

# Sea\_Cucumber\_Wasting\_JF\_DT\_Dec2021

Jonathan Farr and Declan Taylor

12/12/2021

## Loading in our datafiles

Reading in packages used for data cleaning/sorting:

```
library(here)
```

```
## here() starts at C:/Users/Jonathan Farr/Github/DS_cucumber
```

```
library(lubridate)
```

```
## Warning: package 'lubridate' was built under R version 4.1.2
```

```
##
```

```
## Attaching package: 'lubridate'
```

```
## The following objects are masked from 'package:base':
```

```
##
```

```
##      date, intersect, setdiff, union
```

```
library(tidyverse)
```

```
## -- Attaching packages ----- tidyverse 1.3.1 --
```

```
## v ggplot2 3.3.5      v purrr   0.3.4
```

```
## v tibble  3.1.6      v dplyr  1.0.7
```

```
## v tidyr   1.1.4      v stringr 1.4.0
```

```
## v readr   2.1.1      v forcats 0.5.1
```

```
## Warning: package 'tibble' was built under R version 4.1.2
```

```
## Warning: package 'tidyr' was built under R version 4.1.2
```

```
## Warning: package 'readr' was built under R version 4.1.2
```

```
## -- Conflicts ----- tidyverse_conflicts() --
```

```
## x lubridate::as.difftime() masks base::as.difftime()
```

```
## x lubridate::date()       masks base::date()
```

```
## x dplyr::filter()         masks stats::filter()
```

```
## x lubridate::intersect()  masks base::intersect()
```

```
## x dplyr::lag()            masks stats::lag()
```

```
## x lubridate::setdiff()    masks base::setdiff()
```

```
## x lubridate::union()      masks base::union()
```

## Longitudinal temperature data for figure 4

```
# Importing data and formatting the date/time information.
DailyLog <- read_csv("data/DailyLog.csv", col_names = TRUE) %>%
  # Format `Date` column to POSIX standard
  mutate("Date" = dmy(Date)) %>%
  # Create column with date and time info
  mutate("dateTime" = paste(Date, Time, sep = "_")) %>%
  # Format `dateTime` to POSIX standard
  mutate(dateTime = ymd_hms(dateTime)) %>%
  # Make `Sea_Table` and `Bucket_ID` factorial data.
  mutate(Sea_Table = as.factor(Sea_Table),
         Bucket_ID = as.factor(Bucket_ID)) %>%
  # Renaming!
  select(date = Date,
         date_time = dateTime,
         sea_table = Sea_Table,
         table_position,
         bucketID = Bucket_ID,
         cukeID = Cuke_ID,
         treatment = Treatment,
         temp_C = Temp_C,
         alive = Alive,
         death_time = `Time of Death`,
         poop = Poop,
         evisceration = Evisceration,
         resp_evisc = respiratory_evisceration,
         spawn = Spawn) %>%
  mutate(tableID = paste(sea_table, table_position)) %>%
  tail(-34)

# Making a new dataframe to get some values about the range and variation in our
# temperature data.
tempRange <- DailyLog %>%
  filter(date == "2021-11-11" | date == "2021-11-12") %>%
  select(date, date_time, tableID, treatment, temp_C)

# Generating a list of the times at which the 5 cucumbers died.
death_time <- DailyLog %>%
  # Filter for rows with death data
  filter(FALSE == is.na(alive) | FALSE == is.na(death_time)) %>%
  # Use POSIXct standard for death_time
  mutate(death_time = ymd_hms(paste(date, death_time))) %>%
  select(death_time, bucketID, temp_C) %>%
  mutate(placeholder = 1)

# Generating a dataframe for the temperature data over time so that we can
# create a plot.
Temp_Time <- DailyLog %>%
  select(date_time, date, sea_table, table_position, bucketID, tableID, temp_C)
```

