Statistical analysis of larva and female blowfly preference

May 2023

The following code was used to assess the preference of larvae and females using random simulations. It is part of the study entitled "*Choice assays using two interacting factors in larvae and females blowflies*" by Vanessa A. S. Cunha, Sophie Tandonnet, Diniz Lima Ferreira, Andre V. Rodrigues and Tatiana Teixeira Torres.

Preliminary steps

Packages used

```
library(nlme)
library(stats)
library(lme4)
```

```
## Loading required package: Matrix

##
## Attaching package: 'lme4'

## The following object is masked from 'package:nlme':
##
##     lmList
```

```
library(ggplot2)
library(ggpubr)
```

Setting working directory and reading data frame

```
#setting the working directory (change this to the directory containing the data)
##setwd("/path/to/JOVE/directory/")

#reading the data frame
dt_total <- read.table("Supplemental File S2 - Sheet1.tsv", sep = "\t", header = T)
head(dt_total)
```

```
##            Specie stage Batch Replicate Rotten_cold Rotten_hot Fresh_cold
## 1 Lucilia cuprina    L3     1         1           4          1          0
## 2 Lucilia cuprina    L3     1         2           4          1          0
## 3 Lucilia cuprina    L3     1         3           4          1          0
## 4 Lucilia cuprina    L3     1         4           3          2          0
## 5 Lucilia cuprina    L3     1         5           5          0          0
## 6 Lucilia cuprina    L3     1         6           5          0          0
##   Fresh_hot Did_not_choose Total
## 1         0              0     5
## 2         0              0     5
## 3         0              0     5
## 4         0               0     5
## 5         0               0     5
## 6         0               0     5
```

Assessing larval preference

Creating the subset for larvae and calculating the Preference Indexes (PIs)

```
#setting the larvae database
dt_larvae <- subset(dt_total, dt_total$stage == "L3")

#selecting only the necessary columns for the analysis:
# (stage and the choices: Rotten_cold, Rotten_hot, Fresh_cold, Fresh_hot, Did_not_choose)
↪
dt_larvae <- dt_larvae[,c(2,5:9)]

#Calculating and creating the column for the preference index for temperature
dt_larvae$PI_temp <- (dt_larvae$Rotten_hot + dt_larvae$Fresh_hot)/(dt_larvae$Rotten_hot +
↪  dt_larvae$Fresh_hot + dt_larvae$Rotten_cold + dt_larvae$Fresh_cold)

#Calculating and creating the column for the preference index for meat type
dt_larvae$PI_meat <-  (dt_larvae$Fresh_hot + dt_larvae$Fresh_cold)/(dt_larvae$Rotten_hot
↪  + dt_larvae$Fresh_hot + dt_larvae$Rotten_cold + dt_larvae$Fresh_cold)
```

Calculating the probability of larvae not choosing

```
#Determining the probability of larvae not choosing
total_larvae <- sum(c(dt_larvae$Rotten_cold, dt_larvae$Rotten_hot, dt_larvae$Fresh_cold,
↪  dt_larvae$Fresh_hot, dt_larvae$Did_not_choose))

not_choose_larvae <- sum(dt_larvae$Did_not_choose)

(prob_not_choose_larvae <- not_choose_larvae/total_larvae)
```

```
## [1] 0.04444444
```

Simulating the larvae data

```
#seed used to generate the results of the article.
# NOTE: This can be changed or not used at all.
set.seed(1)
```

Creating data frames to contain results of the simulations

```
#Creating an empty data frame to hold the simulated data for larvae
dt_sim_larvae <- data.frame(1:nrow(dt_larvae))
dt_sim_larvae$X1.nrow.dt_larvae. <- NULL #Deleting column created in empty dataset

#Classifying the stage of the simulated data for later comparison (we put the '!' just to
↪  order it first alphabetically)
dt_sim_larvae$stage <- "!random"
```

```r
#Setting the values as zero for now
dt_sim_larvae$Rotten_cold <- 0
dt_sim_larvae$Rotten_hot <- 0
dt_sim_larvae$Fresh_cold <- 0
dt_sim_larvae$Fresh_hot <- 0
dt_sim_larvae$Did_not_choose <- 0

#Creating the data frames for the statistical metrics
results_meat <- data.frame(1:1000)
results_meat$X1.1000 <- NULL
results_meat$mean <- 0
results_meat$est_sim <- 0
results_meat$std_sim <- 0
results_meat$t_sim <- 0
results_meat$p_sim <- 0
results_meat$est_real <- 0
results_meat$std_real <- 0
results_meat$t_real <- 0
results_meat$p_real <- 0

results_temp <- data.frame(1:1000)
results_temp$X1.1000 <- NULL
results_temp$mean <- 0
results_temp$est_sim <- 0
results_temp$std_sim <- 0
results_temp$t_sim <- 0
results_temp$p_sim <- 0
results_temp$est_real <- 0
results_temp$std_real <- 0
results_temp$t_real <- 0
results_temp$p_real <- 0
```

