulated
##
## data:  simulationOutput
## dispersion = 0.98916, p-value = 0.908
## alternative hypothesis: two.sided
```

```
testResiduals(res)        # KS test for uniformity
```




```
## $uniformity
##
##  Asymptotic one-sample Kolmogorov-Smirnov test
##
## data:  simulationOutput$scaledResiduals
## D = 0.04289, p-value = 0.2667
## alternative hypothesis: two-sided
##
##
## $dispersion
##
##  DHARMa nonparametric dispersion test via sd of residuals fitted vs.
##  simulated
##
## data:  simulationOutput
## dispersion = 0.98916, p-value = 0.908
## alternative hypothesis: two.sided
##
##
## $outliers
##
##  DHARMa outlier test based on exact binomial test with approximate
##  expectations
##
## data:  simulationOutput
## outliers at both margin(s) = 5, observations = 547, p-value = 0.005235
```

```
## alternative hypothesis: true probability of success is not equal to 0.001998002
## 95 percent confidence interval:
##  0.002974454 0.021201740
## sample estimates:
## frequency of outliers (expected: 0.001998001998002 )
##                                         0.009140768
```

```
m_simpler <- lmer(log(timetorob + eps) ~ treatment +
                  (1|sample) + (1|flower.),
                  data = robbed_flowers)

anova(m_simpler, m_rob)  # compare fits with likelihood ratio test
```

```
## refitting model(s) with ML (instead of REML)

## Data: robbed_flowers
## Models:
## m_simpler: log(timetorob + eps) ~ treatment + (1 | sample) + (1 | flower.)
## m_rob: log(timetorob + eps) ~ treatment + (1 | sample) + (1 | plantsp) + (1 | flower.) + (1 | species
##           npar    AIC    BIC  logLik -2*log(L) Chisq Df Pr(>Chisq)
## m_simpler    5 1579.3 1600.9 -784.67    1569.3
## m_rob        7 1583.3 1613.5 -784.67    1569.3     0  2          1
```

```
m_final <- lmer(log(timetorob + eps) ~ treatment + plantsp + species +
                (1|sample) + (1|flower.),
                data = robbed_flowers)

summary(m_final)
```

```
## Linear mixed model fit by REML. t-tests use Satterthwaite's method [
## lmerModLmerTest]
## Formula: log(timetorob + eps) ~ treatment + plantsp + species + (1 | sample) +
##     (1 | flower.)
##    Data: robbed_flowers
##
## REML criterion at convergence: 1574.9
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -5.4214 -0.5981  0.0023  0.6554  2.3538
##
## Random effects:
##  Groups   Name        Variance Std.Dev.
##  sample   (Intercept) 0.072468 0.26920
##  flower.  (Intercept) 0.009033 0.09504
##  Residual             0.971469 0.98563
## Number of obs: 547, groups:  sample, 49; flower., 31
##
## Fixed effects:
##                   Estimate Std. Error       df t value Pr(>|t|)
## (Intercept)         1.3490     0.1101  45.7070  12.247  4.9e-16 ***
## treatmentmreu      -0.3818     0.1048 198.6570  -3.644 0.000343 ***
```

```
## plantspmertensia    0.3322     0.1654  38.5995   2.009 0.051572 .
## speciesflavifrons  -0.2561     0.1413  33.6391  -1.813 0.078789 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation of Fixed Effects:
##             (Intr) trtmnt plntsp
## treatmentmr -0.546
## plntspmrtns -0.021  0.045
## spcsflvfrns -0.450 -0.120 -0.505
```

```
emm <- emmeans(m_final, ~ treatment)
regrid(emm, transform = "response")
```

```
##  treatment response    SE   df lower.CL upper.CL
##  control       3.99 0.420 61.6     3.15     4.83
##  mreu          2.72 0.249 45.7     2.22     3.22
##
## Results are averaged over the levels of: plantsp, species
## Degrees-of-freedom method: inherited from kenward-roger when re-gridding
## Confidence level used: 0.95
```

```
emm_full <- emmeans(m_final, ~treatment | plantsp * species)

emm_full <- emmeans(m_final, ~ treatment | plantsp * species,
                    at = list(plantsp = unique(robbed_flowers$plantsp),
                              species = unique(robbed_flowers$species)))
regrid(emm_full, transform = "response")
```

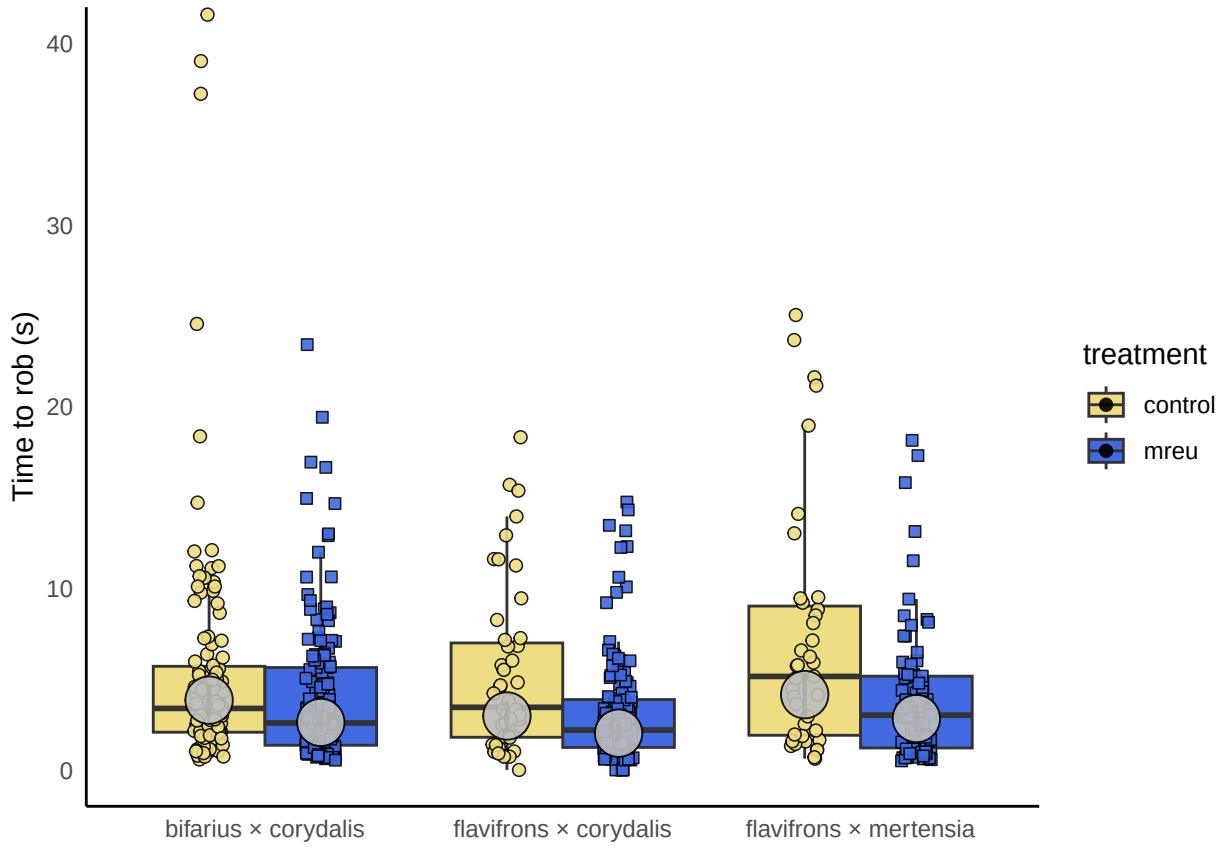
```
## plantsp = corydalis, species = bifarius:
##  treatment response    SE   df lower.CL upper.CL
##  control       3.84 0.429 48.5     2.98     4.71
##  mreu          2.62 0.272 45.6     2.07     3.17
##
## plantsp = mertensia, species = bifarius:
##  treatment response    SE   df lower.CL upper.CL
##  control       5.36 1.070 43.4     3.21     7.51
##  mreu          3.66 0.727 44.7     2.19     5.12
##
## plantsp = corydalis, species = flavifrons:
##  treatment response    SE   df lower.CL upper.CL
##  control       2.97 0.405 56.1     2.16     3.78
##  mreu          2.03 0.233 37.9     1.55     2.50
##
## plantsp = mertensia, species = flavifrons:
##  treatment response    SE   df lower.CL upper.CL
##  control       4.15 0.611 61.7     2.93     5.37
##  mreu          2.83 0.380 49.7     2.07     3.59
##
## Degrees-of-freedom method: inherited from kenward-roger when re-gridding
## Confidence level used: 0.95
```

```r
# 1) Fit a model that includes species
m_full <- lmer(
  log(timetorob + eps) ~ treatment + plantsp + species +
    (1|sample) + (1|flower.),
  data = robbed_flowers
)

# 2) Estimated means by treatment within plantsp x species
emm_full <- emmeans(m_full, ~ treatment | plantsp * species)

# 3) Back transform to seconds, then coerce to a data frame
emm_df <- as.data.frame(regrid(emm_full, transform = "response"))

# 4) Keep only combos that actually exist in your data
obs <- robbed_flowers %>% distinct(plantsp, species)
emm_df_obs <- emm_df %>% semi_join(obs, by = c("plantsp","species"))

emm_df_obs
```

```
## plantsp = corydalis, species = bifarius:
##  treatment response        SE    df lower.CL upper.CL
##  control   3.843434 0.4290790 48.53 2.980955 4.705914
##  mreu      2.620439 0.2722524 45.62 2.072300 3.168577
##
## plantsp = corydalis, species = flavifrons:
##  treatment response        SE    df lower.CL upper.CL
##  control   2.972762 0.4051733 56.06 2.161121 3.784404
##  mreu      2.026099 0.2330316 37.88 1.554303 2.497895
##
## plantsp = mertensia, species = flavifrons:
##  treatment response        SE    df lower.CL upper.CL
##  control   4.148045 0.6107058 61.71 2.927148 5.368941
##  mreu      2.828373 0.3795151 49.70 2.065980 3.590765
##
## Degrees-of-freedom method: inherited from kenward-roger when re-gridding
## Confidence level used: 0.95
```

