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

Vincent Talen
389015
BFV2
June 30, 2023
Tsjerk Wassenaar (WATS)

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

Theme08 - Introduction to Systems Biology
Reproducing a Research Article

Vincent Talen
389015
Bioinformatics
Institute for Life Science & Technology
Hanze University of Applied Sciences
Tsjerk Wassenaar (WATS)
June 30, 2023

Table of Contents

List of Abbreviations

MTE	Metabolic Theory of Ecology
ODE	Ordinary differential equation
RMR	Routine metabolic rate
IR	Leaf ingestion rate

List of Figures

List of Tables

1 Introduction

Climate change is a rising threat to the earth, the average overall surface temperature is predicted to increase by 0.2°C per decade and up to 2–5°C by the end of this century, leading to massive disruptions at all levels of biological organization across ecosystems *[1], [2]*. Life-history traits, population dynamics, species interactions and ecological processes are strongly influenced by temperature *[3]–[5]*, especially physiological traits related to energy acquisition and expenditure *[6]*, such as metabolic rate *[7], [8]* and ingestion rate *[9], [10]*, which together determine the energy balance of organisms. Furthermore, most organisms are ectotherms *[11]* on who temperature has an even greater effect *[12]*, thus a key in understanding ecosystems' response to global warming is understanding the thermal physiology of ectotherms *[13]*.

A powerful framework to investigate ecosystem functioning in the context of global warming is the Metabolic Theory of Ecology (MTE) *[6]*. It combines the effects of body mass and temperature on biochemical processes in order to predict individual physiological performances *[14], [15]*, this can then be scaled up from individuals to population, community and ecosystem levels *[16]*. As metabolic losses increase exponentially with warming, organisms generally increase energy supply through nutrient ingestion *[17], [18]*, but metabolism increases more rapidly than nutrient ingestion with temperature. The resulting mismatch causes a decreasing energetic efficiency as temperature rises *[19]*, but has not been studied or measured directly.

Despite the functional significance and vulnerability to warming of detritivore populations *[20], [21]*, most studies on the impact of global warming on consumer-resource dynamics have mainly focused on carnivore and herbivore populations. Detritivores are heterotrophs that consume plant litter and decompose them into smaller inorganic molecules, performing what is called the first stage of remineralisation. These inorganic compounds can then be used by primary producers, such as plants and algae, to synthesize new organic molecules, completing the nutrient cycle in the ecosystem. Thus, leaf litter decomposition by detritivores is a crucial process in the ecosystem as it allows the nutrients stored in organic matter to be recycled and reused by other organisms.

Previous studies have not yet fully explained how thermal constraints on detritivores scale up to their entire ecosystems. Thermal bio-energetic models are greatly relevant for studying the impact of temperature and body size changes on detritivore-resource dynamics *[22]–[24]* and understanding the balance between key physiological processes that determine detritivore fitness *[25]* is crucial for predicting the responses of populations and freshwater ecosystems to global warming *[19], [26], [27]*.

The goal of this research is to reproduce and improve on the research done by *Réveillon et al. [28]* on the modelling of the consumer-resource dynamics by greatly improving the model code written in R, resulting in better reproducibility of this research and it being more easily expandable. *Réveillon et al.* investigated the thermal energetic mismatch between energy demand (i.e. metabolic rate) and supply (i.e. ingestion rate) and simulated the consequences of this thermal mismatch for seasonal population dynamics and carbon fluxes *[28]*.

1.1 Theory

The consumer-resource model created by *Réveillon et al.* describes the seasonal dynamics of *Gammarus Fossarum* and Leaf Litter biomasses in a temperate stream, a diagram showing an overview of the system dynamics can be seen below in Figure 2. Even though the model assumes that all individuals of the population have the same body mass, the exploration on the effects of temperature-induced changes in population body size is still possible. The fluctuation in Gammarus population biomass is driven by the balance between carbon intake through food ingestion and carbon loss through respiration. Changes in leaf litter biomass are due to herbivory pressure of Gammarus on seasonal litter fall stock.