Running the simulation 1000 times

```r
for(z in 1:1000)
{
  #reseting the simulated data frame
  dt_sim_larvae$Rotten_cold <- 0
  dt_sim_larvae$Rotten_hot <- 0
  dt_sim_larvae$Fresh_cold <- 0
  dt_sim_larvae$Fresh_hot <- 0
  dt_sim_larvae$Did_not_choose <- 0
  #running the simulation for one simulated data frame (= the number of rows of the data
  ↪  frame)
  for(i in 1:nrow(dt_sim_larvae))
  {
    #simulating the randomized choice of the 5 larvae (5 values from 1 to 0)
    choice <- runif(5, 0, 1)
    #analyzing the choice of each larva
    for(j in 1:5)
    {
      #if the choice value for this larva is in the probability range of "not choosing",
      ↪  we counted as a "Did_not_choose"
      if(choice[j] >= 1-prob_not_choose_larvae)
      {
        dt_sim_larvae$Did_not_choose[i] <- dt_sim_larvae$Did_not_choose[i] + 1
      }
      #else, verify if the choice is in the probability range of any other choice
      ↪  category and count as the respective category (Rotten_cold, Rotten_hot,
      ↪  Fresh_cold, Fresh_hot)
      else
      {
        choice[j] <- choice[j]/(1-prob_not_choose_larvae)
        if(choice[j] <= 0.25)
        {
          dt_sim_larvae$Rotten_cold[i] <- dt_sim_larvae$Rotten_cold[i] + 1
        }
        else
        {
          if(choice[j] <= 0.5)
          {
            dt_sim_larvae$Rotten_hot[i] <- dt_sim_larvae$Rotten_hot[i] + 1
          }
          else
          {
            if(choice[j] <= 0.75)
            {
              dt_sim_larvae$Fresh_cold[i] <- dt_sim_larvae$Fresh_cold[i] + 1
            }
            else
            {
              dt_sim_larvae$Fresh_hot[i] <- dt_sim_larvae$Fresh_hot[i] + 1
            }
          }
        }
      }
    }
  }
  #Calculating the simulated PIs
  dt_sim_larvae$PI_temp <- (dt_sim_larvae$Rotten_hot +
↪  dt_sim_larvae$Fresh_hot)/(dt_sim_larvae$Rotten_hot + dt_sim_larvae$Fresh_hot +
↪  dt_sim_larvae$Rotten_cold + dt_sim_larvae$Fresh_cold)
  dt_sim_larvae$PI_meat <-  (dt_sim_larvae$Fresh_hot +
↪  dt_sim_larvae$Fresh_cold)/(dt_sim_larvae$Rotten_hot + dt_sim_larvae$Fresh_hot +
↪  dt_sim_larvae$Rotten_cold + dt_sim_larvae$Fresh_cold)

  #Merging the simulated and real dataframe
  dt_test <- rbind(dt_sim_larvae, dt_larvae)

  #Generalized linear model comparing the simulated and real data
  m_meat <- glm(PI_meat ~ stage, family = "quasibinomial",   data = dt_test)
  summ_meat <- summary(m_meat)
  m_temp <- glm(PI_temp ~ stage, family = "quasibinomial",   data = dt_test)
  summ_temp <- summary(m_temp)

  #Saving the statistical metrics of the test in the 'results' data frame
  results_meat$mean[z] <- mean(dt_sim_larvae$PI_meat)
  results_meat$est_sim[z] <- summ_meat$coefficients[[1,1]]
  results_meat$std_sim[z] <- summ_meat$coefficients[[1,2]]
  results_meat$t_sim[z] <- summ_meat$coefficients[[1,3]]
  results_meat$p_sim[z] <- summ_meat$coefficients[[1,4]]
  results_meat$est_real[z] <- summ_meat$coefficients[[2,1]]
  results_meat$std_real[z] <- summ_meat$coefficients[[2,2]]
  results_meat$t_real[z] <- summ_meat$coefficients[[2,3]]
  results_meat$p_real[z] <- summ_meat$coefficients[[2,4]]

  results_temp$mean[z] <- mean(dt_sim_larvae$PI_temp)
  results_temp$est_sim[z] <- summ_temp$coefficients[[1,1]]
  results_temp$std_sim[z] <- summ_temp$coefficients[[1,2]]
  results_temp$t_sim[z] <- summ_temp$coefficients[[1,3]]
  results_temp$p_sim[z] <- summ_temp$coefficients[[1,4]]
  results_temp$est_real[z] <- summ_temp$coefficients[[2,1]]
  results_temp$std_real[z] <- summ_temp$coefficients[[2,2]]
  results_temp$t_real[z] <- summ_temp$coefficients[[2,3]]
  results_temp$p_real[z] <- summ_temp$coefficients[[2,4]]
}
```

Results

Verifying how many of the simulated PIs are random

Ideally, almost none of the simulations should be statistically different from a random choice.

```
ts_meat <- nrow(results_meat[which(results_meat$p_sim < 0.05),])
cat(c("simulated data compared to null hypothesis: PI MEAT\n",ts_meat))
```

```
## simulated data compared to null hypothesis: PI MEAT
##  31
```

```
ts_temp <- nrow(results_temp[which(results_temp$p_sim < 0.05),])
cat(c("simulated data compared to null hypothesis: PI TEMP\n",ts_temp))
```

```
## simulated data compared to null hypothesis: PI TEMP
##  28
```

We found that 31 (3.1%) and 28 (2.8%) of the simulated larval datasets are different from a random meat and temperature PI, respectively. These percentages are quite low (<5%). This result is a good control that indicates that almost all our simulated datasets represent a random choice.