```r
# 1) Make a readable x group for both raw and emmeans
robbed_flowers <- robbed_flowers %>%
  mutate(group = interaction(species, plantsp, sep = " × "))

# 2) Get emmeans by treatment within plant × bee, back to seconds, keep only observed combos
emm_full <- emmeans(m_full, ~ treatment | plantsp * species)
emm_df <- as.data.frame(regrid(emm_full, transform = "response"))

obs <- robbed_flowers %>% distinct(plantsp, species)
emm_df_obs <- emm_df %>%
  inner_join(obs, by = c("plantsp","species")) %>%
  mutate(group = interaction(species, plantsp, sep = " × "))

pd <- position_dodge(width = 0.75)

timetorob_emm <- ggplot(robbed_flowers,
```

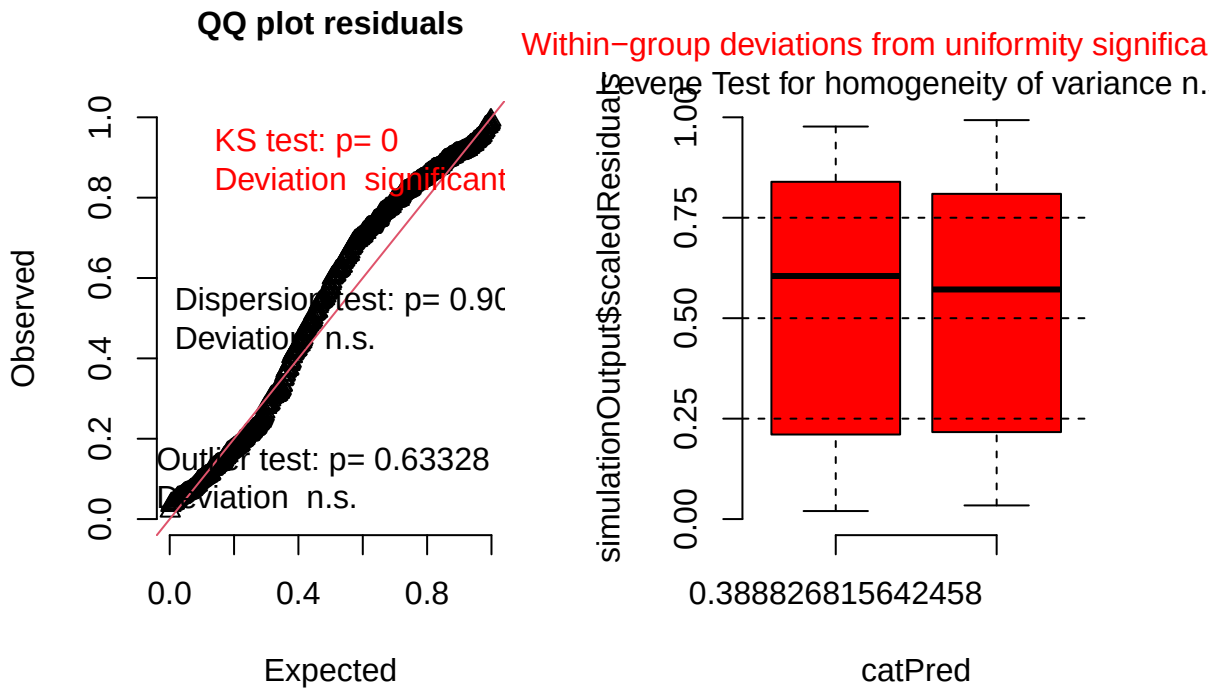
```
                        aes(x = group, y = timetorob, fill = treatment)) +
  geom_boxplot(outlier.shape = NA, position = pd) +
  geom_point(aes(fill = treatment, shape = treatment),
             position = position_jitterdodge(jitter.width = 0.1, dodge.width = 0.75),
             size = 2, alpha = 0.9, color = "black", stroke = 0.4) +
  geom_pointrange(data = emm_df_obs,
                  aes(x = group, y = response, ymin = lower.CL, ymax = upper.CL,
                      group = treatment),      # key line
                  position = pd,              # same dodge as boxplot
                  shape = 21, fill = "gray", color = "black",
                  stroke = 0.5, alpha = 0.9,
                  size = 2, linewidth = 0.8) +
  scale_fill_manual(values = c("mreu" = "#4169E1", "control" = "#EEDD82")) +
  scale_shape_manual(values = c("control" = 21, "mreu" = 22), guide = "none") +
  coord_cartesian(ylim = c(0, 40)) +
  labs(x = NULL, y = "Time to rob (s)") +
  theme_minimal() +
  theme(panel.grid.major = element_blank(),
        panel.grid.minor = element_blank(),
        axis.line = element_line(color = "black"))

timetorob_emm
```



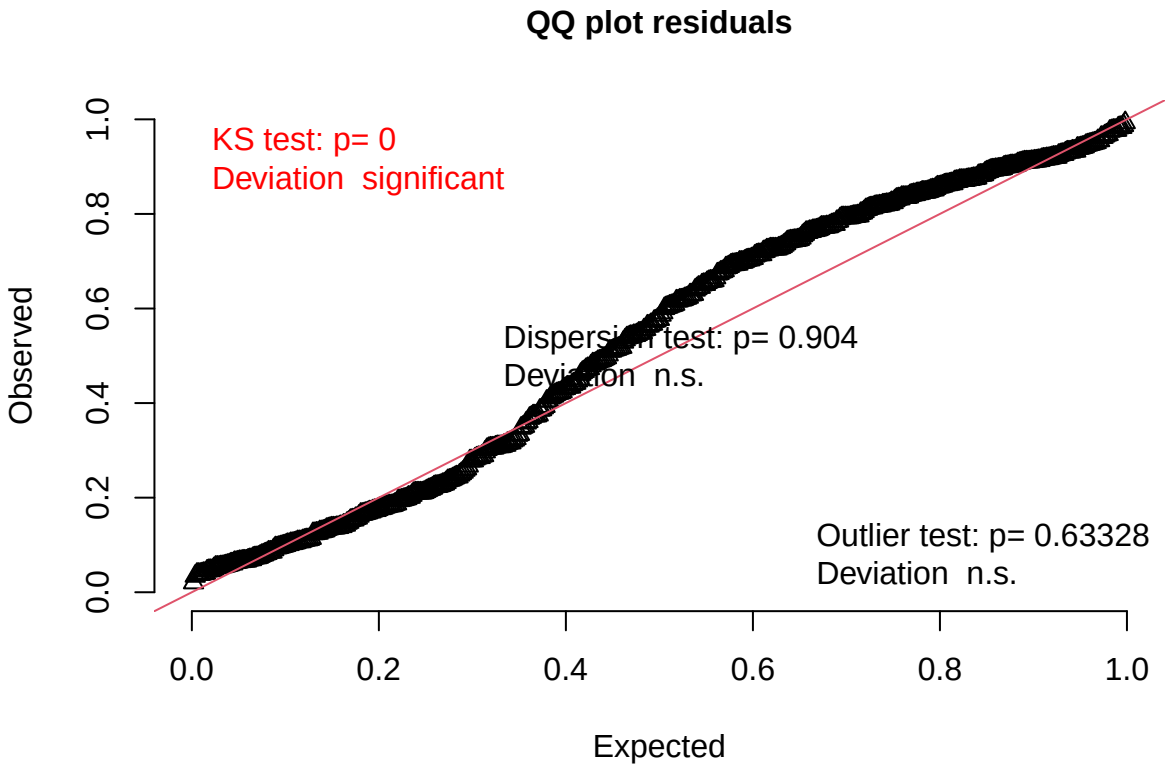

```
m_feed <- lmer (log(timefeeding+eps) ~ treatment + (1| sample) + (1| plantsp) + (1| flower.) + (1| speci
summary(m_feed)
```

```
## Linear mixed model fit by REML. t-tests use Satterthwaite's method [
## lmerModLmerTest]
## Formula: log(timefeeding + eps) ~ treatment + (1 | sample) + (1 | plantsp) +
##     (1 | flower.) + (1 | species)
##    Data: robbed_flowers
##
## REML criterion at convergence: 1550.9
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -2.3373 -0.8452  0.1215  0.7419  2.5009
##
## Random effects:
##  Groups   Name        Variance Std.Dev.
##  sample   (Intercept) 0.038813 0.19701
##  flower.  (Intercept) 0.149059 0.38608
##  species  (Intercept) 0.001508 0.03883
##  plantsp  (Intercept) 0.000914 0.03023
##  Residual             0.898334 0.94780
## Number of obs: 547, groups:  sample, 49; flower., 31; species, 2; plantsp, 2
##
## Fixed effects:
##                Estimate Std. Error        df t value Pr(>|t|)
## (Intercept)     1.48487    0.11947   5.57325  12.429 2.83e-05 ***
## treatmentmreu  -0.02728    0.09618 154.99290  -0.284    0.777
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation of Fixed Effects:
##             (Intr)
## treatmentmr -0.497
```