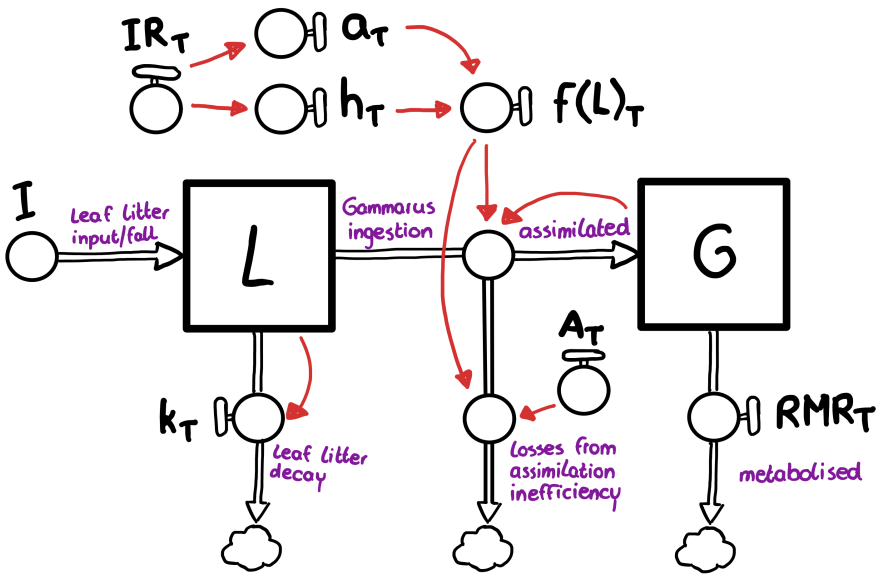

Figure 2: Diagram showing dynamics of the model (Made by Vincent Talen)

State variables

The model has two state variables, leaf litter standing stocks (L) and *Gammarus* population biomass (G), both in in $mg\,C/m^2$. Both state variables have an ordinary differential equation (ODE) that describes their temporal change, Equation 1a and 1b respectively.

$$\frac{dL}{dt} = I - f(L)_{\mathrm{T}} G - k_{\mathrm{T}} L \tag{1a}$$

$$\frac{dG}{dt} = G\left[f(L)_{\mathrm{T}} A_{\mathrm{T}} - RMR_{\mathrm{T}}\right] \tag{1b}$$

The standing stock of leaf litter is sustained by the seasonal leaf litter fall inputs (I) and decreases due to feeding activity by the *Gammarus* population ($f(L)_T G$) and litter decomposition ($k_T L$) by other degradation processes (e.g. microbial decomposition and leaching). Dynamics of the *Gammarus* population is described as the balance of carbon intake through litter ingestion ($f(L)_T\, A_T$) and loss through respiration (RMR_T), henceforth Routine Metabolic Rate (RMR). Each of the parameters of these two ODEs is described by their own equation and dependent on temperature, except for leaf litter input, all parameters are shown in Table 1 below.

Table 1: Model state variables' parameters

Parameter	Equation	Unit	Explanation
I	-	$mg\,C\,m^2$	Leaf litter input
$f(L)_\mathrm{T}$	2	$mg\,C\,mg\,C^{-1}\,day^{-1}$	Gammarus functional response
RMR_T	3b	$mg\,C/day$	Gammarus routine metabolic rate
A_T	4	-	Gammarus assimilation efficiency
k_T	5	day^{-1}	Leaf litter microbial decomposition

Parameter equations

The first parameter that is used in both differential equations is the *Gammarus* population feeding rate, which follows the Holling type II functional response (Equation 2).

$$f(L)_T = \frac{a_T L}{1 + a_T h_T L} \tag{2}$$

with a_T being the *Gammarus* attack rate on leaves (mg^{-2}/day), h_T the *Gammarus* handling time (day^{-1}), both at temperature T, and with L being the leaf litter biomass in $mg\,C/m^2$. Attack rate is estimated by assuming that the proportion of ingested leaf litter follows an exponential decay of time, which can be calculated as $DC_\mathrm{T} = -log(1 - IR_\mathrm{T} * t/M_L)$. Where IR_T is the daily leaf ingestion rate (IR) at temperature T ($mg\,C\,mg\,C^{-1}\,day^{-1}$) calculated with Equation 3b, t is the duration of the experiment (d) and M_L is the mean initial C mass of leaf discs in microcosms. After the decay rate is calculated the attack rate can finally be calculated by dividing the estimated decay rate by the experimental duration ($a_\mathrm{T} = DC_\mathrm{T}/t$). The handling time is calculated as the inverse of the ingestion rate ($h_\mathrm{T} = 1/IR_\mathrm{T}$).