Verifying how many comparisons between real and simulated data are not similar

```
ds_meat <- nrow(results_meat[which(results_meat$p_real < 0.05),])
cat(c("Observed data compared to simulated data: PI MEAT\n",ds_meat))
```

```
## Observed data compared to simulated data: PI MEAT
##  1000
```

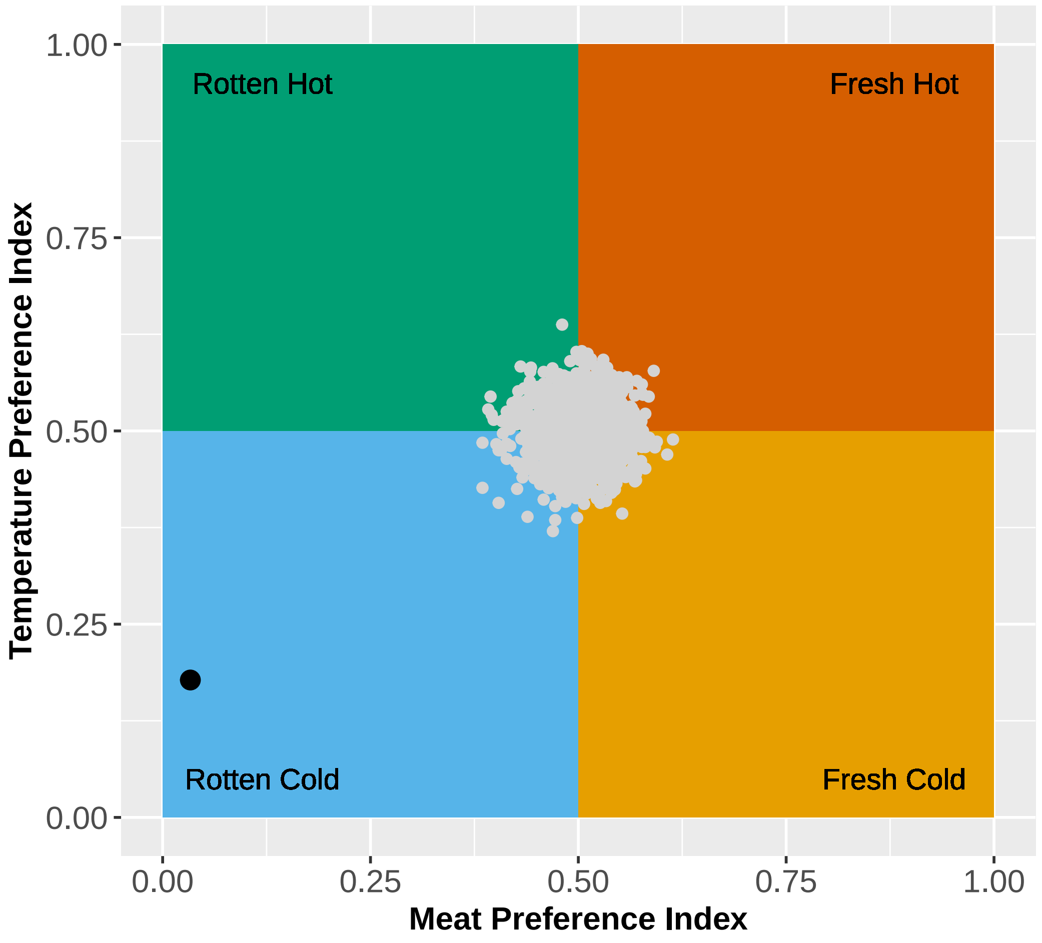
```
ds_temp <- nrow(results_temp[which(results_temp$p_real < 0.05),])
cat(c("Observed data compared to simulated data: PI TEMP\n",ds_temp))
```

```
## Observed data compared to simulated data: PI TEMP
## 1000
```

All (1000) comparisons between our observed and simulated data are significant at a level of 5%: the larval preference is different from a random choice in *Lucilia cuprina*. Larvae display a strong preference for the rotten and cold subtrate (see plot below).

Plotting the graphs for larval data

```
x <- data.frame(1:1001)
x$X1.1001 <- NULL
x$PI_meat <- 0
x$PI_temp <- 0
x$stage <- 0

x$PI_meat[1001]<-mean(dt_larvae$PI_meat)
x$PI_temp[1001]<-mean(dt_larvae$PI_temp)
x$stage[1001] <- "experimental"

x$PI_meat[1:1000] <- results_meat$mean
x$PI_temp[1:1000] <- results_temp$mean
x$stage[1:1000] <- "random"

cbcol <- c("#E69F00", "#D55E00", "#56B4E9", "#009E73")
alpha = 0.1
color <- NA
color[1:1000] <- "light grey"
color[1001] <- "black"
sizes <- NA
sizes[1:1000] <- 1.5
sizes[1001] <- 3

PI<- ggplot(x, aes(PI_meat, PI_temp)) +
  geom_rect(aes(xmin=0,xmax=0.5, ymin=0,ymax=0.5), fill="#56B4E9", alpha=alpha)+
  geom_rect(aes(xmin=0,xmax=0.5, ymin=0.5,ymax=1), fill="#009E73", alpha=alpha)+
  geom_rect(aes(xmin=0.5,xmax=1, ymin=0,ymax=0.5), fill="#E69F00", alpha=alpha)+
  geom_rect(aes(xmin=0.5,xmax=1, ymin=0.5,ymax=1), fill="#D55E00", alpha=alpha)+
  labs(x = 'Meat Preference Index', y = 'Temperature Preference Index') +
  theme(axis.title.x = element_text(size=12, face="bold", colour = "black"), axis.title.y
   ↪  = element_text(size=12, face="bold", colour = "black"))+
  theme(axis.text.x = element_text(size=12), axis.text.y = element_text(size=12),)+
  geom_text(aes(x=0.12,y=0.05,label="Rotten Cold"))+
  geom_text(aes(x=0.12,y=0.95,label="Rotten Hot"))+
  geom_text(aes(x=0.88,y=0.05,label="Fresh Cold"))+
  geom_text(aes(x=0.88,y=0.95,label="Fresh Hot"))+
  geom_point(colour=color, size=sizes)

PI
```