## Binary data for analyses and figures related to mortality, spawning, ulcers and evisceration

Functions that generate data frames with death\_time, spawning, evisceration, and poop data. The function exists so the dataframe can be easily made in one click.

```
create_individualData <- function(datafile){  
  # Import Data  
  DailyLog <- read_csv(here(paste0("data/", datafile)), col_names = TRUE) %>%  
    # Format `Date` column to POSIX standard  
    mutate("Date" = dmy(Date)) %>%  
    mutate("dateTime" = paste(Date, Time, sep = "_")) %>%  
    mutate(dateTime = ymd_hms(dateTime))  
  
  # Generate data frame to hold just the binary variables. Upper limit of  
  # dataframe is intentionally too big (I'm just removing the top 35 rows).  
  SelectedData <- DailyLog %>%  
    # remove the first 34 lines of data frame (pre-experiment data).  
    tail(-34) %>%  
    # Select and rename variables  
    select(date = Date,  
           date_time = dateTime,  
           sea_table = Sea_Table,  
           table_position,  
           bucketID = Bucket_ID,  
           cukeID = Cuke_ID,  
           treatment = Treatment,  
           temp_C = Temp_C,  
           alive = Alive,  
           death_time = `Time of Death`,  
           poop = Poop,  
           evisceration = Evisceration,  
           resp_evisc = respiratory_evisceration,  
           spawn = Spawn) %>%  
    # Generate a `combinedID` from bucketID (1-30) and cukeID (A or B), which is  
    # unique to each individual cucumber in the study.  
    mutate(combinedID = paste(bucketID, cukeID),  
           tableID = paste(sea_table, table_position))  
  
  # For each individual-level variable, create a dataframe that selects only  
  # the rows for which that variable's column has data. Each dataframe is named  
  # after the variable it represents.  
  # Format that data to change all the values to "1". Select only the  
  # combinedID and <variable> columns, and keep only the distinct combinedID  
  # values. This effectively generates a list of cucumbers in which the  
  # <variable> process (i.e. pooping) occurred.  
  death_time <- SelectedData %>%  
    # Filter for rows with death data  
    filter(FALSE == is.na(alive) | FALSE == is.na(death_time)) %>%  
    # Use POSIXct standard for death_time  
    mutate(death_time = ymd_hms(paste(date, death_time))) %>%  
    select(death_time, combinedID)  
  
  evisceration <- SelectedData %>%
```

```

filter(FALSE == is.na(evisceration)) %>%
# sub text entries like "yes" for `1`.
mutate(evisceration = gsub("[A-z]{3}", 1, evisceration))%>%
select(combinedID, evisceration) %>%
distinct(combinedID, .keep_all = TRUE)

resp_evisc <- SelectedData %>%
  filter(FALSE == is.na(resp_evisc)) %>%
  mutate(resp_evisc = gsub("[A-z]{3}", 1, resp_evisc))%>%
  select(combinedID, resp_evisc) %>%
  distinct(combinedID, .keep_all = TRUE)

poop <- SelectedData %>%
  filter(FALSE == is.na(poop)) %>%
  mutate(poop = gsub("[A-z]{3}", 1, poop)) %>%
  select(combinedID, poop) %>%
  distinct(combinedID, .keep_all = TRUE)

spawn <- SelectedData %>%
  # text entries may be "yes" or "eggs".
  mutate(spawn = gsub("[A-z]{3,4}", 1, spawn, ignore.case = TRUE)) %>%
  filter(FALSE == is.na(spawn)) %>%
  select(combinedID, spawn) %>%
  distinct(combinedID, .keep_all = TRUE)

# List the variables for which dataframes were created.
SelectedVariables <- c("death_time", "evisceration", "resp_evisc", "poop", "spawn")

# Generate a 1-column data frame which is a list of all the combinedID values
# (i.e. all the unique bins in our experiment).
IndividualData <- SelectedData %>%
  select(date_time,
         tableID,
         bucketID,
         cukeID,
         combinedID,
         treatment) %>%
  distinct(combinedID, .keep_all = TRUE) %>%
  drop_na()

# For each of the individual data frames made above, join the `1` data
# (selected occurrence data) to the 1-column dataframe for only the values
# with 1s. The rest are left as NAs.
for(i in SelectedVariables) {
  variable <- get(i)
  IndividualData <- full_join(IndividualData, variable, by = "combinedID")

  IndividualData[,i] <- IndividualData[,i] %>%
    replace_na()
}

# Replace NA values with 0s and make each column numeric; turns each binary
# response variable into a binary data column with 0s and 1s.

```

```

IndividualData$evisceration <- IndividualData$evisceration %>%
  replace_na(0) %>%
  as.numeric(IndividualData$evisceration)
IndividualData$respc_evisc <- IndividualData$respc_evisc %>%
  replace_na(0) %>%
  as.numeric(IndividualData$respc_evisc)
IndividualData$poop <- IndividualData$poop %>%
  replace_na(0) %>%
  as.numeric(IndividualData$poop)
IndividualData$spawn <- IndividualData$spawn %>%
  replace_na(0) %>%
  as.numeric(IndividualData$spawn)

# Fix data types to factors and assign IndividualData to the global environment
IndividualData <- IndividualData %>%
  mutate(treatment = as.factor(treatment),
         cukeID = as.factor(cukeID),
         poop = as.factor(poop),
         tableID = as.factor(tableID),
         bucketID = as.factor(bucketID))
}

# This function generates a dataframe stress scores and
# joins said dataframe to the 'master' IndividualData dataframe.
add_stressData <- function(datafile){
  # Import data
  StressData <- read_csv(here(paste0("data/", datafile)), col_names = TRUE) %>%
    # Format `Date` column to POSIX standard
    mutate("Date" = dmy(Date)) %>%
    mutate("dateTime" = paste(Date, Time, sep = "_")) %>%
    mutate(dateTime = ymd_hms(dateTime)) %>%
    select(date = Date,
           time = Time,
           date_time = dateTime,
           sea_table = Sea_Table,
           table_position = Table_Position,
           bucketID = Bucket_ID,
           cukeID = Cuke_ID,
           treatment = Treatment,
           activity = Activity_Score,
           squeeze = Squeeze_score,
           droop = Droop_score) %>%
    mutate(combinedID = paste(bucketID, cukeID))

  # Generate initial data values for droop and squeeze based on the
  # readings taken on the first day of the experiment.
  initial_stress_values <- StressData %>%
    # dplyr method of doing: filter(date_time == "2021-11-09 09:40:00")
    mutate(date = as.character(date),
           time = as.character(time)) %>%
    filter(date == "2021-11-09" &
           time == "09:40:00") %>%
    # Make droop and squeeze data factorial

