Energetic mismatch induced by warming decreases leaf litter decomposition by aquatic detritivores

Theme08 - Introduction to Systems Biology
Reproducing a Research Article

Figure 1: Gammarus fossarum

Student: Vincent Talen
Student number: 389015
Class: BFV2
Study: Bio-Informatics
Institute: Institute for Life Science & Technology
University: Hanze University of Applied Sciences
Teacher: Tsjerk Wassenaar
Date: July 26, 2022

Abstract

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Tincidunt lobortis feugiat vivamus at augue eget arcu dictum. Arcu ac tortor dignissim convallis aenean et. Sed vulputate odio ut enim blandit volutpat maecenas. Ut diam quam nulla porttitor massa. Ipsum dolor sit amet consectetur adipiscing elit duis. Phasellus faucibus scelerisque eleifend donec pretium. Varius duis at consectetur lorem donec massa sapien. Eget dolor morbi non arcu risus quis varius. Id semper risus in hendrerit gravida rutrum quisque. Pellentesque habitant morbi tristique senectus et. Ut etiam sit amet nisl. Egestas fringilla phasellus faucibus scelerisque eleifend donec pretium vulputate sapien. Nibh tellus molestie nunc non blandit massa enim. Viverra mauris in aliquam sem fringilla ut. Mollis aliquam ut porttitor leo a diam. Sodales ut etiam sit amet nisl purus in mollis nunc. Pellentesque habitant morbi tristique senectus et netus et malesuada fames.

Table of Contents

List of Figures

List of Tables

1 Introduction

1.1 Goal

1.2 Theory

2 Methods

2.1 The software model

2.2 Model configuration

RMR was calculated as the slope of oxygen depletion over a ca 35 min timeframe, corrected for the mean linear trend over time of oxygen concentration in controls as follows: RMR = (ci – cf)/(ti – tf), where ci and cf are initial and final oxygen concentrations (μmol O2/L) in the wells and ti and tf are initial and final times points (day-1) of the respiration experiment. Values were converted into C release rate (μg C/day) assuming a respiratory coefficient of 0.78 (as estimated for Gammarus pulex, Wright & Wright, 1979).

Statistical Analysis Equations

The following equations were used to express the mass (M in mg) and temperature (T in Kelvin) dependence of individual RMR and IR:

$$I = \alpha M^b e^{Ea\left(\frac{T-T_0}{k_B T_0 T}\right)} \tag{1a}$$

$$I = \alpha M^b e^{p\left(\frac{T-T_0}{k_B T_0 T}\right) - q\left(\frac{T-T_0}{k_B T_0 T}\right)^2} \tag{1b}$$

The standard MTE formulation (1a) is simply a particular case of the quadratic formulation (1b) where $q = 0$ and the equation is reduced to the MTE model where p can thus be interpreted as the activation energy.

Energetic efficiency was also calculated as follows: $E = (IR/RMR) * A_T$, where the ratio of IR to RMR is the ingestion to metabolism efficiency and A_T is the assimilation efficiency at temperature T. The temperature (T in Kelvin) dependence of assimilation efficiency was expressed, using empirical equations and values for detritivores Assimilation efficiency was following a logistic equation with the MTE equation both at the numerator and the denominator With the following formulation, assimilation efficiency is confined between 0 and 1 (no or complete assimilation):

$$A_T = \frac{\alpha e^{Ea\left(\frac{T-T_0}{k_B T_0 T}\right)}}{1 + \alpha e^{Ea\left(\frac{T-T_0}{k_B T_0 T}\right)}} \tag{2}$$

Consumer-Resource Dynamics Model

Below are the ordinary differential equations describing temporal change in leaf litter standing stocks (L) and Gammarus population biomass (G) (Equation 3a and 3b).

$$\frac{dL}{dt} = I - f(L)_{\mathrm{T}} G - k_{\mathrm{T}} L \tag{3a}$$

$$\frac{dG}{dt} = G\left[f(L)_{\mathrm{T}} A - RMR_{\mathrm{T}}\right] \tag{3b}$$

3 Results

4 Discussion and Conclusion

4.1 Discussion

- Basically unreproducible in general, long pieces of code that were repeated over and over that could not be understood. No easy way to reproduce other than to immediately copy and paste multiple hundreds lines of code where only few values would be changed.
- Formulas for metabolic and ingestion rates were heavily rewritten from the base formula. They were unrecognizable so they were cleaned up to be understandable and resemble the actual formula more. It should also be noted that the position where the mean in the quadratic portion of the exponent is actually different from what would be done following the formula.