Assesing the preference for female oviposition site

Creating the subset for female and calculating the PIs

```
#setting the female database
dt_fem <- subset(dt_total, dt_total$stage != "L3")

#selecting only the necessary columns for the analysis:
# (stage, Rotten_cold, Rotten_hot, Fresh_cold, Fresh_hot, Did_not_choose, Total)
dt_fem <- dt_fem[,c(2,5:10)]

#Calculating and creating the column for the preference index for temperature
dt_fem$PI_temp <- (dt_fem$Rotten_hot + dt_fem$Fresh_hot)/(dt_fem$Rotten_hot +
↪  dt_fem$Fresh_hot + dt_fem$Rotten_cold + dt_fem$Fresh_cold)

#Calculating and creating the column for the preference index for meat type
dt_fem$PI_meat <-  (dt_fem$Fresh_hot + dt_fem$Fresh_cold)/(dt_fem$Rotten_hot +
↪  dt_fem$Fresh_hot + dt_fem$Rotten_cold + dt_fem$Fresh_cold)

#This colunm is suppose to be ZERO, because we removed data prior to build the database
#We left this part in case someone wants to calculated the probability of female not
↪  choosing
prob_not_choose_fem <- dt_fem$Did_not_choose/dt_fem$Total
```

Getting some basic statistics on our observed data

```
#Testing normality for the number of eggs to support the sample function
shapiro.test(dt_fem$Total)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  dt_fem$Total
## W = 0.94487, p-value = 0.1231
```

The distribution of the number of eggs can be considered normal.

```
#mean and standard deviation of the egg distribution
(mean_eggs <- mean(dt_fem$Total))
```

```
## [1] 132.4333
```

```
(sd_eggs <- sd(dt_fem$Total))
```

```
## [1] 46.17484
```

Establishing the the eggs distribution from the real data

In this section, we determine the probability that a female will choose one, two, three or four substrates to lay her eggs from the real data.

```
#Creating a column that account the amount of sites chosen (always run every line in the
↪  order below)
dt_fem$Sites_chosen <- 0
dt_fem$Sites_chosen <- ifelse(dt_fem$Rotten_cold > 0, dt_fem$Sites_chosen+1,
↪  dt_fem$Sites_chosen)
dt_fem$Sites_chosen <- ifelse(dt_fem$Rotten_hot > 0, dt_fem$Sites_chosen+1,
↪  dt_fem$Sites_chosen)
dt_fem$Sites_chosen <- ifelse(dt_fem$Fresh_cold > 0, dt_fem$Sites_chosen+1,
↪  dt_fem$Sites_chosen)
dt_fem$Sites_chosen <- ifelse(dt_fem$Fresh_hot > 0, dt_fem$Sites_chosen+1,
↪  dt_fem$Sites_chosen)
```

Probability of choosing two sites

```
prob_eggs_two <- length(dt_fem$Sites_chosen[which(dt_fem$Sites_chosen ==
↪  2)])/length(dt_fem$Sites_chosen)
if(prob_eggs_two > 0)
{
  #Given it has chosen two, probability for each choice (greatest to least)
  #creating a database with only two choices
  choice_two <- subset(dt_fem,dt_fem$Sites_chosen ==2)
  #changing into matrix (some functions only work with matrixes)
  choice_two <- as.matrix(choice_two[,c(2:5,7)])
  colnames(choice_two) <- c("greatest", "to", "the", "least", "total")
  #transforming into ratio
  for(i in 1:nrow(choice_two))
  {
    for(j in 1:(ncol(choice_two)-1))
    {
      choice_two[i,j] <- choice_two[i,j]/choice_two[i,5]
    }

  }
  #deleting the column with total values (not using anymore)
  choice_two <- choice_two[,c(1:4)]
  #sorting the number of eggs (greatest to least; the sorting will help in the
  ↪  simulation; the column names have no meaning from now on)
  for(i in 1:nrow(choice_two))
  {
    choice_two[i,] <- sort(choice_two[i,], decreasing = T)
  }
  #means and standard deviation of the greatest and the least values
  m_grt_2 <- mean(choice_two[,1])
  m_to_2 <- mean(choice_two[,2])
  sd_grt_2 <- sd(choice_two[,1])
  sd_to_2 <- sd(choice_two[,2])
}
```

Probability of choosing three sites

```
prob_eggs_three <- length(dt_fem$Sites_chosen[which(dt_fem$Sites_chosen ==
↪  3)])/length(dt_fem$Sites_chosen)
if(prob_eggs_three > 0)
{
  #Given it has chosen three, probability for each choice (greatest to least)
  #creating a database with only three choices
  choice_three <- subset(dt_fem,dt_fem$Sites_chosen == 3)
  #changing into matrix (some functions only work with matrixes)
  choice_three <- as.matrix(choice_three[,c(2:5,7)])
  colnames(choice_three) <- c("greatest", "to", "the", "least", "total")
  #transforming into ratio
  for(i in 1:nrow(choice_three))
  {
    for(j in 1:(ncol(choice_three)-1))
    {
      choice_three[i,j] <- choice_three[i,j]/choice_three[i,5]
    }

  }
  #deleting the column with total values (not using anymore)
  choice_three <- choice_three[,c(1:4)]
  #sorting the number of eggs (greatest to least; the sorting will help in the
  ↪  simulation; the column names have no meaning from now on)
  for(i in 1:nrow(choice_three))
  {
    choice_three[i,] <- sort(choice_three[i,], decreasing = T)
  }
  #means and standard deviation of the greatest and the least values
  m_grt_3 <- mean(choice_three[,1])
  m_to_3 <- mean(choice_three[,2])
  m_the_3 <- mean(choice_three[,3])
  sd_grt_3 <- sd(choice_three[,1])
  sd_the_3 <- sd(choice_three[,2])
  sd_to_3 <- sd(choice_three[,3])
}
```