```

```

mutate(droop = as.factor(droop),
       squeeze = as.factor(squeeze)) %>%
# Columns to be added to IndividualData, with droop and squeeze indicated
# as initial data.
select(combinedID,
       in_activity = activity,
       in_droop = droop,
       in_squeeze = squeeze)

IndividualData <-> full_join(IndividualData, initial_stress_values, by = "combinedID")
}

# This function adds weight data to the master IndividualData dataframe.
add_weightData <- function(datafile){
  WeightData <- read_csv(here(paste0("data/", datafile)), col_names = TRUE) %>%
  mutate(combinedID = paste(Bucket_ID, Cuke_ID)) %>%
  # Create an average weight from the two weight measurements taken at the
  # start of the experiment.
  mutate(weight_g = (Weight_g + Weight_2)/2) %>%
  select(weight_g, combinedID)

  # Join the weight data to the full dataframe.
  IndividualData <-> full_join(IndividualData, WeightData, by = "combinedID")
}

```

Now, we run all three functions once, sequentially. Returned data frame should have 16 variables across. IndividualData df is the desired output

```

create_individualData("DailyLog_final.csv")
add_stressData("BehaviourData_final.csv")
add_weightData("SizeData.csv")

```

## Second, we read in longitudinal data frames with behaviours measured over time

This data will later be used for stiffness analysis

```

behav = read_csv("data/BehaviourData_Final.csv") %>%
  mutate(
    Treatment=fct_relevel(Treatment, c("Control", "Room", "Heat")),
    Bucket_ID = as.factor(Bucket_ID),
    Cuke_ID=as.factor(Cuke_ID),
    Unique_ID=paste(Bucket_ID, Cuke_ID, sep = '_'),
    Table_ID = paste(Sea_Table, Table_Position, sep='_'),
    Date = as.Date(Date, format="%d-%m-%Y")) %>%
  dplyr:: select(-c(Sea_Table, Table_Position))

```

## Data for ulcer analysis

```

lesion = read_csv("data/BehaviourData_Final.csv") %>%
  mutate(

```

```

Treatment=fct_relevel(Treatment, c("Control","Room","Heat")),
Bucket_ID = as.factor(Bucket_ID),
Cuke_ID=as.factor(Cuke_ID),
Unique_ID=paste(Bucket_ID, Cuke_ID, sep = '_'),
Table_ID = paste(Sea_Table, Table_Position, sep='_'),
Date = as.Date(Date, format="%d-%m-%Y")) %>%
dplyr::select(c(Unique_ID, Bucket_ID, Table_ID, Date, Treatment, Number_lesions)) %>%
na.omit()

# find max number of lesions per individual
lesion_max = lesion %>%
group_by(Unique_ID) %>%
mutate(max_lesions = max(Number_lesions)) %>%
distinct(Unique_ID, .keep_all=TRUE) %>%
dplyr::select(-"Number_lesions")

# reading in weight data
size = read_csv("data/SizeData.csv") %>%
mutate(
  Bucket_ID = as.factor(Bucket_ID),
  Cuke_ID=as.factor(Cuke_ID),
  Unique_ID=paste(Bucket_ID, Cuke_ID, sep = '_'),
  mean_weight = (Weight_g+Weight_2)/2) %>%
dplyr::select(c(Unique_ID, mean_weight))

# adding weight data to lesion data (using IndividualData)
individual_pooping = IndividualData %>%
mutate(Unique_ID = paste(bucketID, cukeID, sep="_"))%>%
select(c("Unique_ID", "poop", "evisceration"))

lesion_max = merge(lesion_max, individual_pooping, by=c("Unique_ID"))
lesion_max = merge(lesion_max, size, by=c("Unique_ID"))

```

## Data for mortality analysis = DeathData

```

# Restrict the IndividualData dataframe just to cucumbers that had a comment
# in the death column (only done for cucumbers that died).
DeathData <- IndividualData %>%
dplyr::mutate(death_time = gsub(".{1,}", 1, death_time)) %>%
dplyr::select(-in_activity)

# Replace NA values in the death_time colum with 0s.
DeathData$death_time <- DeathData$death_time %>%
replace_na(0) %>%
as.numeric(IndividualData$death_time)

DeathData <- DeathData %>%
rename("death" = death_time)

```

## Data for stiffness = stiff

```
stiff = behav %>%
  dplyr::select(-c(Activity_Score, Number_lesions, 'Bodywall lesions')) %>%
  na.omit()
```