To express the mass (M) and temperature (T) dependence of the RMR and IR of individuals the following equations were used: To express the dependence of RMR and IR of individuals on mass (M) and temperature (T) the following equations using the MTE formula were used:

$$I = \alpha M^b e^{Ea\left(\frac{T-T_0}{k_B T_0 T}\right)} \tag{3a}$$

$$I = \alpha M^b e^{p\left(\frac{T-T_0}{k_B T_0 T}\right) - q\left(\frac{T-T_0}{k_B T_0 T}\right)^2} \tag{3b}$$

where α is the metabolic or the ingestion expression level at reference temperature T_0, b is the mass-scaling exponent, M is the dry body mass (mg), Ea is the activation energy (eV) and k_B is the Boltzmann's constant ($8.62 * 10^{-5}\,eV\,K^{-1}$).
To allow for the investigation into the curvature strength of the relationship between the measured rate (I) and temperature, a deviation of the MTE expression is used within Equation 3b's exponential term. The curvature is described by the fitted polynomial first- and second-order terms `p` and `q`, respectively *[29], [30]*. If $q = 0$ is used in the quadratic formulation (Equation 3b) and the equation is reduced to the MTE model, then `p` can be interpreted as the activation energy *[29]*. This particular case is formulated as Equation 3a.

To express the temperature dependence of assimilation efficiency (A_T), empirical equations and values for detritivores from *Lang et al. [31]* are used. So the assimilation efficiency is confined between 0 and 1 (no assimilation or complete assimilation) a logistic equation is used where the MTE equation is used both at the numerator and denominator, resulting in Equation 4.

$$A_T = \frac{\alpha e^{Ea\left(\frac{T-T_0}{k_B T_0 T}\right)}}{1 + \alpha e^{Ea\left(\frac{T-T_0}{k_B T_0 T}\right)}} \tag{4}$$

where α is the normalization constant of assimilation efficiency, Ea is the activation energy (eV) and k_B is the Boltzmann's constant ($8.62 * 10^{-5}\,eV\,K^{-1}$).

The temperature dependence of microbial decomposition is expressed using the Arrhenius equation, since carbon fluxes in aquatic ecosystems are largely caused by microbial decomposition *[32]* causing leaf litter to also be affected by this.

$$k_T = k_{10^\circ C} e^{-Ea\left(\frac{1}{k_B T} - \frac{1}{283.15 k_B}\right)} \tag{5}$$

where $k_{10^\circ C}$ is the leaf litter decomposition rate at 10°C (283.15K), Ea is the activation energy (eV) and k_B is the Boltzmann's constant.

2 Materials and Methods

2.1 The software model

The model was implemented using the `R` programming language *[33]* (version 4.1.3), in combination with multiple packages/libraries that made it possible to perform the data manipulation and calculations. The table below (Table 2) shows the list of packages that were used for this project, including their exact versions. It is recommended to use the exact versions of the packages listed to guarantee compatibility when reproducing this project and model.

Table 2: Software and packages

Software	**Package**	**Version**
R		4.1.3
	data.table	1.14.2
	deSolve	1.3.4
	ggpubr	0.4.0
	lme4	1.1-29
	quantmod	0.4.20
	reshape2	1.4.4
	tidyverse	1.3.1

The `ode` function from the deSolve *[34]* package is the core tool used to implement the model, it applies the ordinary differential equations (ODE), that make up the model, over time with parameters. All the data is placed into `data.table`s from the `data.table` library *[35]*, allowing for fast and intuitive operations. To visualize the data and create plots the packages `ggpubr` *[36]*, `reshape2` *[37]* and `tidyverse` *[38]* were used. The `lme4` *[39]* and `quantmod` *[40]* packages were used to create models and prediction data used for the actual lines in the plots.

2.2 Model configuration

Réveillon et al. performed multiple laboratory experiments and statistical analyses to estimate values for the initial state variables and the parameters that together accurately describe the dynamics to develop the consumer-resource model *[28]*. The experimental values that *Réveillon et al.* used will also be used for this project in order to attain the goal of this project, namely to improve on the code implementation of the consumer-resource model. All values below will come from their research article *([28])*.

Each scenario that has been simulated for this project has been done for the same five temperatures (5, 10, 15, 20 and 25 degrees Celsius) and were run for the same 7-year duration, of which the first year is excluded because of transient dynamics following the input of leaf litter and detritivores in the system. The annual leaf litter fall was represented as an event of 15 consecutive days at the beginning of each year, with an even amount of leaf litter fall each day as to mimic the phenology of forest vegetation in the study region *[28]*. It was also assumed that each Gammarus individual had the same body mass, meaning that a population size structure was not implemented.