Probability of choosing four sites

```
prob_eggs_four <- length(dt_fem$Sites_chosen[which(dt_fem$Sites_chosen ==
↪  4)])/length(dt_fem$Sites_chosen)
if(prob_eggs_four > 0)
{
  #Given it has chosen four, probability for each choice (greatest to least)
  #creating a database with only four choices
  choice_four <- subset(dt_fem,dt_fem$Sites_chosen == 4)
  #changing into matrix (some functions only work with matrixes)
  choice_four <- as.matrix(choice_four[,c(2:5,7)])
  colnames(choice_four) <- c("greatest", "to", "the", "least", "total")
  #transforming into ratio
  for(i in 1:nrow(choice_four))
  {
    for(j in 1:(ncol(choice_four)-1))
    {
      choice_four[i,j] <- choice_four[i,j]/choice_four[i,5]
    }

  }
  #deleting the column with total values (not using anymore)
  choice_four <- choice_four[,c(1:4)]
  #sorting the number of eggs (greatest to least; the sorting will help in the
  ↪  simulation; the column names have no meaning from now on)
  for(i in 1:nrow(choice_four))
  {
    choice_four[i,] <- sort(choice_four[i,], decreasing = T)
  }
  #means and standard deviation of the greatest and the least values
  m_grt_4 <- mean(choice_four[,1])
  m_to_4 <- mean(choice_four[,2])
  m_the_4 <- mean(choice_four[,3])
  m_lst_4 <- mean(choice_four[,4])
  sd_grt_4 <- sd(choice_four[,1])
  sd_the_4 <- sd(choice_four[,2])
  sd_to_4 <- sd(choice_four[,3])
  sd_lst_4 <- sd(choice_four[,4])
}
```

Probability of choosing one site

```
prob_eggs_one <- 1 - prob_eggs_two - prob_eggs_three - prob_eggs_four
```

Simulating female data

Creating tables to hold the results of the simulations

```
#seed used to generate the results of the article/video. This can be changed or not used
↪  at all.
set.seed(1)

#Creating the simulated dataframe for female
dt_sim_fem <- data.frame(1:nrow(dt_fem))
dt_sim_fem$X1.nrow.dt_fem. <- NULL #Deleting the column automatically created

#Classifying the stage of the simulated data for later comparison (we put the '!' just to
↪  order it first alphabetically)
dt_sim_fem$stage <- "!random"

#Setting the values as zero for now
dt_sim_fem$Rotten_cold <- 0
dt_sim_fem$Rotten_hot <- 0
dt_sim_fem$Fresh_cold <- 0
dt_sim_fem$Fresh_hot <- 0
dt_sim_fem$Did_not_choose <- 0
dt_sim_fem$Total <- 0
dt_sim_fem$Sites_chosen <- 0

#Creating the dataframes for the statistical metrics
results_meat <- data.frame(1:1000)
results_meat$X1.1000 <- NULL
results_meat$mean <- 0
results_meat$est_sim <- 0
results_meat$std_sim <- 0
results_meat$t_sim <- 0
results_meat$p_sim <- 0
results_meat$est_real <- 0
results_meat$std_real <- 0
results_meat$t_real <- 0
results_meat$p_real <- 0

results_temp <- data.frame(1:1000)
results_temp$X1.1000 <- NULL
results_temp$mean <- 0
results_temp$est_sim <- 0
results_temp$std_sim <- 0
results_temp$t_sim <- 0
results_temp$p_sim <- 0
results_temp$est_real <- 0
results_temp$std_real <- 0
results_temp$t_real <- 0
results_temp$p_real <- 0
```