```
res_feed <- simulateResiduals(m_feed, n = 1000)
plot(res_feed)                 # full diagnostic plot
```

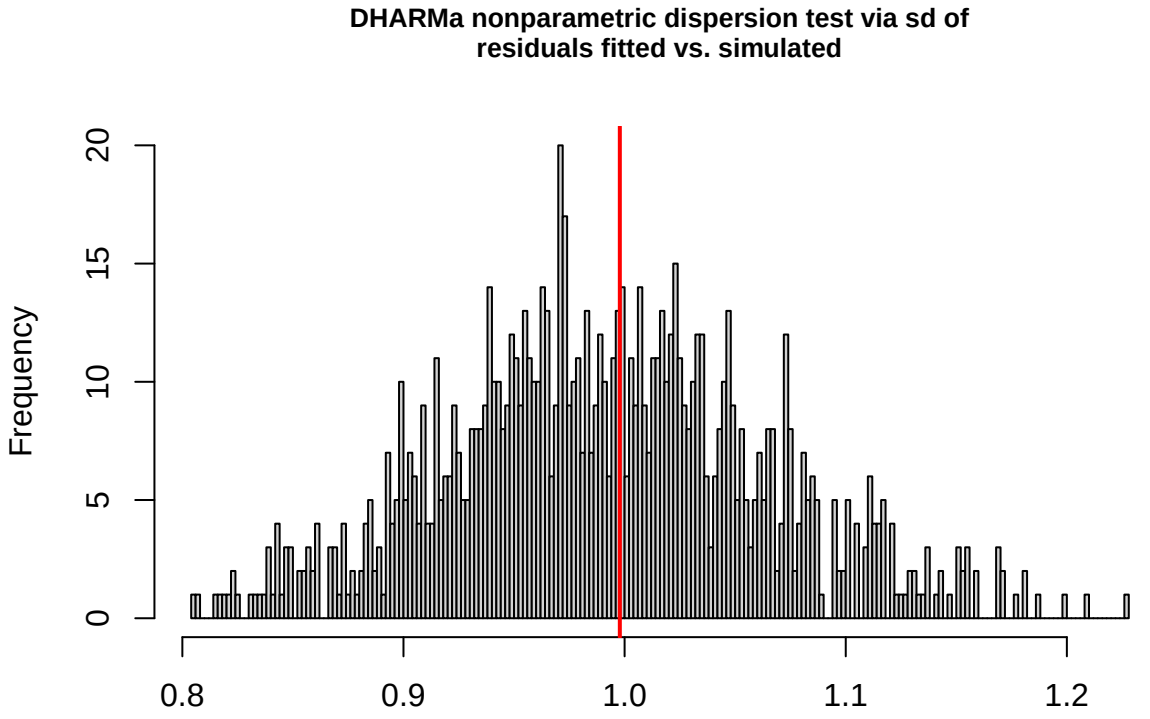



```
testUniformity(res_feed)       # residuals should be uniform
```



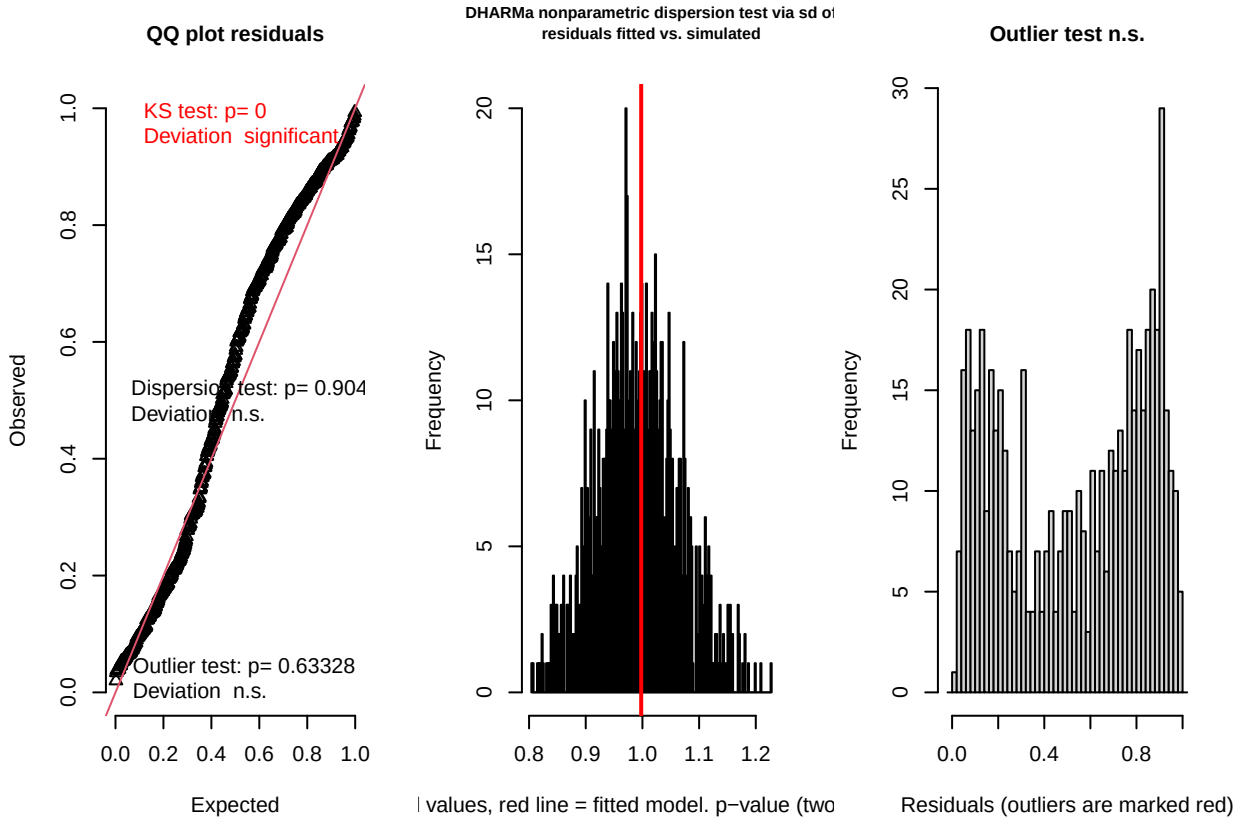

```
##
##  Asymptotic one-sample Kolmogorov-Smirnov test
##
## data:  simulationOutput$scaledResiduals
## D = 0.11613, p-value = 7.822e-07
## alternative hypothesis: two-sided
```

```
testDispersion(res_feed)       # checks for over/under dispersion
```




```
##
##  DHARMa nonparametric dispersion test via sd of residuals fitted vs.
##  simulated
##
## data:  simulationOutput
## dispersion = 1.0073, p-value = 0.904
## alternative hypothesis: two.sided
```

```
testResiduals(res_feed)        # KS test for uniformity
```




```
## $uniformity
##
##  Asymptotic one-sample Kolmogorov-Smirnov test
##
## data:  simulationOutput$scaledResiduals
## D = 0.11613, p-value = 7.822e-07
## alternative hypothesis: two-sided
##
##
## $dispersion
##
##  DHARMa nonparametric dispersion test via sd of residuals fitted vs.
##  simulated
##
## data:  simulationOutput
## dispersion = 1.0073, p-value = 0.904
## alternative hypothesis: two.sided
##
##
## $outliers
##
##  DHARMa outlier test based on exact binomial test with approximate
##  expectations
##
## data:  simulationOutput
## outliers at both margin(s) = 0, observations = 547, p-value = 0.6333
```

```
## alternative hypothesis: true probability of success is not equal to 0.001998002
## 95 percent confidence interval:
##  0.000000000 0.006721149
## sample estimates:
## frequency of outliers (expected: 0.001998001998002 )
##                                                    0
```

```
m_simple_feed <- lmer(log(timefeeding + eps) ~ treatment +
                  (1|sample) + (1|flower.),
                  data = robbed_flowers)

anova(m_simple_feed, m_feed)  # compare fits with likelihood ratio test
```

```
## refitting model(s) with ML (instead of REML)
```

```
## Data: robbed_flowers
## Models:
## m_simple_feed: log(timefeeding + eps) ~ treatment + (1 | sample) + (1 | flower.)
## m_feed: log(timefeeding + eps) ~ treatment + (1 | sample) + (1 | plantsp) + (1 | flower.) + (1 | spe
##               npar    AIC    BIC  logLik -2*log(L) Chisq Df Pr(>Chisq)
## m_simple_feed    5 1555.3 1576.8 -772.64    1545.3
## m_feed           7 1559.3 1589.4 -772.63    1545.3 0.002  2      0.999
```

```
m_final_feed <- lmer(log(timefeeding + eps) ~ treatment + plantsp + species +
                (1|sample) + (1|flower.),
                data = robbed_flowers)