Since the consumer-resource model consists of two state variables and almost each of their parameters are again expressed as equations which also have their own parameters, a lot of values were used to describe and simulate the system dynamics. Most of the equations' parameters are static between scenarios, with only the mass (M) and temperature (T) changing. The static parameter values will be listed in a separate table for each equation, beginning with the global parameter values used in almost every equation or formula in Table 3.

Table 3: Global parameter values

Parameter	Value	Unit	Explanation
T	$5, 10, 15, 20, 25$	C	Temperatures simulations were run at
T_0	285.65	K	Reference temperature
k_B	$8.62 * 10^{-5}$	$eV\ K^{-1}$	Boltzmann's constant
M	4.26	mg	Gammarus mean individual dry body mass

Both the Metabolic- and Ingestion Rate are both calculated using Equation 3b, they thus *do* use the same parameters but *not* the same parameter values. The values that are used to calculate the metabolic rate are listed in Table 4 and the values used to calculate the ingestion rate are listed in Table 5.

Table 4: Metabolic Rate (3b) parameter values

Parameter	Value	Unit	Explanation
α	$e^{2.41599}$	-	Metabolic expression level at reference temperature T_0
b	0.62308	-	Mass-scaling exponent
p	0.66731	-	Curve steepness
q	0.21153	-	Quadratic term

Table 5: Ingestion Rate parameter values

Parameter	Value	Unit	Explanation
α	$e^{5.26814}$	-	Ingestion expression level at reference temperature T_0
b	0.81654	-	Mass-scaling exponent
p	0.31876	-	Curve steepness
q	0.18909	-	Quadratic term

For calculating the assimilation efficiency, estimates provided by *Lang et al. [31]* and the rescaled intercept (α) by *Réveillon et al. [28]* were used. These values are listed below in Table 6.

Table 6: Assimilation Efficiency parameter values

Parameter	Value	Unit	Explanation
α	$e^{-0.84730}$	-	Normalization constant
Ea	0.16400	eV	Activation energy
T_0	285.65	K	Reference temperature

Parameter values used for the attack rate formula can be seen in Table 7. Initial mass of the leaf discs is derived from the pre-weighed batches of six dry leaf discs the individuals were allowed to feed on during the experiment. The mean dry mass of these batches was $10.25mg$ *[28]*, which was then converted from dry mass to C content through the use of the the conversion factor (0.45) of dry mass to C content of leaf litter. Lastly the C mass was converted from mg to μg by multiplying by 1000.

Table 7: Attack rate formula static parameter values

Parameter	Value	Unit	Explanation
t	2	day	Duration of the feeding experiment
M_L	4612.5	$\mu g\, C$	Initial C mass of leaf discs in microcosms

The parameter values used in the Arrhenius equation that expresses the microbial decomposition rate of leaf litter are listed in Table 8 and are in situ estimates provided by *Follstad Shah et al. [41]*.

Table 8: Leaf Decomposition Rate parameter values

Parameter	Value	Unit	Explanation
$k_{10^\circ C}$	0.00956	day^{-1}	Litter decomposition rate at 10°C (283.15°K)
Ea	0.37000	eV	Activation energy

One of the attributes that was calculated as part of the analysis is the annual persistence time above thresholds for both litter standing stock and Gammarus stock, the threshold values that were used are listed in Table 9.

Table 9: Annual persistence time threshold values

Threshold	Value	Unit	Explanation
L	$6 * 10^4$	$mg\, C/m^2$	Threshold for litter standing stock
G	$5 * 10^3$	$mg\, C/m^2$	Threshold for Gammarus stock

Main analysis

For the main analysis three scenarios were simulated, the first was the reference scenario (TSR_R) with all its parameters values based on experimental estimates and the Gammarus mean body mass is constant across temperatures. Because body mass is temperature dependent *[42]* two other scenarios were simulated that implemented the dependency of temperature on body mass, these simulations were based on empirical results from a meta-analysis by *Forster et al. ([43])*: a scenario that uses the mean relationship between body mass and temperature for aquatic organisms (TSR_A) and a scenario that corresponds to the largest body size decrease with temperature (TSR_M). The values used for the main analysis' scenarios are listed below in Table 10.