4.2 General conclusion and perspective

5 References

R Core Team. 2019. *R: A Language and Environment for Statistical Computing*. Vienna, Austria: R Foundation for Statistical Computing. https://www.R-project.org.

Réveillon, Tom, Thibaut Rota, Éric Chauvet, Antoine Lecerf, and Arnaud Sentis. 2022. "Energetic Mismatch Induced by Warming Decreases Leaf Litter Decomposition by Aquatic Detritivores." *Journal of Animal Ecology* n/a (n/a). https://doi.org/https://doi.org/10.1111/1365-2656.13710.

6 Appendices

Appendix A: 'model.R'

```r
## ---------------------------
##
## Script name: model.R
##
## Purpose of script: Implements the biological model of consumer-resource dynamics
##
## Author: Vincent Talen
##
## Date Created: 20 June 2022
##
## Email: v.k.talen@st.hanze.nl
##
## ---------------------------
##
## Notes:
##   - Goal: formulate formula functions in a more readable/recognizable manner
##
## ---------------------------


##############
#   Libs    #
##############
library(deSolve)


##############
#   Code    #
##############
## ---- loading in data ----
# Dataframe with respiration rate and ingestion rate (µg C/hour)
input_file <- "Data_Mismatch.txt"
data <- read.table(input_file, header = TRUE, dec = ".")
data <- na.omit(data)

# Consider individuals as a qualitative variable
data$Indiv <- as.factor(data$Indiv)

# Suppress outlier individuals and rows with negative values
data <- data[!data$Indiv %in% c("12", "30", "76", "78"), ]
data <- data[data$Respi > 0 & data$Nutri > 0, ]


## ---- mte formulations ----
# Convert temperature into the Boltzmann term (°K)
boltz_const <- 8.62 * 10^-5
inverse_temps <- 1 / ((data$Temp + 273.15) * boltz_const)
mean_inverse_temp <- mean(inverse_temps)