## Data for spawning = spawnData

```
# This function generates a dataframe that our spawning statistical analysis
# uses. `dplyr` functions that conflict with other packages are captioned
# explicitly .
create_spawnData <- function(datafile){
  SelectedData <- read_csv(here(paste0("data/", datafile)), col_names = TRUE) %>%
    # Format `Date` column to POSIX standard
    dplyr::mutate("Date" = dmy(Date),
                  "dateTime" = paste(Date, Time, sep = "-"),
                  dateTime = ymd_hms(dateTime))

  SelectedData <- SelectedData %>%
    # remove the first 34 lines of data frame (improperly formatted
    # pre-experiment data).
    tail(-34) %>%
    # Select and rename variables
    dplyr::select(date = Date,
                  date_time = dateTime,
                  sea_table = Sea_Table,
                  table_position,
                  bucketID = Bucket_ID,
                  cukeID = Cuke_ID,
                  treatment = Treatment,
                  temp_C = Temp_C,
                  alive = Alive,
                  death_time = `Time of Death`,
                  poop = Poop,
                  evisceration = Evisceration,
                  resp_evisc = respiratory_evisceration,
                  spawn = Spawn) %>%
    # replace any word such as "Yes", "yes", "egg", or "Eggs" in the spawn
    # column with a `1`.
    dplyr::mutate(spawn = gsub("[A-z]{3,4}", 1, spawn, ignore.case = TRUE))

  # Select only buckets and spawn (`1`) data for those buckets that spawned.
  spawnData <- SelectedData %>%
    dplyr::filter(FALSE == is.na(spawn)) %>%
    dplyr::select(bucketID, spawn) %>%
    distinct(bucketID, .keep_all = TRUE)

  # Generate a list (1-col dataframe) with all of the buckets (bins) in the
  # experiment listed by their ID numbers.
  bucketList <- SelectedData %>%
    dplyr::select(bucketID, treatment) %>%
```



```

distinct(bucketID, .keep_all = TRUE) %>%
drop_na()

# Join the spawn information (`1`) counts with the total list of all buckets
# to create a dataframe of 1s and 0s of spawn data.
spawnData <- full_join(bucketList, spawnData, by = "bucketID") %>%
  dplyr::mutate(spawn = as.numeric(spawn))
spawnData$spawn <- replace_na(spawnData$spawn, 0)

# Save this dataframe to the RStudio global environment.
spawnData <- spawnData
}

# This function is exactly the same as the above, but the regular expressions
# are slightly adjusted to only pick up sperm data instead of sperm and egg
# spawning.
create_spermData <- function(datafile){

  SelectedData <- read_csv(here(paste0("data/", datafile)), col_names = TRUE) %>%
    dplyr::mutate("Date" = dmy(Date)) %>%
    dplyr::mutate("dateTime" = paste(Date, Time, sep = "_")) %>%
    dplyr::mutate(dateTime = ymd_hms(dateTime))

  SelectedData <- SelectedData %>%
    tail(-34) %>%
    dplyr::select(date = Date,
                  date_time = dateTime,
                  sea_table = Sea_Table,
                  table_position,
                  bucketID = Bucket_ID,
                  cukeID = Cuke_ID,
                  treatment = Treatment,
                  temp_C = Temp_C,
                  alive = Alive,
                  death_time = `Time of Death`,
                  poop = Poop,
                  evisceration = Evisceration,
                  resp_evisc = respiratory_evisceration,
                  spawn = Spawn) %>%
    # Note the regular expressions, which must start with [Y, y], and must be
    # 3 total characters.
    dplyr::mutate(spawn = gsub("y[A-z]{2}", 1, spawn, ignore.case = TRUE))

  spawnData <- SelectedData %>%
    dplyr::filter(FALSE == is.na(spawn)) %>%
    dplyr::select(bucketID, spawn) %>%
    distinct(bucketID, .keep_all = TRUE)

  bucketList <- SelectedData %>%
    dplyr::select(bucketID, treatment) %>%
    distinct(bucketID, .keep_all = TRUE) %>%
    drop_na()

```

```

spawnData <- full_join(bucketList, spawnData, by = "bucketID") %>%
  dplyr::mutate(spawn = as.numeric(spawn))
spawnData$spawn <- replace_na(spawnData$spawn, 0)

spermData <- spawnData
}

# Running the 2 functions above to generate the dataframes required to run the
# statistics below.
create_spawnData("DailyLog_final.csv")
create_spermData("DailyLog_final.csv")

```

## Data for evisceration = EviscData

```

# Drop the final row (all NAs) and the death_time and in_activity columns
# (contain NAs) from `IndividualData`, so we can use gamlss models.
EviscData <- IndividualData %>%
  dplyr::select(-c(death_time, in_activity, in_droop, in_squeeze))

```

## Running analyses

Now, we use our data frames generated above to conduct analyses on our data First - load in the packages used for analyses

```
library(gamlss)
library(FSA)
library(ordinal)
library(MuMIn)
library(FSA)
library(Hmisc)
```

## Temperature trends

making sure that temps were different in our treatments

```
# Investigating the range and average temperatures of our 3 treatments.
```

```
# 12C
```

```
range(subset(tempRange, treatment=="control")$temp_C)
```

```
## [1] 11.2 14.0
```

```
mean(subset(tempRange, treatment=="control")$temp_C)
```

```
## [1] 12.50439
```

```
# 17C
```

```
range(subset(tempRange, treatment=="room")$temp_C)
```

```
## [1] 14.9 17.9
```

```
mean(subset(tempRange, treatment=="room")$temp_C)
```

```
## [1] 16.78684
```

```
# 22C
```

```
range(subset(tempRange, treatment=="heat")$temp_C)
```

```
## [1] 20.0 23.3
```

```
mean(subset(tempRange, treatment=="heat")$temp_C)
```

```
## [1] 21.81587
```

## Ulcer Statistics

Conducting regression analyses for differences in minor ulcers across temperature treatments

```

# we are using the data frame lesion_max
# which denotes the maximum number of ulcers per individual cucumber

# finding the distribution that best fits our response variable (max_lesions)
gamlss::fitDist(max_lesions, data=lesion_max, type="counts", try.gamlss=TRUE)

```

```

##      |
##      system is computationally singular: reciprocal condition number = 1.18607e-17
##      |
##      Lapack routine dgesv: system is exactly singular: U[2,2] = 0
##      |

```

```

##
## Family:  c("GEOM", "Geometric")
## Fitting method: "nlminb"
##
## Call:  gamlssML(formula = y, family = DIST[i])
##
## Mu Coefficients:
## [1]  0.9651
##
## Degrees of Freedom for the fit: 1 Residual Deg. of Freedom    55
## Global Deviance:      239.135
##           AIC:      241.135
##           SBC:      243.16

```

```

# geometric distribution is best (woohoo!)