Table 10: Main analysis initial state variables biomass values

State variable	Value	Unit	Explanation
L	300 000	$mg\, C/m^2$	Leaf litter stock biomass
G	15	$mg\, C/m^2$	Gammarus population biomass density
I	300 000	$mg\, C\, m^{-2}\, year^{-1}$	Annual leaf litter input

3 Results

The research by *Réveillon et al. [28]* on the modelling of the consumer-resource dynamics has been successfully reproduced and the code and figures have been greatly improved on, having been made more understandable and user-friendly. The code has been compacted, reformatted and converted to use dynamic functions used for every scenario, instead of repeating the exact same code for each scenario. Plot titles and legends have been added to the figures generated for the analysis and their general style has been changed so the results illustrated are more comprehensible. In this chapter the original figures and some code chunks will be shown next to the new, reproduced, figures and code chunks, depicting the successful reproduction of the research and the improvements that have been made.

3.1 Figures

As part of the main analysis a figure illustrating the population dynamics has been made for each of the three scenarios and three final figures comparing the three mains characteristics of the consumer-resource dynamics; the mean biomass, the mean biomass slope and the persistence time above the thresholds. Because the datapoints overlap in some cases a small horizontal jitter was added to improve visibility.

Population dynamics

The population dynamics of a scenario shows the biomasses of leaf litter and gammarus over the duration of the simulation per temperature. With this figure it can be perceived how the population dynamics behave and differ between temperatures and if the simulations look realistic and thus seem to be successful. The first figure shown in Figure 3 is the original made by *Réveillon et al.* and the second figure, Figure 4, is reproduced as part of this project. For this figure the added background grid can be used as a visual reference to actually see how high the biomass lines go, which was not possible before.

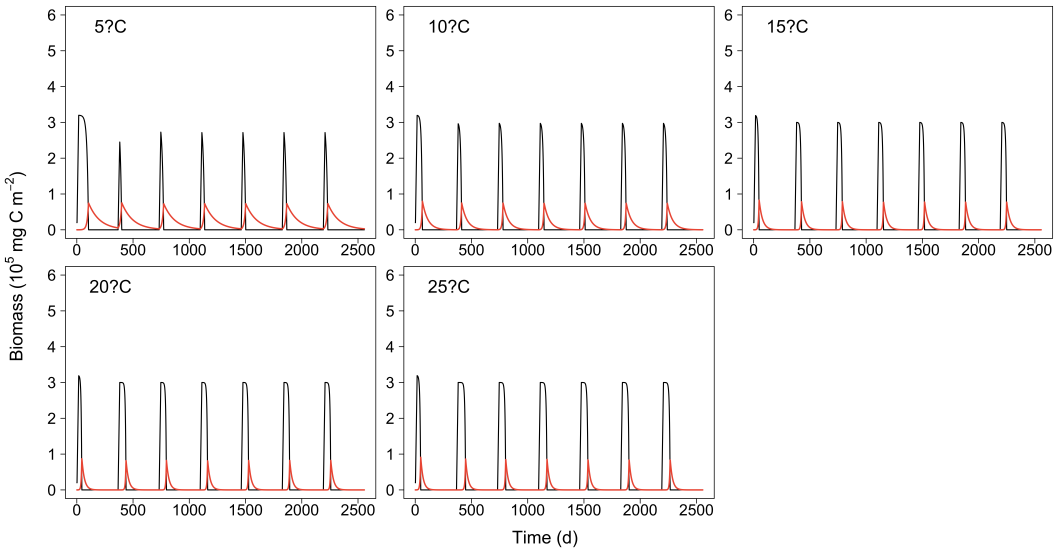

Figure 3: Population Dynamics SD (made by Réveillon et al.)