```

```r
# Quadratic function for metabolic rate (μg C/day)
calcMetabolicRate <- function(T.C, M) {
  alpha <- exp(2.41599)       # metabolic expression level at reference temperature
  b <- 0.62308                # body mass-scaling exponent
  p <- 0.66731                # curve steepness (of the relationship)
  q <- 0.21153                # quadratic term
  T.K <- T.C + 273.15         # convert temperature from Celsius to Kelvin

  # Repeating part with temperatures
  temp_dependancy_part <- (1 / (T.K * boltz_const)) - mean_inverse_temp

  # Calculate metabolic rate with full formula
  metabolic_rate <- alpha * M^b * exp(-p * temp_dependancy_part) * exp(-q * temp_dependancy_part^2)
  return(metabolic_rate)
}

# Quadratic function for ingestion rate (μg C/day)
calcIngestionRate <- function(T.C, M) {
  alpha <- exp(5.26814)       # ingestion expression level at reference temperature
  b <- 0.81654                # body mass-scaling exponent
  p <- 0.31876                # curve steepness (of the relationship)
  q <- 0.18909                # quadratic term
  T.K <- T.C + 273.15         # convert temperature from Celsius to Kelvin

  # Repeating part with temperatures
  temp_dependancy_part <- (1 / (T.K * boltz_const)) - mean_inverse_temp

  # Calculate ingestion rate with full formula
  ingestion_rate <- alpha * M^b * exp(-p * temp_dependancy_part) * exp(-q * temp_dependancy_part^2)
  return(ingestion_rate)
}


## ---- assimilation efficiency ----
# Assimilation efficiency function based on exponential decay (quadratic model)
# Is a logistic equation with the MTE equation both at the numerator and the denominator
calcAssimEff <- function(temp) {
  mte_equation <- (exp(-0.84730) * exp(0.16400 * ((temp + 273.15) - 285.65) / (boltz_const * 285.65 * (
  return(mte_equation / (1 + mte_equation))
}


## ---- attack rate parameter ----
leaf_mass <- 10.25 * 1000      # Mean initial leaf discs mass: 10.25 ± 0.68 mg
leaf_mass <- leaf_mass * 0.45  # Converted from mg to in μgC with a factor: 0.45

# Attack rate function based on exponential decay (quadratic model)
calcAttackRate <- function(temp, mass) {
  expt_duration <- 2
  decay_rate <- -log(1 - calcIngestionRate(temp, mass) * expt_duration / leaf_mass)
  attack_rate <- decay_rate / expt_duration
  return(attack_rate)
}
```

```r


## ---- handling time parameter ----
# Handling time function based on exponential decay (quadratic model)
calcHandlingTime <- function(temp, mass) {1 / (calcIngestionRate(temp, mass) / 1000) }

## ---- leaf decomposition- and respiration rate ----
# Reference temperature
ref_temp <- 10

# Function for the leaf litter microbial decomposition rate
calcLeafDecomp <- function(temp) { 0.00956 * exp(-0.37000 * (1 / ((temp + 273.15) * boltz_const) - ref_

# Function for the leaf litter respiration rate
calcLeafRespi <- function(temp) { 2.5 / 0.45 * 10^-3 * exp(-0.65000 * (1 / ((temp + 273.15) * boltz_cons


## ---- consumer-resource model ----
GammLeafModel <- function(temp, gamm_indv_mass, leaf_fall, gamm_start_biomass) {
  Nutri <- function(time, state, parms) {
    with(as.list(c(state, parms)), {
      fL <- a * L / (1 + a * h * L)         # Holling type II functional response
      dL <- -fL * G - k * L                 # Biomass changes of leaf litter stock
      dG <- G * (fL * A - M)                # Biomass changes of Gammarus population
      list(c(dL, dG))
    })
  }

  # Leaf litter fall event function
  leafFallEvent <- function(time, state, parms) {
    with(as.list(c(state, parms)), {
      return(c(L + leaf_fall, G))
    })
  }

  # Get time points to trigger litter fall event (first 15 days of the year)
  getFallTimesYearX <- function(year) { seq(year * 365 + 1, year * 365 + 15) }
  leaf_fall_times <- unlist(lapply(seq(0, 6), getFallTimesYearX))

  # Model parameters
  parameters <- c(
    M = calcMetabolicRate(temp, gamm_indv_mass) / 1000,  # Gammarus metabolic rate (in mgC/day)
    a = calcAttackRate(temp, gamm_indv_mass),            # Gammarus attack rate (in mgC/day)
    h = calcHandlingTime(temp, gamm_indv_mass),          # Gammarus handling time (in 1/day)
    A = calcAssimEff(temp),                              # Gammarus assimilation efficiency
    k = calcLeafDecomp(temp),                            # Leaf microbial decomposition (in 1/day)
    Rl = calcLeafRespi(temp)                             # [UNUSED???] Leaf respiration (in mgC/mgleaf/
  )

  # Times and starting conditions
  times <- seq(0, 365 * 7, by = 1)                       # Times in days for 7 years
  state <- c(L = leaf_fall, G = gamm_start_biomass)      # Starting biomasses (in g/m2)

  # Model output
```

```r
  out <- ode(time = times, func = Nutri, y = state, parms = parameters,
             events = list(func = leafFallEvent, time = leaf_fall_times))

  # Turn deSolve class object into dataframe and change very low and negative values to 0
  data_table <- as.data.table(out) %>% mutate(across(c(L, G), ~ fifelse(.x < 10^-3, 0, .x)))
  return(data_table)
}


## ---- temperature-size rule models ----
# Average TSR response
calcTSR.Avg <- function(temp, mass) {
  conv_fact <- 6.5                                  # Avg. conversion factor from dry to fresh mas
  change_slope <- -3.90 - 0.53 * log10(mass)        # Slope of change in mass per carbon
  change_prop <- log(1 + change_slope / 100)        # Proportion of change in mass per C
  change_const <- exp(log(mass) - 12.5 * change_prop)    # Constant of change in mass at 12.5°C

  dry_mass <- change_const * exp(change_prop * (temp))   # Dry body mass (mg)
  fresh_mass <- dry_mass / conv_fact                     # Fresh body mass (mg)
  return(dry_mass)
}

# Maximum TSR response
calcTSR.Max <- function(temp, mass) {
  conv_fact <- 6.5                                  # Avg. conversion factor from dry to fresh mas
  change_slope <- -8.0                              # Slope of change in mass per carbon
  change_prop <- log(1 + change_slope / 100)        # Proportion of change in mass per C
  change_const <- exp(log(mass) - 12.5 * change_prop)    # Constant of change in mass at 12.5°C

  dry_mass <- change_const * exp(change_prop * (temp))   # Dry body mass (mg)
  fresh_mass <- dry_mass / conv_fact                     # Fresh body mass (mg)
  return(dry_mass)
}
```