```

```

# creating a full model
ulcers_Full_model =gamlss(max_lesions ~ mean_weight + Treatment + poop + evisceration+
  random(as.factor(Table_ID)) + random(as.factor(Bucket_ID)),
  family = GEOM(), data = lesion_max)

```

```

## GAMLSS-RS iteration 1: Global Deviance = 234.4137
## GAMLSS-RS iteration 2: Global Deviance = 234.4131

```

```

#backward model selection to find top model
step.lesions.backward <- stepGAIC(ulcers_Full_model,
  direction = "backward", trace = F)

```

```

## Start:  AIC= 250.98
## max_lesions ~ mean_weight + Treatment + poop + evisceration +
## random(as.factor(Table_ID)) + random(as.factor(Bucket_ID))

```

```

summary(step.lesions.backward) # null model is best fit

```

```

## *****
## Family:  c("GEOM", "Geometric")
##
## Call:  gamlss(formula = max_lesions ~ 1, family = GEOM(),
## data = lesion_max, trace = FALSE)

```

```
##
## Fitting method: RS()
##
## -----
## Mu link function: log
## Mu Coefficients:
##           Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.9651      0.1570   6.146 9.35e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## -----
## No. of observations in the fit: 56
## Degrees of Freedom for the fit: 1
##      Residual Deg. of Freedom: 55
##              at cycle: 1
##
## Global Deviance:      239.1351
##           AIC:        241.1351
##           SBC:        243.1604
## *****
```

## Mortality Statistics

Using a kruskal-wallis test to examine differences in mortality across temperature treatments We also used logistic regression to test for within-treatment factors affecting survival

```
# Statistics! First, kruskal-wallis test
kruskal.test(death ~ treatment, data = DeathData)
```

```
##
## Kruskal-Wallis rank sum test
##
## data: death by treatment
## Kruskal-Wallis chi-squared = 11.383, df = 2, p-value = 0.003374
```

```
# p-value = 0.003374
# second, Dunn Test for pairwise comparisons
FSA::dunnTest(death ~ treatment, data = DeathData)
```

```
## Dunn (1964) Kruskal-Wallis multiple comparison
```

```
## p-values adjusted with the Holm method.
```

```
##      Comparison      Z      P.unadj      P.adj
## 1 control - heat -2.935046 0.003334979 0.010004937
## 2 control - room  0.000000 1.000000000 1.000000000
## 3 heat - room    2.935046 0.003334979 0.006669958
```

```

# We looked at what non-treatment factors affected death to learn more about
# what may increase a heated cucumber's chance of death.
# Full model, no random effects. BI() used as family as death is represented by
# a 1, and survival a 0. Respiratory evisceration is omitted because all those
# that eviscerated their respiratory tree died.
death.mod.full <- gamlss(death ~ evisceration + poop + in_droop + in_squeeze + weight_g,
                        family = BI(),
                        data = DeathData)

```

```

## GAMLSS-RS iteration 1: Global Deviance = 30.7848
## GAMLSS-RS iteration 2: Global Deviance = 30.7845

```

```

# Forwards selection.
backward.death.mod <- stepGAIC(death.mod.full,
                              direction = "backward",
                              trace = F)

```

```

## Start:  AIC= 44.78
## death ~ evisceration + poop + in_droop + in_squeeze + weight_g

```

```

# Checking out the model output:
formula(backward.death.mod)

```

```

## death ~ 1

```

```

## death ~ 1
summary(backward.death.mod)

```

```

## *****
## Family:  c("BI", "Binomial")
##
## Call:  gamlss(formula = death ~ 1, family = BI(), data = DeathData,
##           trace = FALSE)
##
## Fitting method: RS()
##
## -----
## Mu link function:  logit
## Mu Coefficients:
##           Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -2.3224      0.4686  -4.956 7.25e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## -----
## No. of observations in the fit:  56
## Degrees of Freedom for the fit:  1
##      Residual Deg. of Freedom:  55
##              at cycle:  2
##
## Global Deviance:      33.6988

```

```
##          AIC:      35.6988
##          SBC:      37.72415
## *****
```

```
## null model is best explanation!
```