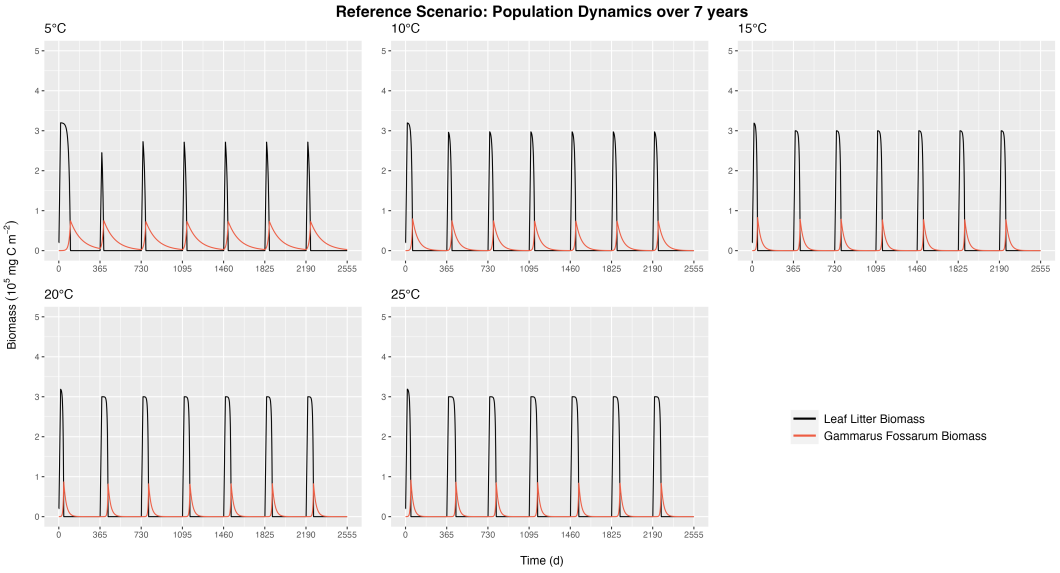

Figure 4: Population Dynamics Reference Scenario (made by Vincent Talen)

Mean Biomass Slope

The mean slope of biomass decreases are calculated from the 6 annual cycles of the three scenarios at 5 different temperatures and are shown as the points, prediction data was also made to plot the continuous lines with. In both the original figure (Figure 5) and the reproduced figure (Figure 6), it can be seen that temperature dependence of the Gammarus biomass heavily influences the slope and a heavier depencence results in a steeper decrease for Gammarus Fossarum biomass, this in turn also causes the slope of leaf litter decrease to become lower for higher temperatures. Just like the other reproduced figures a grid background, plot titles and a legend have been added to Figure 6.

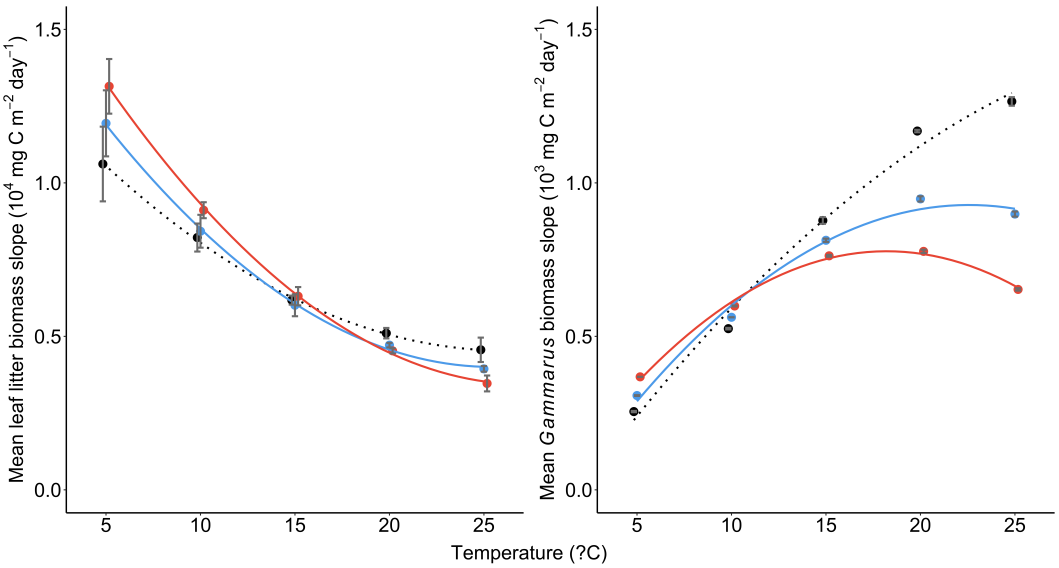

Figure 5: Mean Biomass Slope Scenarios (made by Réveillon et al.)