Appendix B: 'functions.R'

```r
## --------------------------
##
## Script name: functions.R
##
## Purpose of script: Functions
##
## Author: Vincent Talen
##
## Date Created: 20 June 2022
##
## Email: v.k.talen@st.hanze.nl
##
## --------------------------
##
## Notes:
##   - x
```

```r
##
## --------------------------


#############
#   Libs    #
#############
library(data.table)
library(ggpubr)
library(reshape2)
library(tidyverse)
source("r_code/model.R")


#############
# Functions #
#############
# ---- scenario data gathering and preparations ----
getScenarioDataList <- function(gamm_indv_mass, leaf_fall, gamm_start_biomass) {
  # Get data for given values for each temperature using the model function that performs an ode
  data_list <- lapply(temperatures, GammLeafModel, gamm_indv_mass, leaf_fall, gamm_start_biomass) %>%
    setNames(temperatures)
  return(data_list)
}

prepareBigDataFrame <- function(df_list) {
  # Function to get the population metabolism for a temperature with the population biomass
  getPopMetabolism <- function(cur_temp, gamm_pop_biomass) {
    # Get metabolic rate for current temperature
    meta_rate   <- calcMetabolicRate(cur_temp, gamm_indv_mass) / 1000  # Gammarus metabolic rate (in mg
    # Calculate population metabolism
    pop_metabolism <- meta_rate * gamm_pop_biomass
    return(fifelse(pop_metabolism < 0, 0, pop_metabolism))
  }
  # Function to get the population ingestion for a temperature with the population- and leaf biomasses
  getPopIngestion <- function(cur_temp, leaf_biomass, gamm_pop_biomass) {
    # Get ingestion- and attack rates for current temperature
    ingest_rate <- calcIngestionRate(cur_temp, gamm_indv_mass) / 1000  # Gammarus ingestion rate (in mg
    attack_rate <- calcAttackRate(cur_temp, gamm_indv_mass)            # Gammarus attack rate (in mg C/
    # Calculate population leaf ingestion
    pop_ingestion <- (attack_rate * leaf_biomass / (1 + attack_rate * 1 / (ingest_rate / 1000) * leaf_bi
    return(fifelse(pop_ingestion < 0, 0, pop_ingestion))
  }

  # Bind all dataframes from list to single big one and
  # drop last days to have 2555 days/rows left per temperature
  big_df <- rbindlist(df_list)[!time == 2555] %>%
    # Rename 'time' column to conform to naming scheme
    setnames("time", "Time") %>%
    # Add temperature and year columns to facilitate future calculations
    "$<-"(Temp, rep(temperatures, each = 2555)) %>%
    "$<-"(Year, rep(rep(1:7, each = 365), 5)) %>%
    # Add population metabolism and leaf ingestion columns
```

```r
    "$<-"(M, getPopMetabolism(.$Temp, .$G)) %>%
    "$<-"(I, getPopIngestion(.$Temp, .$L, .$G)) %>%
    # Set column order to a nicer one
    setcolorder(c("Time", "L", "G", "M", "I", "Temp", "Year"))
  return(big_df)
}

# ---- plotting list of scenario dataframes ----
createPlotForTemp <- function(cur_temp, cur_data) {
  # Divide L & G values to create a better readable plot
  divided_data <- copy(cur_data)
  set(divided_data, i = NULL, "L", divided_data$L / 10^5)
  set(divided_data, i = NULL, "G", divided_data$G / 10^5)

  # Create plot
  plot <- ggplot(divided_data, aes(x = time, y = value)) +
    # Set axis limits and step size
    scale_x_continuous(breaks = seq(0, 7 * 365, 365)) +
    ylim(NA, ceiling(max(divided_data[, -1])) + 1) +
    # Add the data (lines)
    geom_line(aes(y = L, color = "Leaf Litter Biomass")) +
    geom_line(aes(y = G, color = "Gammarus Fossarum Biomass")) +
    # Add styling
    labs(title = sprintf("%s°C", cur_temp), x = "", y = "") +
    # theme(plot.title = element_text(hjust = 0.075, vjust = -11)) +
    #theme(plot.margin = margin(0.1, 0.25, 0, 0, "cm")) +
    scale_color_manual(name = "", values = c("black","tomato2"),
                       limits = c("Leaf Litter Biomass", "Gammarus Fossarum Biomass"))
  return(plot)
}

plotScenario <- function(data, image_title, file_out) {
  # Use lapply to create plots for each temperature in the list and collect the legend from a plot
  plot_list <- lapply(seq_along(data), function(i) { createPlotForTemp(names(data)[i], data[[i]]) })
  plot_legend <- get_legend(plot_list[[1]])

  # Remove legends from the plots and add extracted legend to end of the list
  plot_list <- lapply(plot_list, function(cur_plot) { cur_plot + theme(legend.position = "none") }) %>%
    "[[<-"(length(plot_list) + 1, plot_legend)

  # Place plots and legend in an arranged grid
  col_num <- 3
  my.grid <- ggarrange(plotlist = plot_list, ncol = col_num, nrow = ceiling(length(plot_list) / col_num)
    annotate_figure(top = text_grob(image_title), bottom = text_grob("Time (d)"),
                    left = text_grob(bquote("Biomass "(10^5~ mg~ C~ m^-2)), rot = 90))

  # Save the created arranged grid with the lossless 'lzw' compression that greatly reduces file size
  ggsave(file_out, bg = "white", width=15, height=8, units="in", dpi=300, compression = "lzw")
  dev.off()
}
```