summary(m_final_feed)
```

```
## Linear mixed model fit by REML. t-tests use Satterthwaite's method [
## lmerModLmerTest]
## Formula: log(timefeeding + eps) ~ treatment + plantsp + species + (1 |
##     sample) + (1 | flower.)
##    Data: robbed_flowers
##
## REML criterion at convergence: 1553.9
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -2.4078 -0.8581  0.1051  0.7212  2.5243
##
## Random effects:
##  Groups   Name        Variance Std.Dev.
##  sample   (Intercept) 0.04099  0.2025
##  flower.  (Intercept) 0.14595  0.3820
##  Residual             0.89875  0.9480
## Number of obs: 547, groups:  sample, 49; flower., 31
##
## Fixed effects:
##                    Estimate Std. Error        df t value Pr(>|t|)
## (Intercept)         1.42529    0.12345  54.07662  11.545 3.25e-16 ***
## treatmentmreu      -0.03522    0.09736 158.43629  -0.362    0.718
```

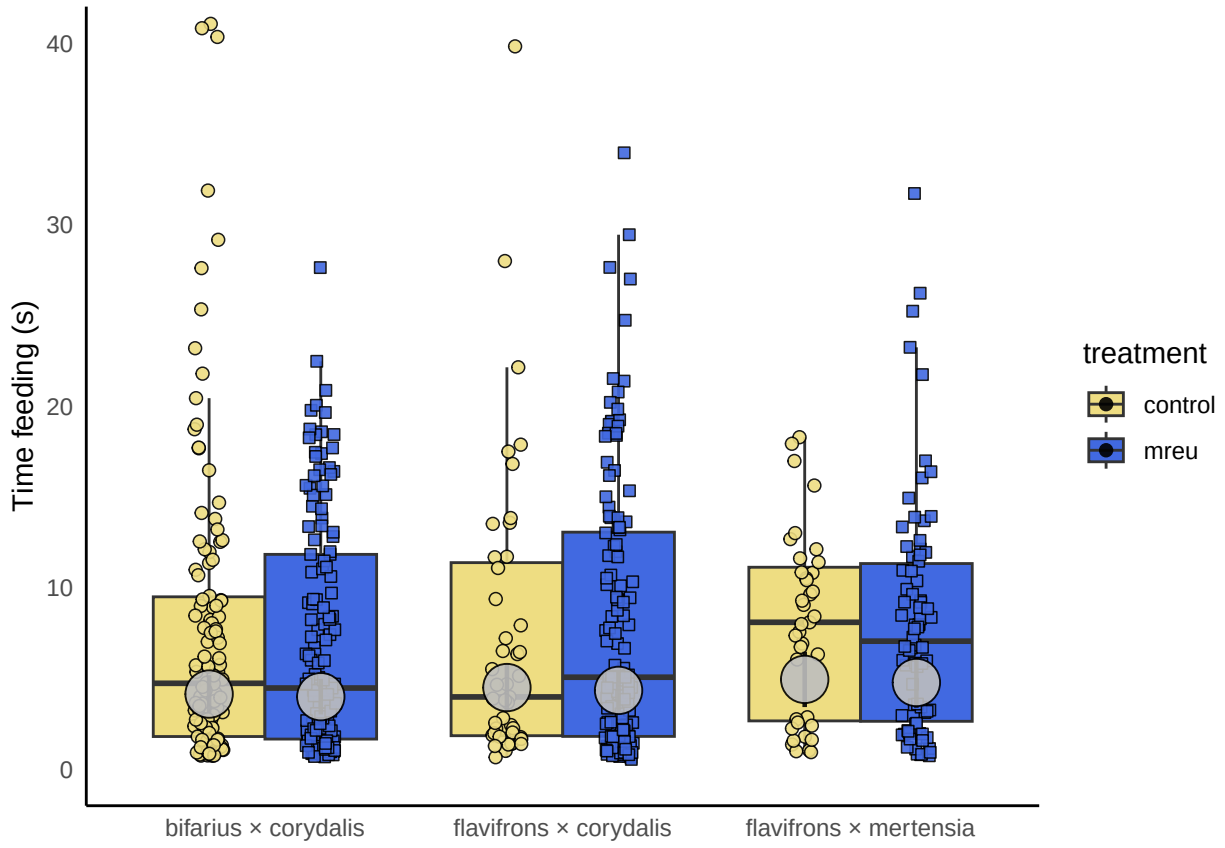
```
## plantspmertensia    0.09511    0.14487  39.17081   0.656    0.515
## speciesflavifrons   0.08234    0.12240  33.32582   0.673    0.506
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation of Fixed Effects:
##             (Intr) trtmnt plntsp
## treatmentmr -0.429
## plntspmrtns  0.001  0.048
## spcsflvfrns -0.340 -0.134 -0.502
```

```
# 2) Estimated means by treatment within plantsp x species
emm_full_feed <- emmeans(m_final_feed, ~ treatment | plantsp * species)

# 3) Back transform to seconds, then coerce to a data frame
emm_df_feed <- as.data.frame(regrid(emm_full_feed, transform = "response"))

# 4) Keep only combos that actually exist in your data
obs_feed <- robbed_flowers %>% distinct(plantsp, species)
emm_df_obs_feed <- emm_df_feed %>% semi_join(obs_feed, by = c("plantsp","species"))%>% mutate(group = in

emm_df_obs_feed
```

```
##   treatment   plantsp    species response        SE       df lower.CL upper.CL
## 1   control corydalis   bifarius 4.149057 0.5187797 51.68344 3.107898 5.190217
## 2      mreu corydalis   bifarius 4.005144 0.4867052 48.01444 3.026565 4.983723
## 3   control corydalis flavifrons 4.506004 0.6438422 62.37899 3.219137 5.792870
## 4      mreu corydalis flavifrons 4.349739 0.5541286 45.90561 3.234275 5.465204
## 5   control mertensia flavifrons 4.956592 0.7642041 70.72813 3.432711 6.480473
## 6      mreu mertensia flavifrons 4.784736 0.6935994 60.28136 3.397464 6.172009
##                     group
## 1   bifarius × corydalis
## 2   bifarius × corydalis
## 3 flavifrons × corydalis
## 4 flavifrons × corydalis
## 5 flavifrons × mertensia
## 6 flavifrons × mertensia
```

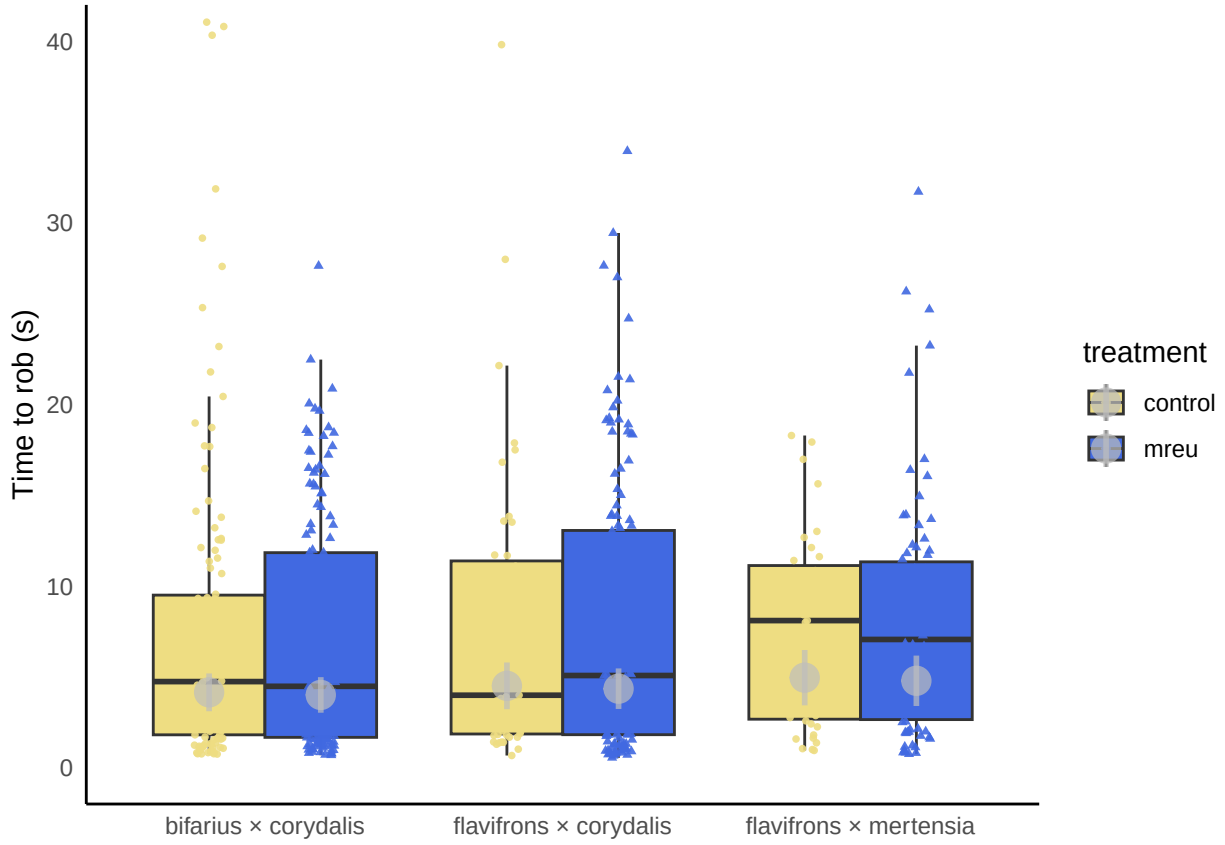
```
pd <- position_dodge(width = 0.75)

timefeeding_emm <- ggplot(robbed_flowers,
                          aes(x = group, y = timefeeding, fill = treatment)) +
  geom_boxplot(outlier.shape = NA, position = pd) +

  # jittered datapoints with black outlines
  geom_point(
    aes(fill = treatment, shape = treatment),
    position = position_jitterdodge(jitter.width = 0.1, dodge.width = 0.75),
    size = 2, alpha = 0.9,
    color = "black",      # outline
    stroke = 0.4
  ) +
```

```
  # emmeans layer: gray filled points with black outline, aligned by treatment
  geom_pointrange(
    data = emm_df_obs_feed,
    aes(x = group, y = response, ymin = lower.CL, ymax = upper.CL,
        group = treatment),
    position = pd,
    shape = 21,           # filled circle allows fill and outline
    fill = "gray",
    color = "black",      # outline and CI line
    stroke = 0.5,
    alpha = 0.9,
    size = 2,
    linewidth = 0.8
  ) +

  scale_fill_manual(values = c("mreu" = "#4169E1", "control" = "#EEDD82")) +
  scale_shape_manual(values = c("control" = 21, "mreu" = 22), guide = "none") +

  coord_cartesian(ylim = c(0, 40)) +
  labs(x = NULL, y = "Time feeding (s)") +
  theme_minimal() +
  theme(panel.grid.major = element_blank(),
        panel.grid.minor = element_blank(),
        axis.line = element_line(color = "black"))

timefeeding_emm
```




```
# 3) Plot: raw boxplot + jitter + emmeans point with 95% CI
pd <- position_dodge(width = 0.75)

timefeeding_emm <- ggplot(robbed_flowers,
                          aes(x = group, y = timefeeding, fill = treatment)) +
  geom_boxplot(outlier.shape = NA, position = pd) +
  geom_point(aes(color = treatment, shape = treatment),
             position = position_jitterdodge(jitter.width = 0.1, dodge.width = 0.75),
             size = 1, alpha = 0.9) +

  # emmeans layer: big gray points + thick CI lines
  geom_pointrange(data = emm_df_obs_feed,
                  aes(x = group, y = response, ymin = lower.CL, ymax = upper.CL),
                  color = "gray",
                  alpha = 0.7,
                  position = pd,
                  size = 1,           # bigger point
                  linewidth = 1) +      # thicker CI lines

  scale_fill_manual(values = c("mreu" = "#4169E1", "control" = "#EEDD82")) +
  scale_color_manual(values = c("mreu" = "#4169E1", "control" = "#EEDD82")) +
  scale_shape_manual(values = c("control" = 16, "mreu" = 17), guide = "none") +

  coord_cartesian(ylim = c(0, 40)) +
  labs(x = NULL, y = "Time to rob (s)") +
  theme_minimal() +
```