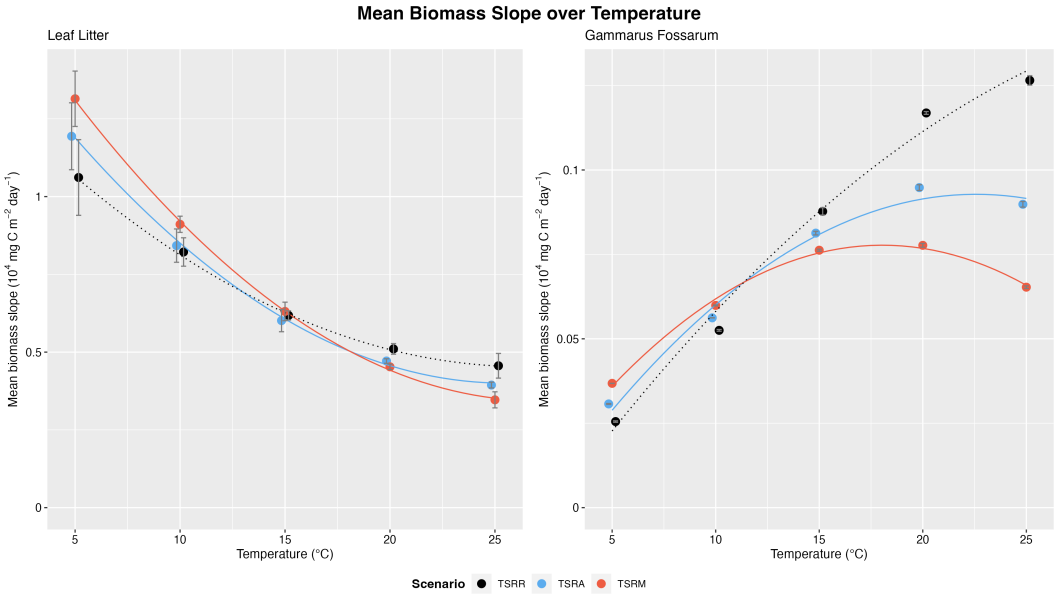

Figure 6: Mean Biomass Slope over Temperature (made by Vincent Talen)

Persistence Time

The original plots by *Réveillon et al.* can be seen in Figure 7 and the reproduced plots in Figure 8 and the same changes have been made as for the the mean biomass slope figure. To calculate the persistence time the amount of days the biomasses are above the thresholds specified in Table 9 are counted per year. In the resulting plots it can be seen that there is a negative effect of warming on the Gammarus biomass since the persistence time greatly decreases with higher temperatures, however once the temperature goes above 15 degrees Celsius this effect gets weaker causing the persistence time of the Gammarus biomass to go back slightly.

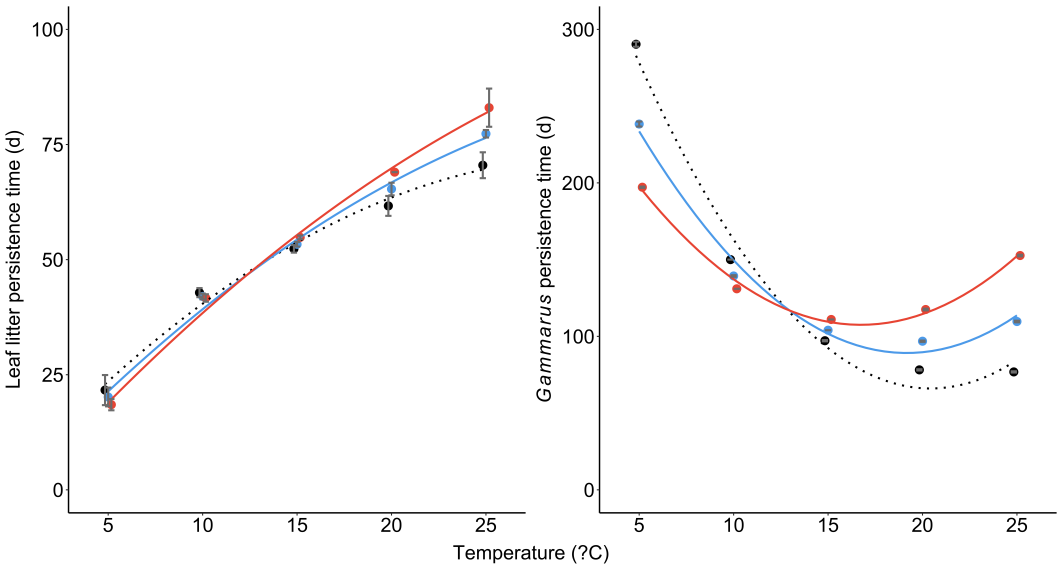

Figure 7: Persistence Time Scenarios (made by Réveillon et al.)