Running the simulation 1000 times

```r
for(z in 1:1000)
{
  #Reseting the values for the simulated data frame every time
  dt_sim_fem$Rotten_cold <- 0
  dt_sim_fem$Rotten_hot <- 0
  dt_sim_fem$Fresh_cold <- 0
  dt_sim_fem$Fresh_hot <- 0
  dt_sim_fem$Did_not_choose <- 0
  #running the simulation for one simulated data frame (= the number of rows of the data
  ↪  frame)
  for(i in 1:nrow(dt_sim_fem))
  {
    #sorting the amount of eggs from the mean/sd of the actual data for one simulated
    ↪  female
    eggs_n <- as.integer(pmax(rnorm(1,mean_eggs,sd_eggs),1))
    #establishing if this simulated female is choosing one or more oviposition sites
    dice <- runif(1,0,1)
    choice_n <- ifelse(dice > (1-prob_eggs_one), 1,
                       ifelse(dice > (1-prob_eggs_one-prob_eggs_two), 2,
                              ifelse(dice >
                              ↪  (1-prob_eggs_one-prob_eggs_two-prob_eggs_three),3,4)))
    #if the choice is for one site, place all simulated eggs in one site
    if(choice_n == 1)
    {
      choice_1 <- sample.int(4,1, replace = T)
      if(choice_1 == 1)
      {
        dt_sim_fem$Rotten_cold[i] <- eggs_n
      }
      else
      {
        if(choice_1 == 2)
        {
          dt_sim_fem$Rotten_hot[i] <- eggs_n
        }
        else
        {
          if(choice_1 == 3)
          {
            dt_sim_fem$Fresh_cold[i] <- eggs_n
          }
          else
          {
            dt_sim_fem$Fresh_hot[i] <- eggs_n
          }
        }
      }
    }
    #if the choice is for two sites, lay a proportion (base on actual data) on each sites
    if(choice_n == 2)
    {
      eggs_2 <- c(0,0)
      eggs_2[1] <- as.integer(pmax(rnorm(1,m_grt_2,sd_grt_2) * eggs_n, 1))
      eggs_2[2] <- pmax(eggs_n - eggs_2[1], 1)
      ifelse(eggs_2 < 0 , print("here2"), NA )

      choice_2 <- sample.int(4,2, replace = F)
      for(k in 1:choice_n)
      {
        if(choice_2[k] == 1)
        {
          dt_sim_fem$Rotten_cold[i] <- eggs_2[k]
        }
        else
        {
          if(choice_2[k] == 2)
          {
            dt_sim_fem$Rotten_hot[i] <- eggs_2[k]
          }
          else
          {
            if(choice_2[k] == 3)
            {
              dt_sim_fem$Fresh_cold[i] <- eggs_2[k]
            }
            else
            {
              dt_sim_fem$Fresh_hot[i] <- eggs_2[k]
            }
          }
        }
      }
    }
    #if the choice is for three sites, lay a proportion (base on actual data) on every
     ↪  site
    if(choice_n == 3)
    {
      eggs_3 <- c(0,0,0)
      eggs_3[1] <- as.integer(pmax(rnorm(1,m_grt_3,sd_grt_3) * eggs_n, 1))
      eggs_3[2] <- as.integer(pmax(rnorm(1,m_to_3,sd_to_3) * eggs_n, 1))
      eggs_3[3] <- pmax(eggs_n - eggs_3[1]-eggs_3[2], 1)
      choice_3 <- sample.int(4,3, replace = F)
      for(k in 1:choice_n)
      {
        if(choice_3[k] == 1)
        {
          dt_sim_fem$Rotten_cold[i] <- eggs_3[k]
        }
        else
        {
          if(choice_3[k] == 2)
          {
            dt_sim_fem$Rotten_hot[i] <- eggs_3[k]
          }
          else
          {
            if(choice_3[k] == 3)
            {
              dt_sim_fem$Fresh_cold[i] <- eggs_3[k]
            }
            else
            {
              dt_sim_fem$Fresh_hot[i] <- eggs_3[k]
            }
          }
        }
      }
    }
    #if the choice is for four sites, lay a proportion (base on actual data) on every
     ↪  site
    if(choice_n == 4)
    {
      eggs_4 <- c(0,0,0,0)
      eggs_4[1] <- as.integer(pmax(rnorm(1,m_grt_4,sd_grt_4) * eggs_n, 1))
      eggs_4[2] <- as.integer(pmax(rnorm(1,m_to_4,sd_to_4) * eggs_n, 1))
      eggs_4[3] <- as.integer(pmax(rnorm(1,m_the_4,sd_the_4) * eggs_n, 1))
      eggs_4[4] <- pmax((eggs_n - eggs_4[1]-eggs_4[2]-eggs_4[3]), 1)
      choice_4 <- sample.int(4,4, replace = F)
      for(k in 1:choice_n)
      {
        if(choice_4[k] == 1)
        {
          dt_sim_fem$Rotten_cold[i] <- eggs_4[k]
        }
        else
        {
          if(choice_4[k] == 2)
          {
            dt_sim_fem$Rotten_hot[i] <- eggs_4[k]
          }
          else
          {
            if(choice_4[k] == 3)
            {
              dt_sim_fem$Fresh_cold[i] <- eggs_4[k]
            }
            else
            {
              dt_sim_fem$Fresh_hot[i] <- eggs_4[k]
            }
          }
        }
      }
    }
  }


  #Calculating the simulated PIs
  dt_sim_fem$PI_temp <- (dt_sim_fem$Rotten_hot +
    ↪  dt_sim_fem$Fresh_hot)/(dt_sim_fem$Rotten_hot + dt_sim_fem$Fresh_hot +
    ↪  dt_sim_fem$Rotten_cold + dt_sim_fem$Fresh_cold)
```

```
  dt_sim_fem$PI_meat <-  (dt_sim_fem$Fresh_hot +
↪  dt_sim_fem$Fresh_cold)/(dt_sim_fem$Rotten_hot + dt_sim_fem$Fresh_hot +
↪  dt_sim_fem$Rotten_cold + dt_sim_fem$Fresh_cold)

  #Merging the simulated and real dataframe
  dt_test <- rbind(dt_sim_fem, dt_fem)

  #Generalized linear model comparing the simulated and real data
  m_meat <- glm(PI_meat ~ stage, family = "quasibinomial",   data = dt_test)
  summ_meat <- summary(m_meat)
  m_temp <- glm(PI_temp ~ stage, family = "quasibinomial",   data = dt_test)
  summ_temp <- summary(m_temp)

  #Saving the statistical metrics of the test in the 'results' data frame
  results_meat$mean[z] <- mean(dt_sim_fem$PI_meat)
  results_meat$est_sim[z] <- summ_meat$coefficients[[1,1]]
  results_meat$std_sim[z] <- summ_meat$coefficients[[1,2]]
  results_meat$t_sim[z] <- summ_meat$coefficients[[1,3]]
  results_meat$p_sim[z] <- summ_meat$coefficients[[1,4]]
  results_meat$est_real[z] <- summ_meat$coefficients[[2,1]]
  results_meat$std_real[z] <- summ_meat$coefficients[[2,2]]
  results_meat$t_real[z] <- summ_meat$coefficients[[2,3]]
  results_meat$p_real[z] <- summ_meat$coefficients[[2,4]]

  results_temp$mean[z] <- mean(dt_sim_fem$PI_temp)
  results_temp$est_sim[z] <- summ_temp$coefficients[[1,1]]
  results_temp$std_sim[z] <- summ_temp$coefficients[[1,2]]
  results_temp$t_sim[z] <- summ_temp$coefficients[[1,3]]
  results_temp$p_sim[z] <- summ_temp$coefficients[[1,4]]
  results_temp$est_real[z] <- summ_temp$coefficients[[2,1]]
  results_temp$std_real[z] <- summ_temp$coefficients[[2,2]]
  results_temp$t_real[z] <- summ_temp$coefficients[[2,3]]
  results_temp$p_real[z] <- summ_temp$coefficients[[2,4]]
}
```