## Stiffness Statistics

First, we examine the correlation between the two stiffness scores with `rcorr`. Then, we use `clmm()` from `ordinal` package to analyze data. We also use `dredge` from `MuMIn` to do model selection (code not run because of long processing times) `** Droop = structural maintenance, Squeeze = antipredator**`

```
# examining correlation between droop and squeeze
```

```
Hmisc::rcorr(stiff$Squeeze_score, stiff$Droop_score, type="spearman")
```

```
##      x      y
## x 1.00 0.53
## y 0.53 1.00
##
## n= 431
##
##
## P
##      x      y
## x      0
## y      0
```

```
# only include dates from Nov 9 to Nov 13
```

```
# aka the day before, days during, and day after the heat treatment
```

```
stiff_heat = subset(stiff, Date < "2021-11-14")
```

```
# full model for squeeze score
```

```
Squeeze_Full = ordinal::clmm(as.factor(Squeeze_score) ~ Treatment + as.factor(Date) + Treatment*as.factor(
  (1|Unique_ID) + (1|Bucket_ID) + (1|Table_ID), na.action="na.fail",
  data = stiff_heat)
```

```
# model selection code, not run because of processing time
```

```
# dredge.Squeeze <- MuMIn::dredge(Squeeze_Full)
```

```
# view(dredge.Squeeze) # top model does not include interaction term
```

```
# top squeeze model
```

```
Squeeze_Top = ordinal::clmm(as.factor(Squeeze_score) ~ Treatment + as.factor(Date) +
  (1|Unique_ID) + (1|Bucket_ID) + (1|Table_ID), na.action="na.fail",
  data = stiff_heat)
summary(Squeeze_Top)
```

```
## Cumulative Link Mixed Model fitted with the Laplace approximation
```

```
##
```

```
## formula: as.factor(Squeeze_score) ~ Treatment + as.factor(Date) + (1 |
```

```
## Unique_ID) + (1 | Bucket_ID) + (1 | Table_ID)
```

```
## data: stiff_heat
```

```
##
```

```
## link threshold nobs logLik AIC niter max.grad cond.H
## logit flexible 277 -164.59 351.19 623(2470) 1.08e-04 7.0e+01
##
## Random effects:
## Groups Name Variance Std.Dev.
## Unique_ID (Intercept) 2.486 1.577
## Bucket_ID (Intercept) 0.000 0.000
## Table_ID (Intercept) 0.000 0.000
## Number of groups: Unique_ID 56, Bucket_ID 30, Table_ID 6
##
## Coefficients:
## Estimate Std. Error z value Pr(>|z|)
## TreatmentRoom -3.0387 0.7475 -4.065 4.80e-05 ***
## TreatmentHeat -4.7620 0.8182 -5.820 5.88e-09 ***
## as.factor(Date)2021-11-10 -3.0850 0.6344 -4.863 1.16e-06 ***
## as.factor(Date)2021-11-11 -3.3929 0.6432 -5.275 1.33e-07 ***
## as.factor(Date)2021-11-12 -3.0000 0.6319 -4.748 2.06e-06 ***
## as.factor(Date)2021-11-13 -3.6206 0.6644 -5.450 5.05e-08 ***
## ---
## Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Threshold coefficients:
## Estimate Std. Error z value
## 0|1 -10.189 1.149 -8.872
## 1|2 -5.844 0.871 -6.710
```

#### *## full model for droop score*

```
Droop_Full = ordinal::clmm(as.factor(Droop_score)~ Treatment + as.factor(Date) + Treatment*as.factor(Date) +
  (1|Unique_ID) +(1|Bucket_ID)+(1|Table_ID), na.action="na.fail",
  data = stiff_heat)
```

*# model selection code, not run because of processing time*

```
#dredge.Droop <- MuMIn::dredge(Droop_Full)
```

*#view(dredge.Droop) # again, top model does not include interaction term*

#### *# top model for droop score*

```
Droop_Top = ordinal::clmm(as.factor(Droop_score)~ Treatment + as.factor(Date) +
  (1|Unique_ID) +(1|Bucket_ID)+(1|Table_ID), na.action="na.fail",
  data = stiff_heat)
summary(Droop_Top)
```

## Cumulative Link Mixed Model fitted with the Laplace approximation

##

## formula:

```
## as.factor(Droop_score) ~ Treatment + as.factor(Date) + (1 | Unique_ID) +
```

```
## (1 | Bucket_ID) + (1 | Table_ID)
```

```
## data: stiff_heat
```

##

```
## link threshold nobs logLik AIC niter max.grad cond.H
```

```
## logit flexible 277 -176.18 374.36 963(1952) 9.59e-05 9.6e+01
```

##

## Random effects:

```
## Groups Name Variance Std.Dev.
```

```
## Unique_ID (Intercept) 3.558e-01 5.965e-01
```



```
## Bucket_ID (Intercept) 0.000e+00 0.000e+00
## Table_ID (Intercept) 1.352e-14 1.163e-07
## Number of groups: Unique_ID 56, Bucket_ID 30, Table_ID 6
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## TreatmentRoom      -2.9923    0.5754  -5.201 1.99e-07 ***
## TreatmentHeat       -4.0464    0.6060  -6.677 2.44e-11 ***
## as.factor(Date)2021-11-10 -0.9763    0.5244  -1.862  0.0627 .
## as.factor(Date)2021-11-11 -2.2979    0.5284  -4.349 1.37e-05 ***
## as.factor(Date)2021-11-12 -2.2190    0.5265  -4.214 2.50e-05 ***
## as.factor(Date)2021-11-13 -2.1230    0.5409  -3.925 8.66e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Threshold coefficients:
##      Estimate Std. Error z value
## 0|1  -7.5304    0.8297  -9.077
## 1|2  -4.8521    0.6942  -6.989
```

## Spawning Statistics

To look at per-bucket spawning we simply use KW tests and observed no differences

```
# Find the number of cucumber bins that spawned.
sum(spawnData$spawn)
```

```
## [1] 11
```

```
# 11 bins spawned.
```

```
# Statistics on spawn data as a whole.
kruskal.test(spawn ~ treatment, data = spawnData)
```

```
##
## Kruskal-Wallis rank sum test
##
## data: spawn by treatment
## Kruskal-Wallis chi-squared = 1.9426, df = 2, p-value = 0.3786
```

```
FSA::dunnTest(spawn ~ treatment, data = spawnData)
```

```
## Dunn (1964) Kruskal-Wallis multiple comparison
```

```
## p-values adjusted with the Holm method.
```

```
##      Comparison      Z  P.unadj  P.adj
## 1 control - heat -0.912434 0.3615403 0.7230805
## 2 control - room -1.368651 0.1711084 0.5133251
## 3 heat - room -0.456217 0.6482339 0.6482339
```

```
# p = 0.379
```