Appendix C: 'scenarios.R'

```r
## --------------------------
##
## Script name: scenarios.R
##
## Purpose of script: Simulate scenarios using functions from the functions.R module
##
## Author: Vincent Talen
##
## Date Created: 20 June 2022
##
## Email: v.k.talen@st.hanze.nl
##
## --------------------------
##
## Notes:
##   - x
##
## --------------------------
#setwd("BIN2-P8/endreport")


##############
#   Libs    #
##############
source("r_code/functions.R")


##############
#   Code    #
##############
# Temperatures to do simulations of
temperatures <- c(5, 10, 15, 20, 25)

#####################################
### SCENARIO 0: STANDARD SCENARIO ###
#####################################
# Gammarus mean body mass = 4.26 mgDM
# Annual leaf fall        = 300 gC/m2/an = 300 000 mgC/m2/an
# Gammarus density        = 30 mgDM/m2   = 15 mgC/m2

# Duration of the leaf fall in days
fall_duration_in_days <- 15

# Define scenario values
gamm_indv_mass <- 4.26
leaf_fall <- 300000 / fall_duration_in_days
gamm_start_biomass <- 15

# Get data for temperatures with values of current scenario
scen_df_list <- getScenarioDataList(gamm_indv_mass, leaf_fall, gamm_start_biomass)

```

```r
# Create plots in an arranged grid
file_out <- "figures/Population Dynamics Standard Scenario.tiff"
image_title <- "Standard Scenario: Population Dynamics over 7 years"
plotScenario(scen_df_list, image_title, file_out)

# Combine dataframes into one and add temperature, year, population metabolism- and ingestion columns
TestSD <- prepareBigDataFrame(scen_df_list)

##########################################################################################################
CutSD <- as.data.frame(setDT(TestSD)[TestSD[, tail(.I, -16), by = Year]$V1])
CutSD <- CutSD[!CutSD$Year == "1", ]

#############################
### Mean annual biomasses ###
#############################
# Calculate litter and Gammarus mean annual biomasses
MeanL=as.data.frame(setDT(CutSD)[, .(MeanL=mean(L)), by=list(Temp,Year)])
MeanG=as.data.frame(setDT(CutSD)[, .(MeanG=mean(G)), by=list(Temp,Year)])

# Calculate means and deviations over 6 years for annual biomasses
MeanLSD=as.data.frame(setDT(MeanL)[, .(MeanL=mean(MeanL), SdL=sd(MeanL)), by=list(Temp)])
MeanGSD=as.data.frame(setDT(MeanG)[, .(MeanG=mean(MeanG), SdG=sd(MeanG)), by=list(Temp)])

#################################################################
### Mean annual persistence times above biomass thresholds ###
#################################################################
# Calculate litter and Gammarus mean annual persistence times
ThLSD=CutSD[CutSD$L>60000,]
ThGSD=CutSD[CutSD$G>5000,]

TimeLSD=as.data.frame(setDT(ThLSD)[, .(TimeL=length(Time)), by=list(Temp,Year)])
TimeGSD=as.data.frame(setDT(ThGSD)[, .(TimeG=length(Time)), by=list(Temp,Year)])

# Calculate means and deviations over 6 years for the thresholds
TimeLSD=as.data.frame(setDT(TimeLSD)[, .(MeanTimeL=mean(TimeL), SdTimeL=sd(TimeL)), by=list(Temp)])
TimeGSD=as.data.frame(setDT(TimeGSD)[, .(MeanTimeG=mean(TimeG), SdTimeG=sd(TimeG)), by=list(Temp)])

#####################################################
### Mean biomass maximums and biomass decreases ###
#####################################################
# Find biomass maximums and minimums
CycleLSD=as.data.frame(setDT(TestSD)[, .(MaxLSD=findPeaks(L), MinLSD=findValleys(L)[seq(2,14,2)]), by=li
CycleGSD=as.data.frame(setDT(TestSD)[, .(MaxGSD=findPeaks(G), MinGSD=c(findValleys(G),2555)), by=list(Te
```