Results

Verifying how many of the simulated PIs are not random

Ideally, almost none of the simulations should be statistically different from a random choice.

```
ts_meat <- nrow(results_meat[which(results_meat$p_sim < 0.05),])
cat(c("simulated data compared to null hypothesis: PI MEAT\n",ts_meat))
```

```
## simulated data compared to null hypothesis: PI MEAT
##  33
```

```
ts_temp <- nrow(results_temp[which(results_temp$p_sim < 0.05),])
cat(c("simulated data compared to null hypothesis: PI TEMP\n",ts_temp))
```

```
## simulated data compared to null hypothesis: PI TEMP
##  47
```

We found that 33 (3.3%) and 47 (4.7%) of the simulated larval datasets are different from a random meat and temperature PI, respectively. These percentages are quite low (<5%). This result is a good control that indicates that almost all our simulated datasets represent a random choice.

Verifying how many of the simulated PIs are statistically different from the observed data

```
ds_meat <- nrow(results_meat[which(results_meat$p_real < 0.05),])
cat(c("Observed data compared to simulated data: PI MEAT\n",ds_meat))
```

```
## Observed data compared to simulated data: PI MEAT
##  271
```

```
ds_temp <- nrow(results_temp[which(results_temp$p_real < 0.05),])
cat(c("Observed data compared to simulated data: PI TEMP\n",ds_temp))
```

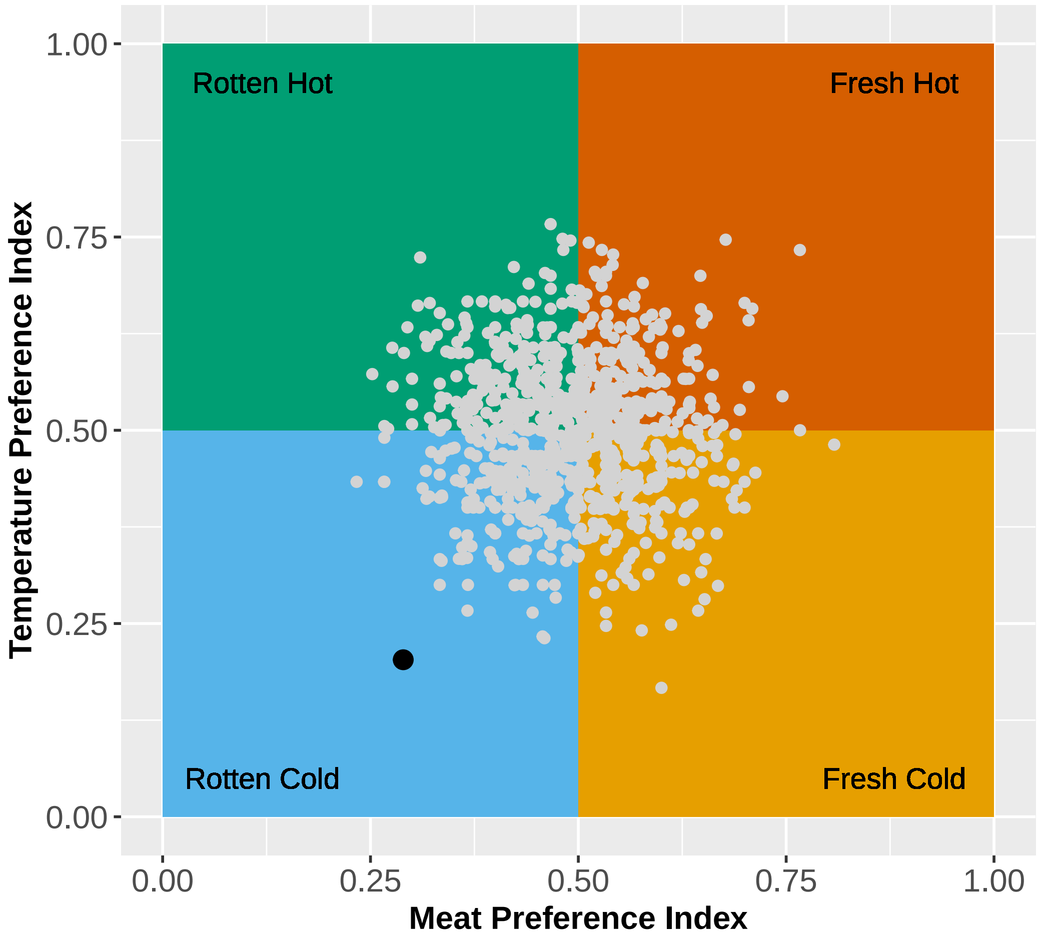
```
## Observed data compared to simulated data: PI TEMP
##  697
```

For the meat type, 271 (27.1%) of the simulated datasets were significantly different from the observed data at a level of 5%. This is indicative of a slight preference towards a meat substrate (rotten, see plot below) for egg-laying.