```
  theme(panel.grid.major = element_blank(),
        panel.grid.minor = element_blank(),
        axis.line = element_line(color = "black"))

timefeeding_emm
```



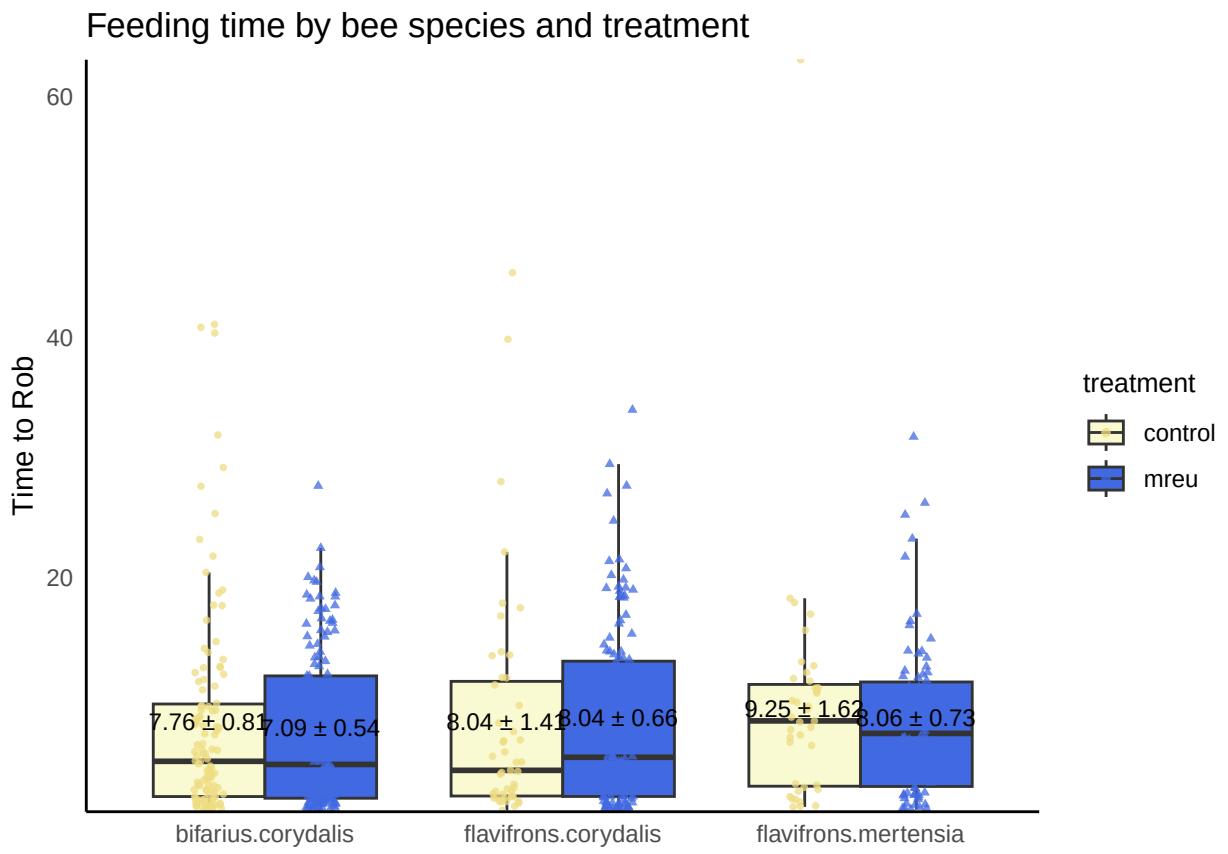
To make sure this significance holds, I did wilcox test only on the mean time to rob. Again, significantly different ($p = < 0.05$)

```
treatment_control <- subset(robbed_flowers, treatment == "control") #control subset
treatment_mreu <- subset(robbed_flowers, treatment == "mreu") #inoculated flowers subset

# Perform the Wilcoxon rank-sum test
wilcox_test_result <- wilcox.test(treatment_control$timetorob, treatment_mreu$timetorob) #wilcox test

# Print the test result
print(wilcox_test_result) #wilcox result - highly significant difference of mean time to rob between tr
```

```
##
##  Wilcoxon rank sum test with continuity correction
##
## data:  treatment_control$timetorob and treatment_mreu$timetorob
## W = 42406, p-value = 1.502e-05
## alternative hypothesis: true location shift is not equal to 0
```

We do the same analysis for feeding time and show that there is no significant difference between treatments on the feeding time of bees. It seems that yeast does not have a significant flavor(?) difference or not not different enough or nasty enough for bees to care. They spend the same amount of time feeding on sterile or yeast inoculated nectar

```
feedingtime_box_species <- ggplot(robbed_flowers, aes(x = interaction(species, plantsp), y = timefeeding
  geom_boxplot(outlier.shape = NA, position = position_dodge(width = 0.75)) +
  geom_point(position = position_jitterdodge(jitter.width = 0.1, dodge.width = 0.75),
             aes(shape = treatment, color = treatment),
             size = 1, alpha = 0.75) +
  stat_summary(
    fun.data = function(x) {
      m <- mean(x)
      se <- sd(x) / sqrt(length(x))
      data.frame(y = m + se * -.5, label = sprintf("%.2f ± %.2f", m, se))
    },
    geom = "text",
    position = position_dodge(width = 0.75),
    size = 3,
    vjust = 0,
    color = "black"
  ) +
  scale_fill_manual(
    values = c("mreu" = "#4169E1",
               "control" = "#FAFAD2")
  )+
  scale_color_manual(
    values = c("mreu" = "#4169E1",
               "control" = "#EEDD82")
  )+
  labs(
    title = "Feeding time by bee species and treatment",
    x = NULL,
    y = "Time to Rob"
  ) +
  #scale_fill_discrete(name = "Treatment") +
  #scale_color_discrete(name = "Treatment") +
  scale_shape_manual(name = "Treatment", values = c(16, 17), guide = "none") +
  scale_y_continuous(expand = c(0, 0)) +  # Ensures bars rest on the x-axis line
  theme_minimal() +
  theme(
    panel.grid.major = element_blank(),
    panel.grid.minor = element_blank(),
    axis.line = element_line(color = "black"),
    axis.line.x = element_line(),
    axis.line.y = element_line(),
    legend.title = element_text(size = 10),
    legend.text = element_text(size = 9)
  )

feedingtime_box_species
```




```
ggsave("results/feeding_pub.pdf", plot = feedingtime_box_species, width = 8, height = 6)
```

ANOVA and wilcox test confirm. No difference on time spent feeding depending on treatment of nectar.

```
anova_result_feeding <- aov(timefeeding ~ species + treatment, data = robbed_flowers)

# Print the ANOVA table
summary(anova_result_feeding)
```

```
##              Df Sum Sq Mean Sq F value Pr(>F)
## species       1     91   90.71   1.518  0.218
## treatment     1     46   46.07   0.771  0.380
## Residuals   544  32510   59.76
```

```
# Perform the Wilcoxon rank-sum test
wilcox_test_result_feeding <- wilcox.test(treatment_control$timefeeding, treatment_mreu$timefeeding)

# Print the test result
print(wilcox_test_result_feeding)
```

```
##
##  Wilcoxon rank sum test with continuity correction
##
## data:  treatment_control$timefeeding and treatment_mreu$timefeeding
```

```
## W = 34333, p-value = 0.8367
## alternative hypothesis: true location shift is not equal to 0
```

This included all plant species combined. Below I break down everything per plant species (Corydalis and Mertensia)

First subset data

```
corydalis <- subset(robbed_flowers, plantsp == "corydalis")
mertensia <- subset(robbed_flowers, plantsp == "mertensia")
```

Since for Corydalis we have 2 bee species, I still did ANOVA. For Mertensia, there is only one bee species so makes more sense to do wilcox. In the case of Corydalis, the two bee species take different times to find the robbing hole (p = 0.03), while time to find the hole for different treatments is stil highly significant (p =0.001). We could do different analysis for each bee species, if worth it.

```
anova_result_robbing_corydalis <- aov(timetorob ~ species + treatment, data = corydalis)
# Print the ANOVA table
summary(anova_result_robbing_corydalis)
```

```
##              Df Sum Sq Mean Sq F value Pr(>F)
## species       1     98   98.34   4.309 0.0385 *
## treatment     1    230  230.28  10.091 0.0016 **
## Residuals   427   9744   22.82
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
anova_result_feeding_corydalis <- aov(timefeeding ~ species + treatment, data = corydalis)
# Print the ANOVA table
summary(anova_result_feeding_corydalis)
```

```
##              Df Sum Sq Mean Sq F value Pr(>F)
## species       1     44   43.61   0.733  0.392
## treatment     1     19   18.60   0.313  0.576
## Residuals   427  25394   59.47
```

For Corydalis, significant difference both for species and treatment, but not the interaction. So, bee species show difference in the time it takes to find a robbing hole (p = 0.03), and significantly different according to treatment (p = 0.001).

Wilcox test for mertenisa, time to rob significantly different between treatments (P = 0.006)

```
wilcox_mertensia_timetorob <- wilcox.test(timetorob ~ treatment, data = mertensia)
wilcox_mertensia_timetorob
```

```
##
##  Wilcoxon rank sum test with continuity correction
##
## data:  timetorob by treatment
## W = 1988, p-value = 0.006991
## alternative hypothesis: true location shift is not equal to 0
```