## Evisceration Statistics

We model the effect of treatment, weight, and guts status, along with random effects, on evisceration

```
sum(EviscData$evisceration)
```

```
## [1] 12
```

```
# N = 12 eviscerated <- should be n=13
```

```
# full model for evisceration
```

```
evisc.mod.full <- gamlss(evisceration ~  
                        treatment + weight_g + poop + bucketID +  
                        random(tableID),  
                        family = BI(),  
                        data = EviscData)
```

```
## GAMLSS-RS iteration 1: Global Deviance = 3e-04
```

```
## GAMLSS-RS iteration 2: Global Deviance = 1e-04
```

```
# backward model selection
```

```
backward.evisc.mod <- stepGAIC(evisc.mod.full,  
                              direction = "backward",  
                              trace = F)
```

```
## Start: AIC= 64
```

```
## evisceration ~ treatment + weight_g + poop + bucketID + random(tableID)
```

```
formula(backward.evisc.mod)
```

```
## evisceration ~ weight_g + poop
```

```
## evisceration ~ poop + weight_g
```

```
summary(backward.evisc.mod)
```

```
## *****
```

```
## Family: c("BI", "Binomial")
```

```
##
```

```
## Call: gamlss(formula = evisceration ~ weight_g + poop, family = BI(),
```

```
## data = EviscData, trace = FALSE)
```

```
##
```

```
## Fitting method: RS()
```

```
##
```

```
## -----
```

```
## Mu link function: logit
```

```
## Mu Coefficients:
```

```
## Estimate Std. Error t value Pr(>|t|)
```

```

## (Intercept)  1.712596   1.128124   1.518   0.1349
## weight_g    -0.004309   0.002029  -2.124   0.0383 *
## poop1       -2.780415   1.120153  -2.482   0.0163 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## -----
## No. of observations in the fit: 56
## Degrees of Freedom for the fit: 3
##      Residual Deg. of Freedom: 53
##                      at cycle: 2
##
## Global Deviance:      44.55608
##           AIC:        50.55608
##           SBC:        56.63214
## *****

```

```

# poop (p = 0.0163) and weight (p = 0.0383) are both significant in explaining
# variation in evisceration. Weight estimate = -0.004309.

```

## Figures

### Figure 3 - temperature trends

```
TempPlot <- ggplot(data = Temp_Time,
  aes(x = date_time,
      y = temp_C,
      # Ignore warning about 'unknown aesthetic'... this code is
      # still important and functioning!
      fill = bucketID)) +
  geom_line(color="black")+
  scale_x_datetime(date_breaks = "1 day", date_labels = "%b %d") +
  geom_hline(aes(yintercept = 22,
    colour = "22°C"),
    size = 1,
    alpha = 0.8) +
  geom_hline(aes(yintercept = 17,
    colour = "17°C"),
    size = 1,
    alpha = 0.8) +
  geom_hline(aes(yintercept = 12,
    colour = "12°C"),
    size = 1,
    alpha = 0.8) +
  labs(x = "Date",
    y = "Temperature (°C)",
    colour = "Treatment Temperature") +
  scale_colour_manual(values = c("Gold", "Orange", "Red")) +
  guides(colour = guide_legend(reverse = TRUE)) +
  scale_y_continuous(breaks = c(12, 14, 16, 18, 20, 22, 24))+
  theme_bw()+
  theme(panel.grid=element_blank())

ggsave("Fig3_temp_plot.png", TempPlot, device = "png", path = here("figures"), width=8, height=5)
```