For the temperature, 697 (69.7%) of the simulated datasets were significantly different from the observed data at a level of 5%. This is indicative of a temperature preference (cold, see plot below) for egg-laying.

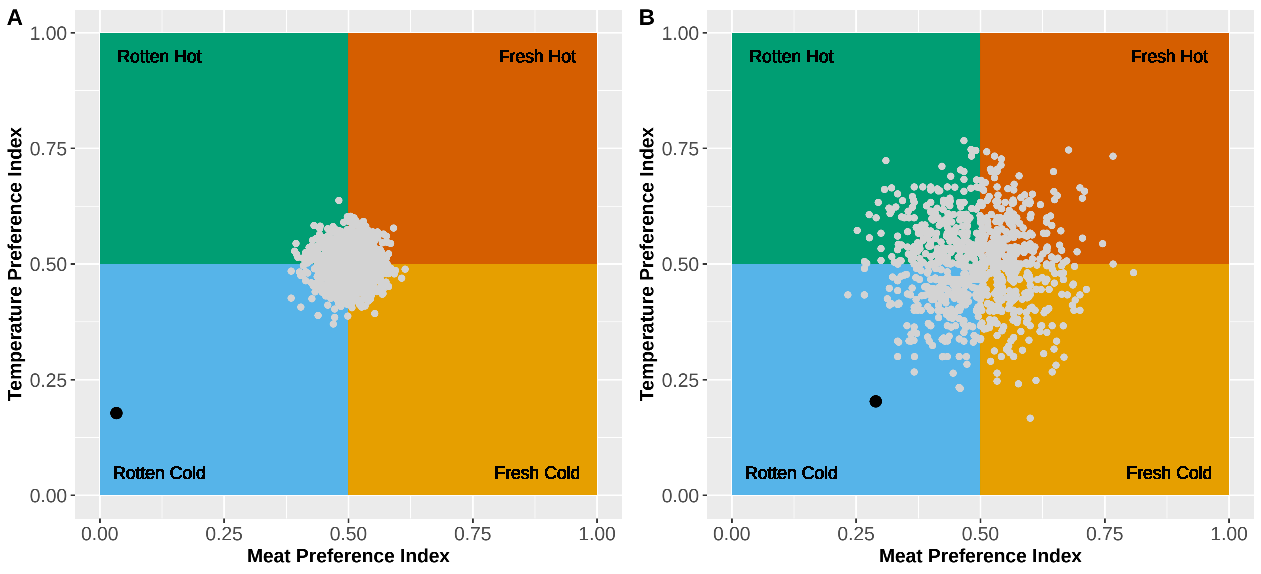
Plotting the graph for female data

```
#### Female graphs ####

x <- data.frame(1:1001)
x$X1.1001 <- NULL
x$PI_meat <- 0
x$PI_temp <- 0
x$stage <- 0

x$PI_meat[1001]<-mean(dt_fem$PI_meat)
x$PI_temp[1001]<-mean(dt_fem$PI_temp)
x$stage[1001] <- "experimental"

x$PI_meat[1:1000] <- results_meat$mean
x$PI_temp[1:1000] <- results_temp$mean
x$stage[1:1000] <- "random"


cbcol <- c("#E69F00", "#D55E00", "#56B4E9", "#009E73")
alpha = 0.1 # 0.08 for larva, 0.1 for fem
color <- NA
color[1:1000] <- "light grey"
color[1001] <- "black"
sizes <- NA
sizes[1:1000] <- 1.5
sizes[1001] <- 3
PII<- ggplot(x, aes(PI_meat, PI_temp)) +
  geom_rect(aes(xmin=0,xmax=0.5, ymin=0,ymax=0.5), fill="#56B4E9", alpha=alpha)+
  geom_rect(aes(xmin=0,xmax=0.5, ymin=0.5,ymax=1), fill="#009E73", alpha=alpha)+
  geom_rect(aes(xmin=0.5,xmax=1, ymin=0,ymax=0.5), fill="#E69F00", alpha=alpha)+
  geom_rect(aes(xmin=0.5,xmax=1, ymin=0.5,ymax=1), fill="#D55E00", alpha=alpha)+
  labs(x = 'Meat Preference Index', y = 'Temperature Preference Index') +
  theme(axis.title.x = element_text(size=12, face="bold", colour = "black"), axis.title.y
  ↪  = element_text(size=12, face="bold", colour = "black"))+
  theme(axis.text.x = element_text(size=12), axis.text.y = element_text(size=12),)+
  geom_text(aes(x=0.12,y=0.05,label="Rotten Cold"))+
  geom_text(aes(x=0.12,y=0.95,label="Rotten Hot"))+
  geom_text(aes(x=0.88,y=0.05,label="Fresh Cold"))+
  geom_text(aes(x=0.88,y=0.95,label="Fresh Hot"))+
  geom_point(colour=color, size=sizes)

PII
```



Plot the two graphs at once

```
PIall <- ggarrange(PI, PII, ncol=2, labels=c("A", "B"))
PIall
```



```
ggsave("graphs.pdf", PIall, width = 10.25, height = 5)
ggsave("Larvae.png", PI, width = 5.25, height = 5)
ggsave("Females.png", PII, width = 5.25, height = 5)
```

R.Version: 4.2.0 RStudio.version: 2022.2.3.492