```r
legit_summary <- corydalis %>%
  group_by(unique_name) %>%
  summarise(trytolegit_binary = sum(trytolegit_binary))

print(legit_summary)
```

```
## # A tibble: 35 x 2
##    unique_name                    trytolegit_binary
##    <chr>                                      <dbl>
##  1 Aug-01_corydalis_3_bifarius                    2
##  2 Aug-01_corydalis_7_bifarius                    1
##  3 Aug-01_corydalis_8_bifarius                    2
##  4 Aug-01_corydalis_9_flavifrons                 11
##  5 Aug-02_corydalis_12_flavifrons                 2
##  6 Aug-02_corydalis_1_flavifrons                  0
##  7 Aug-02_corydalis_2_bifarius                    1
##  8 Aug-02_corydalis_7_bifarius                    2
##  9 Aug-02_corydalis_8_bifarius                    2
## 10 Aug-03_corydalis_11_bifarius                   3
## # i 25 more rows
```

```r
visit_summary <- corydalis %>%
  group_by(unique_name) %>%
  summarise(total_visits = n(),
            legit_visits = sum(trytolegit == 'yes'))

visit_summary
```

```
## # A tibble: 35 x 3
##    unique_name                    total_visits legit_visits
##    <chr>                                 <int>        <int>
##  1 Aug-01_corydalis_3_bifarius              15            2
##  2 Aug-01_corydalis_7_bifarius              11            1
##  3 Aug-01_corydalis_8_bifarius              10            2
##  4 Aug-01_corydalis_9_flavifrons            15           11
##  5 Aug-02_corydalis_12_flavifrons           15            2
##  6 Aug-02_corydalis_1_flavifrons             1            0
##  7 Aug-02_corydalis_2_bifarius               9            1
##  8 Aug-02_corydalis_7_bifarius              12            2
##  9 Aug-02_corydalis_8_bifarius              19            2
## 10 Aug-03_corydalis_11_bifarius              9            3
## # i 25 more rows
```

```r
y_tube <- read.csv("data/y_tube_ori_2025.csv", header = T)

table(y_tube$Initial_Choice)
```

```
##
##  C  T
##  9 13
```

```r
chisq.test(table(y_tube$Initial_Choice))
```

```
##
##  Chi-squared test for given probabilities
##
## data:  table(y_tube$Initial_Choice)
## X-squared = 0.72727, df = 1, p-value = 0.3938
```

```r
t <- t.test(y_tube$Yeast_time, y_tube$Control_time, paired = TRUE, alternative = "greater")
t
```

```
##
##  Paired t-test
##
## data:  y_tube$Yeast_time and y_tube$Control_time
## t = 2.4177, df = 21, p-value = 0.0124
## alternative hypothesis: true mean difference is greater than 0
## 95 percent confidence interval:
##  12.99907      Inf
## sample estimates:
## mean difference
##        45.09091
```

```r
chisq <- chisq.test(table(y_tube$Initial_Choice))
phi <- sqrt(chisq$statistic / sum(chisq$observed))
phi
```

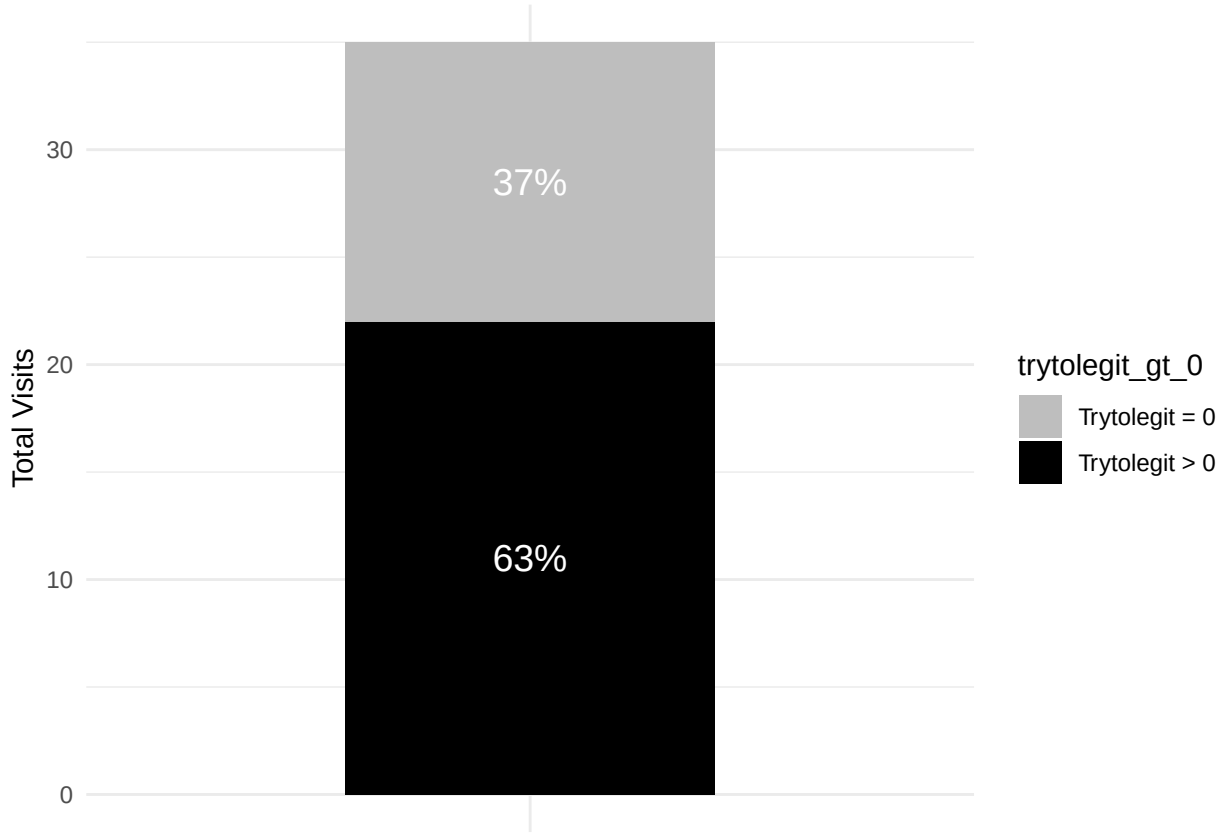
```
## X-squared
## 0.1818182
```

```r
library(effsize)
cohen_d <- cohen.d(y_tube$Yeast_time, y_tube$Control_time, paired = TRUE, hedges.correction = TRUE)
cohen_d
```

```
##
## Hedges's g
##
## g estimate: 0.859222 (large)
## 95 percent confidence interval:
##      lower      upper
## 0.02003978 1.69840420
```

```r
corydalis_bifarius <- subset(corydalis, species == "bifarius")
corydalis_flavifrons <- subset(corydalis, species == "flavifrons")

summary_corydalis <- corydalis %>%
  group_by(unique_name) %>%
  summarise(total_visits = n(),
            legit_visits = sum(trytolegit == 'yes'))
```

```r
# Create a new column indicating whether trytolegit_binary > 0
legit_summary <- legit_summary %>%
  mutate(trytolegit_gt_0 = ifelse(trytolegit_binary > 0, "Trytolegit > 0", "Trytolegit = 0"))

# Summarize the data by the new column
summary_data <- legit_summary %>%
  group_by(trytolegit_gt_0) %>%
  summarise(total_visits = n())

# Calculate total visits
total_visits <- sum(summary_data$total_visits)

# Calculate percentage of visits for each category
summary_data <- summary_data %>%
  mutate(percentage = total_visits / sum(total_visits))

# Plot the barplot
ggplot(summary_data, aes(x = "", y = total_visits, fill = trytolegit_gt_0)) +
  geom_bar(stat = "identity", width = 0.5) +
  geom_text(aes(label = scales::percent(percentage)),
            position = position_stack(vjust = 0.5),
            color = "white", size = 5) +
  labs(x = NULL, y = "Total Visits") +
  scale_fill_manual(values = c("Trytolegit = 0" = "gray", "Trytolegit > 0" = "black")) +
  theme_minimal() +
  theme(axis.text.x=element_blank())  # Hide x-axis label
```




```
legit_summary_mert <- mertensia %>%
  group_by(unique_name) %>%
  summarise(trytolegit_binary = sum(trytolegit_binary))