### Figure 4 and 5 Stiffness

Figure 4 = antipredator stiffness Figure 5 = structural stiffness

```
# making data for plotting
stiff_plot = stiff %>%
  dplyr::mutate(Squeeze_score = as.factor(Squeeze_score),
    Droop_score = as.factor(Droop_score)) %>%
  dplyr::filter(Date != "2021-11-14")

# making vector with facet labels
facet.labs=c("12°C", "17°C", "22°C")
names(facet.labs)=c("Control", "Room", "Heat")

# FIGURE 4
# plotting antipredator response
```

```

squeeze_plot =
  ggplot(data =stiff_plot, aes(x=as.factor(Date), fill=Squeeze_score))+
  geom_bar(alpha=0.8, color="black", size=0.5) +
  scale_x_discrete(breaks=c("2021-11-09", "2021-11-10", "2021-11-11", "2021-11-12", "2021-11-13", "2021-11-14",
                             labels=c(1,2,3,4,5,7,12))+
  xlab("Experiment Day") + ylab("Number of Sea Cucumbers")+
  geom_vline(xintercept=1.5, linetype=2, size=1)+ geom_vline(xintercept=4.5, linetype=2, size=1)+
  scale_y_continuous(expand=c(0,0))+
  scale_fill_manual(name="Antipredator Stiffness", labels=c("0 - Soft", "1 - Standard", "2 - Stiff"), values=c("burlywood4", "tan", "beige"))+
  theme_bw()+
  theme(strip.text.y = element_text(size =12),
        panel.grid=element_blank(),
        legend.position="right")+
  facet_grid(Treatment~., labeller=labeller(Treatment = facet.labs))

ggsave("figures/Fig4_antipredator_stiffness.jpg", plot=squeeze_plot, width=6, height=5)

# FIGURE 5
# plotting structural maintenance
droop_plot =
  ggplot(data =stiff_plot, aes(x=as.factor(Date), fill=Droop_score))+
  geom_bar(alpha=0.8, color="black", size=0.5) +
  scale_x_discrete(breaks=c("2021-11-09", "2021-11-10", "2021-11-11", "2021-11-12", "2021-11-13", "2021-11-14",
                             labels=c(1,2,3,4,5,7,12))+
  xlab("Experiment Day") + ylab("Number of Sea Cucumbers")+
  geom_vline(xintercept=1.5, linetype=2, size=1)+ geom_vline(xintercept=4.5, linetype=2, size=1)+
  scale_y_continuous(expand=c(0,0))+
  scale_fill_manual(name="Structural Stiffness", labels=c("0 - Soft", "1 - Standard", "2 - Stiff"),
                    values=c("burlywood4", "tan", "beige"))+
  theme_bw()+
  theme(strip.text.y = element_text(size =12),
        panel.grid=element_blank(),
        legend.position="right")+
  facet_grid(Treatment~., labeller=labeller(Treatment = facet.labs))

ggsave("figures/Fig5_structural_stiffness.jpg", plot=droop_plot, width=6, height=5)

```

Figure S1: Max minor ulcers across treatments

```

treatlabs = c("12°C", "17°C", "22°C")
names(treatlabs)=c("Control", "Room", "Heat")

ulcers_max = ggplot(data=lesion_max, aes(x=Treatment, y=max_lesions, fill=Treatment))+
  geom_boxplot(outlier.shape= NA, color="black", alpha=0.8)+
  geom_point(alpha=0.5, position=position_dodge2(0.2), color="black")+
  scale_y_continuous(expand=c(0,0), limits = c(-1,13.2))+
  scale_x_discrete(labels=c("12°C", "17°C", "22°C"))+
  scale_fill_manual(values=c("Gold", "Orange", "Red"))+
  ylab("Maximum Lesions / Indiv.") + xlab("Treatment")+
  theme_bw()+
  theme(panel.grid=element_blank(),
        legend.position="none")

```

```
ggsave("figures/FigS1_ulcers_max.jpg",plot=ulcers_max, width=6, height=5)
```

Figure S2 : Correlation between antipredator and structural stiffness

```
squeeze_droop_cor = ggplot(data=stiff, aes(x=Squeeze_score, y=Droop_score, color=Treatment))+
  geom_jitter(width=0.1, height=0.1, alpha=0.6)+
  scale_color_manual(labels=c("Control (12°C)", "Room (17°C)", "Heat (22°C)"), values=c("Gold", "Orange",
  theme_bw()+
  scale_x_continuous(breaks=seq(0,2.3,1), limits = c(-0.2,2.3))+
  scale_y_continuous(breaks=seq(0,2.3,1), limits = c(-0.2,2.3))+
  xlab("Antipredator Defense") + ylab("Structural Maintenance")+
  theme(strip.text.y = element_text(size =12),
  legend.position="NONE")+
  facet_grid(Treatment~., labeller=labeller(Treatment = facet.labs))

ggsave("figures/FigS2_squeeze_droop_correlation.jpg",plot=squeeze_droop_cor, width=6, height=4)
```

Figure S3: Model outputs from stiffness models (effect size & 95% CI)

```
# First, run models and save model objects (in this case Droop_Top and Squeeze_Top)
# see code above under Stiffness Statistics
# make output data frame with coefficient and confidence intervals
droop_output = data.frame(coef = coef(Droop_Top), # coef
  upper = confint(Droop_Top)[,2], # upper CI
  lower = confint(Droop_Top)[,1], # lower CI
  stiff = "Structural Maintenance")[-c(1,2),]
squeeze_output = data.frame(coef = coef(Squeeze_Top), # coef
  upper = confint(Squeeze_Top)[,2], # upper CI
  lower = confint(Squeeze_Top)[,1], # lower CI
  stiff= "Antipredator Defense")[-c(1,2),]

stiff_output= rbind(droop_output, squeeze_output) # bind the two outputs together for plotting

stiff_output$Treatment=c("Temp:17°C", "Temp:22°C", "Day 2", # naming treatments (for gg)
  "Day 3", "Day 4", "Day 5")
stiff_output$Variable = c("Temp", "Temp", "Date", "Date", "Date", "Date")# naming variables (for gg)

# plotting the output
stiffness_output = ggplot(data=stiff_output, aes(x=coef, y=Variable, color=Treatment))+
  geom_point(position=position_dodge(0.8))+
  geom_errorbar(aes(xmin=lower, xmax=upper), size=1, width=0.4, position=position_dodge(0.8))+
  geom_vline(xintercept=0, linetype=2)+
  xlab("Coefficient (Log Odds) ± 95% Confidence Interval")+
  scale_color_manual("Variable Category",values=c("grey30", "grey45", "grey60", "grey75",
  "goldenrod", "darkorange4"))+

  facet_grid(stiff~.)+
  theme_bw()
```

```
ggsave("figures/FigS3_stiffness_model_output.jpg",plot=stiffness_output, width=6, height=4)
```