print(legit_summary_mert)
```

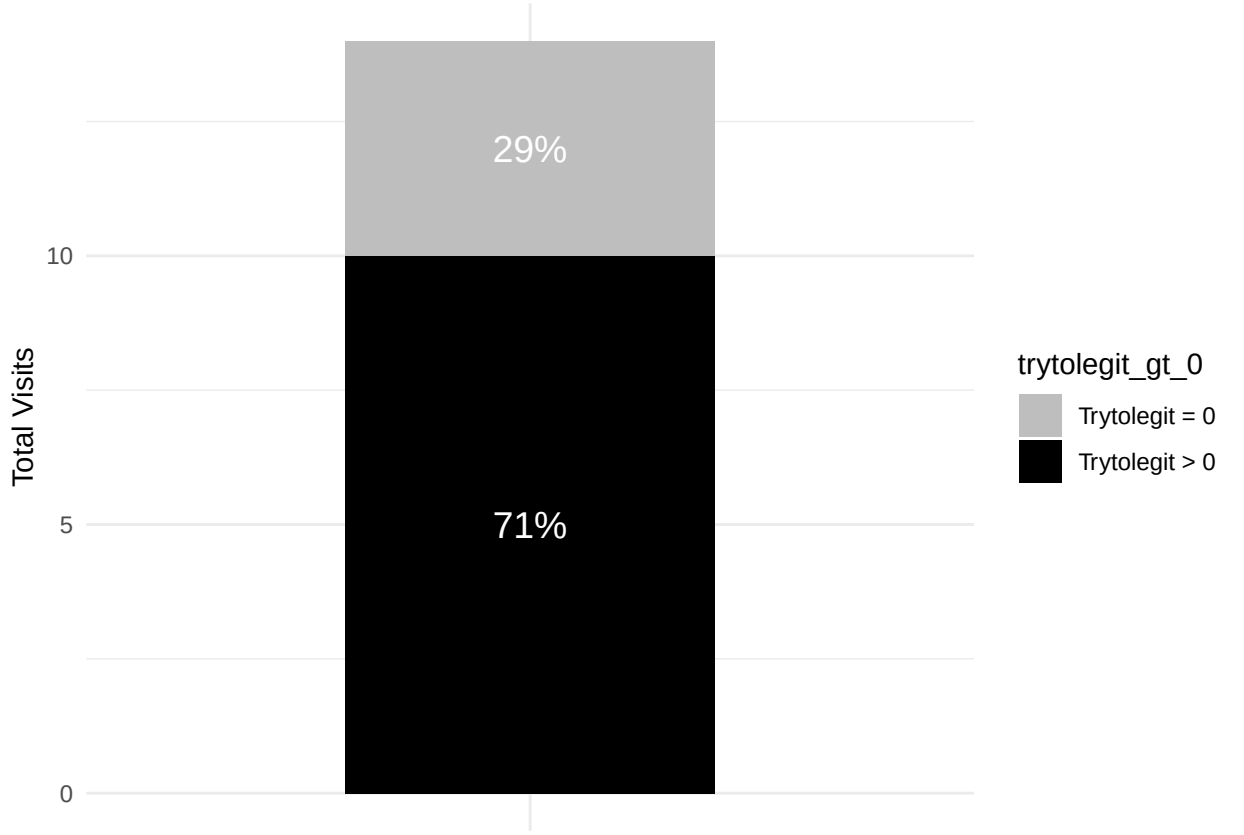
```
## # A tibble: 14 x 2
##    unique_name                    trytolegit_binary
##    <chr>                                      <dbl>
##  1 Aug-10_mertensia_10_flavifrons                 4
##  2 Aug-10_mertensia_3_flavifrons                  1
##  3 Aug-10_mertensia_7_flavifrons                  4
##  4 Aug-10_mertensia_8_flavifrons                  3
##  5 Aug-11_mertensia_1_flavifrons                  5
##  6 Aug-11_mertensia_3_flavifrons                  2
##  7 Aug-14_mertensia_10_flavifrons                 0
##  8 Aug-14_mertensia_1_flavifrons                  2
##  9 Aug-14_mertensia_2_flavifrons                  1
## 10 Aug-15_mertensia_1_flavifrons                  0
## 11 Aug-15_mertensia_2_flavifrons                  0
## 12 Aug-15_mertensia_4_flavifrons                  5
## 13 Aug-15_mertensia_6_flavifrons                  2
## 14 Aug-15_mertensia_8_flavifrons                  0
```

```
visit_summary_mert <- mertensia %>%
  group_by(unique_name) %>%
  summarise(total_visits = n(),
            legit_visits = sum(trytolegit == 'yes'))

visit_summary_mert
```

```
## # A tibble: 14 x 3
##    unique_name                    total_visits legit_visits
##    <chr>                                 <int>        <int>
##  1 Aug-10_mertensia_10_flavifrons           14            4
##  2 Aug-10_mertensia_3_flavifrons             1            1
##  3 Aug-10_mertensia_7_flavifrons            23            4
##  4 Aug-10_mertensia_8_flavifrons            18            3
##  5 Aug-11_mertensia_1_flavifrons             9            5
##  6 Aug-11_mertensia_3_flavifrons             4            2
##  7 Aug-14_mertensia_10_flavifrons            1            0
##  8 Aug-14_mertensia_1_flavifrons            13            2
##  9 Aug-14_mertensia_2_flavifrons             2            1
## 10 Aug-15_mertensia_1_flavifrons             9            0
## 11 Aug-15_mertensia_2_flavifrons             4            0
## 12 Aug-15_mertensia_4_flavifrons            10            5
## 13 Aug-15_mertensia_6_flavifrons             3            2
## 14 Aug-15_mertensia_8_flavifrons             6            0
```

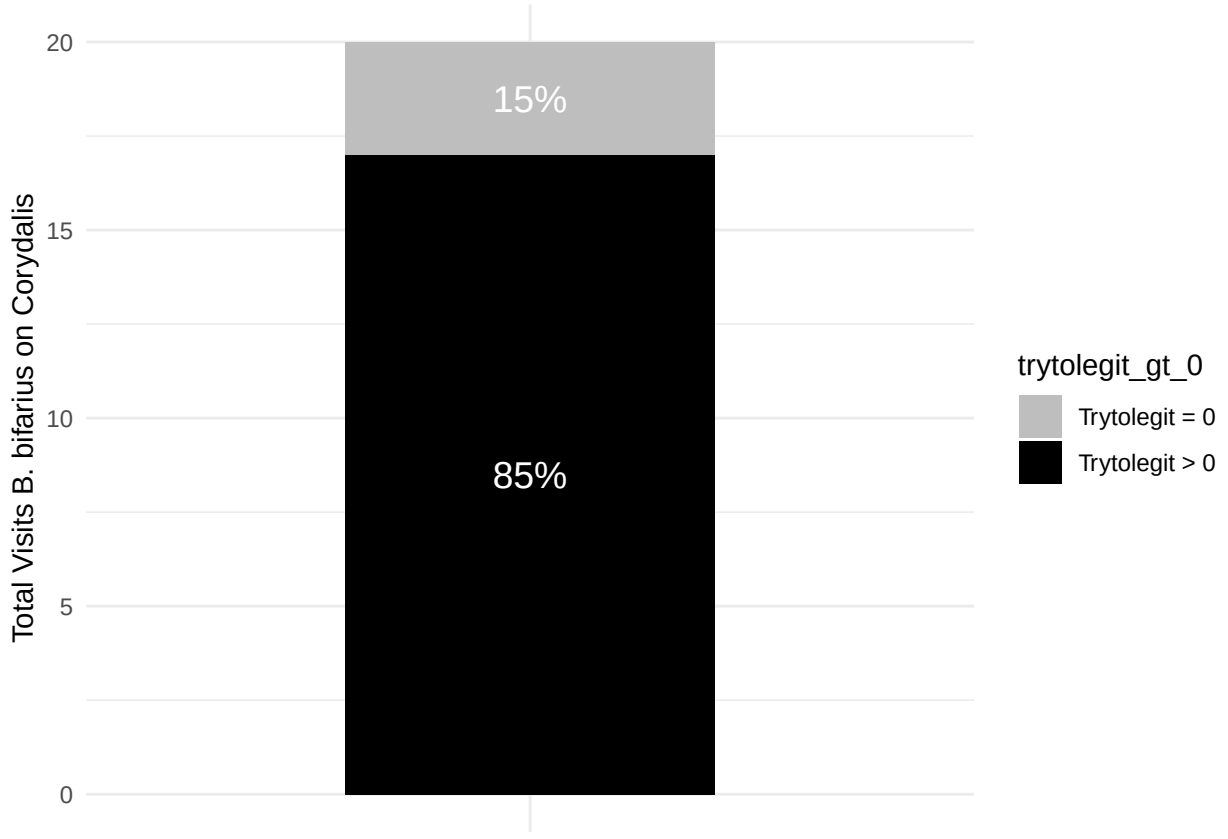
```
summary_mertensia <- mertensia %>%
  group_by(unique_name) %>%
  summarise(total_visits = n(),
            legit_visits = sum(trytolegit == 'yes'))


# Create a new column indicating whether trytolegit_binary > 0
legit_summary_mert <- legit_summary_mert %>%
  mutate(trytolegit_gt_0 = ifelse(trytolegit_binary > 0, "Trytolegit > 0", "Trytolegit = 0"))

# Summarize the data by the new column
summary_data_mert <- legit_summary_mert %>%
  group_by(trytolegit_gt_0) %>%
  summarise(total_visits = n())

# Calculate total visits
total_visits_mert <- sum(summary_data_mert$total_visits)

# Calculate percentage of visits for each category
summary_data_mert <- summary_data_mert %>%
  mutate(percentage = total_visits / sum(total_visits))

# Plot the barplot
ggplot(summary_data_mert, aes(x = "", y = total_visits, fill = trytolegit_gt_0)) +
  geom_bar(stat = "identity", width = 0.5) +
  geom_text(aes(label = scales::percent(percentage)),
            position = position_stack(vjust = 0.5),
            color = "white", size = 5) +
```

```
  labs(x = NULL, y = "Total Visits") +
  scale_fill_manual(values = c("Trytolegit = 0" = "gray", "Trytolegit > 0" = "black")) +
  theme_minimal() +
  theme(axis.text.x=element_blank())  # Hide x-axis label
```



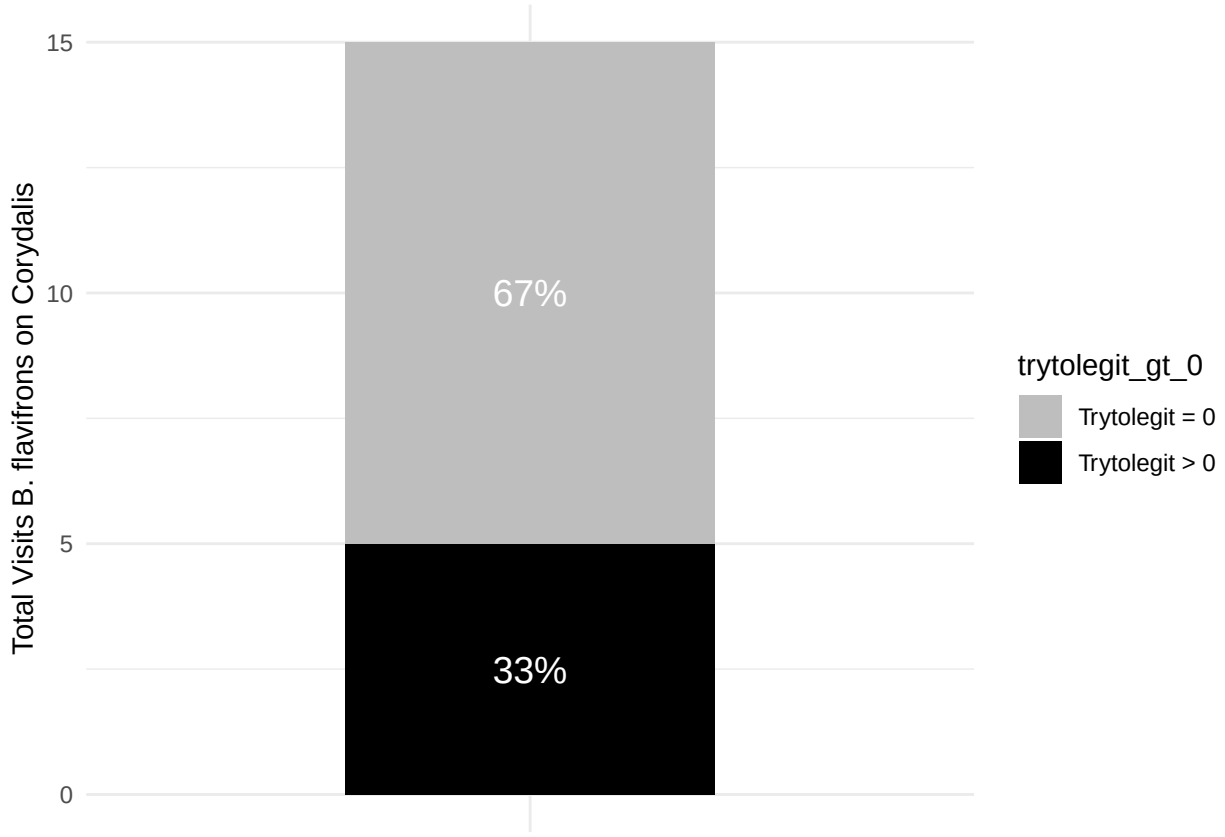

```
summary_cory_bif <- corydalis_bifarius %>%
  group_by(unique_name) %>%
  summarise(total_visits = n(),
            legit_visits = sum(trytolegit == 'yes'))
legit_summary_corybif <- corydalis_bifarius %>%
  group_by(unique_name) %>%
  summarise(trytolegit_binary = sum(trytolegit_binary))

visit_summary_corybif <- corydalis_bifarius %>%
  group_by(unique_name) %>%
  summarise(total_visits = n(),
            legit_visits = sum(trytolegit == 'yes'))


# Create a new column indicating whether trytolegit_binary > 0
legit_summary_corybif <- legit_summary_corybif %>%
  mutate(trytolegit_gt_0 = ifelse(trytolegit_binary > 0, "Trytolegit > 0", "Trytolegit = 0"))

# Summarize the data by the new column
summary_data_corybif <- legit_summary_corybif %>%
```

```
  group_by(trytolegit_gt_0) %>%
  summarise(total_visits = n())

# Calculate total visits
total_visits_corybif <- sum(summary_data_corybif$total_visits)

# Calculate percentage of visits for each category
summary_data_corybif <- summary_data_corybif %>%
  mutate(percentage = total_visits / sum(total_visits))

# Plot the barplot
ggplot(summary_data_corybif, aes(x = "", y = total_visits, fill = trytolegit_gt_0)) +
  geom_bar(stat = "identity", width = 0.5) +
  geom_text(aes(label = scales::percent(percentage)),
            position = position_stack(vjust = 0.5),
            color = "white", size = 5) +
  labs(x = NULL, y = "Total Visits B. bifarius on Corydalis") +
  scale_fill_manual(values = c("Trytolegit = 0" = "gray", "Trytolegit > 0" = "black")) +
  theme_minimal() +
  theme(axis.text.x=element_blank())  # Hide x-axis label
```




```
summary_cory_flav <- corydalis_flavifrons %>%
 group_by(unique_name) %>%
 summarise(total_visits = n(),
           legit_visits = sum(trytolegit == 'yes'))
```

```r
legit_summary_coryflav <- corydalis_flavifrons %>%
  group_by(unique_name) %>%
  summarise(trytolegit_binary = sum(trytolegit_binary))

visit_summary_coryflav <- corydalis_flavifrons %>%
  group_by(unique_name) %>%
  summarise(total_visits = n(),
            legit_visits = sum(trytolegit == 'yes'))


# Create a new column indicating whether trytolegit_binary > 0
legit_summary_coryflav <- legit_summary_coryflav %>%
  mutate(trytolegit_gt_0 = ifelse(trytolegit_binary > 0, "Trytolegit > 0", "Trytolegit = 0"))

# Summarize the data by the new column
summary_data_coryflav <- legit_summary_coryflav %>%
  group_by(trytolegit_gt_0) %>%
  summarise(total_visits = n())

# Calculate total visits
total_visits_coryflav <- sum(summary_data_coryflav$total_visits)

# Calculate percentage of visits for each category
summary_data_coryflav <- summary_data_coryflav %>%
  mutate(percentage = total_visits / sum(total_visits))

# Plot the barplot
ggplot(summary_data_coryflav, aes(x = "", y = total_visits, fill = trytolegit_gt_0)) +
  geom_bar(stat = "identity", width = 0.5) +
  geom_text(aes(label = scales::percent(percentage)),
            position = position_stack(vjust = 0.5),
            color = "white", size = 5) +
  labs(x = NULL, y = "Total Visits B. flavifrons on Corydalis") +
  scale_fill_manual(values = c("Trytolegit = 0" = "gray", "Trytolegit > 0" = "black")) +
  theme_minimal() +
  theme(axis.text.x=element_blank())  # Hide x-axis label
```




```
library(patchwork)
```

```
##
## Attaching package: 'patchwork'

## The following object is masked from 'package:MASS':
##
##     area
```

```
library(cowplot)
```

```
##
## Attaching package: 'cowplot'

## The following object is masked from 'package:patchwork':
##
##     align_plots

## The following object is masked from 'package:lubridate':
##
##     stamp
```

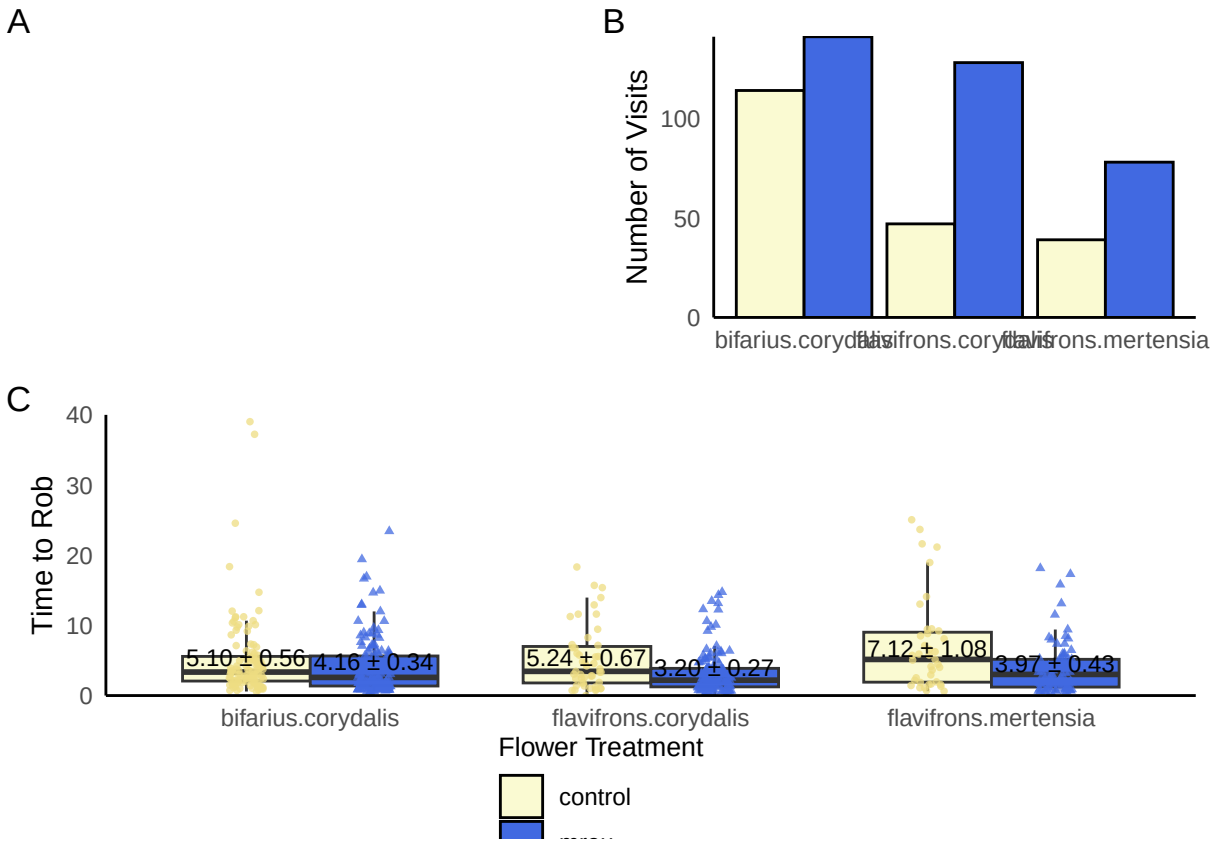
```r
blank_plot <- ggplot()+
  theme_void()+
  theme(panel.background = element_rect(fill = "white", color = NA))

timetorob_legend <- timetorob + theme(legend.position = "none")
plot_legend <- plot + theme(legend.position = "none")

combined_plot <- (blank_plot | plot_legend) / timetorob_legend +
plot_layout(widths = c(1,9), heights = c(1,1))+
  plot_annotation(tag_levels = "A")

# Extract legend from plot1
legend <- cowplot::get_legend(
  plot + theme(legend.position = "right")
)

# Use cowplot to assemble final plot with legend below

final_plot <- cowplot::plot_grid(
  combined_plot,
  legend,
  ncol = 1,
  rel_heights = c(1, 0.1)
)
```

```
## Warning: Removed 1 row containing non-finite outside the scale range
## (`stat_boxplot()`).
```

```
## Warning: Removed 1 row containing non-finite outside the scale range
## (`stat_summary()`).
```

```
## Warning: Removed 1 row containing missing values or values outside the scale range
## (`geom_point()`).
```

```r
final_plot
```




```
ggsave("results/combined_plot.pdf", final_plot, width = 7, height = 8, units = "in")
```