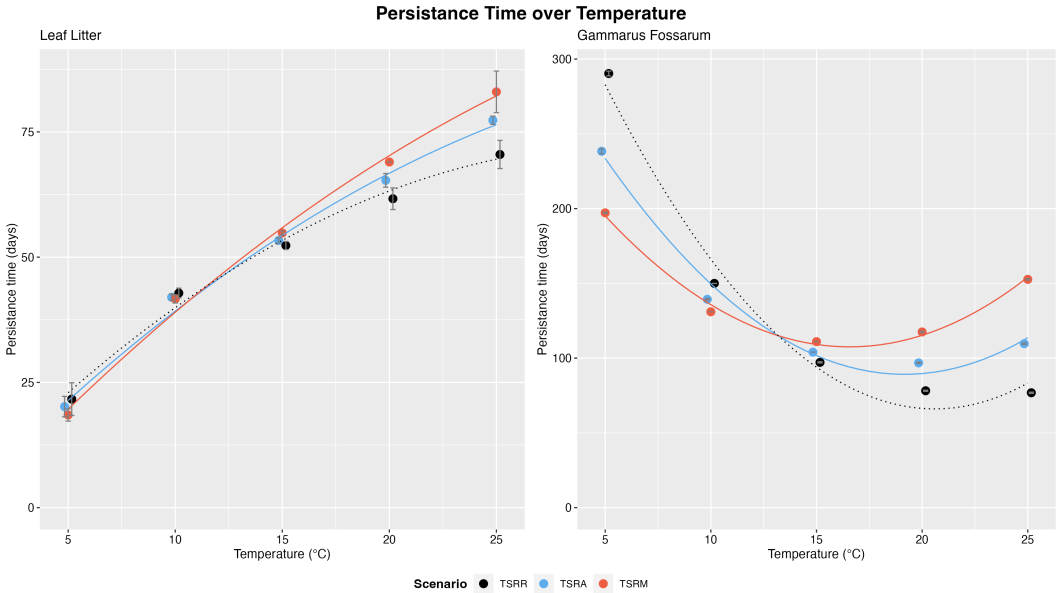

Figure 8: Persistance Time over Temperature (made by Vincent Talen)

3.2 R Code

The original code is a single file that is just over three-thousand lines long that mainly consists out of repeated code between and even within scenarios. The new code has been split into four files; the first only has the functions for the model itself, the second has functions to generate and combine data, the third has a single function that performs the entire simulation for a scenario and the fourth uses, and executes, the functions from the previous three to perform the analysis and creates the final figures. Almost the entire code base has been converted into functions and the code chunks implementing the model equations have been reformatted so they actually resemble their equations with the parameters being variables that can easily be changed.

Since there is a lot of code that has been written for this project, too much to show, only three chunks have been selected and are shown below to illustrate the idea and concept of some of the changes that have been made. For each example the original code will be shown and the equivalent of the new code, these are just partial chunks and might miss the context, the full code files are included as appendices but for optimal viewing experience it is adviced to open the files directly.

Implementation of equations

This first example shows how the functions that implement the equations have been rewritten to represent the equation more, allowing someone to better understand what the function does and making it easier to verify that the equation is correctly implemented. It is now also very intuitive to change parameter values. It needs to be mentioned is that line breaks have been added to the original code so the entire content can even fit it inside this report, the actual code is on one single line and even less readable.

```
AssimQuadra=function(Temp){
  (exp(-0.84730)*exp(0.16400*((Temp+273.15)-285.65)/(Boltz*285.65*(Temp+273.15))))
  /(1+(exp(-0.84730)*exp(0.16400*((Temp+273.15)-285.65)/(Boltz*285.65*(Temp+273.15)))))}
```

model.R:

```
calcAssimEff <- function(T.C) {
  alpha <- exp(-0.84730)    # normalization constant of assimilation efficiency
  Ea <- 0.16400             # activation energy
  T.0 <- 285.65             # reference temperature of 12.5 degrees Celsius in Kelvin
  T.K <- T.C + 273.15       # convert temperature from Celsius to Kelvin

  mte_equation <- alpha * exp( Ea * (T.K - T.0) / (boltz_const * T.0 * T.K) )
  return(mte_equation / (1 + mte_equation))
}
```

Improvement of simple list creation

Inside the wrapper function that implements the ODE model a list of time points is made for when the litter fall event needs to happen, this is done for the first 15 days for every year of the simulation. In the original code this is done completely manually and is hardcoded, in the new code a function has been made that is called with a `lapply` call (the R-equivalent of a for-loop) to dynamically created the time points.

```
# Define the years
y1=365; y2=2*y1; y3=3*y1; y4=4*y1; y5=5*y1; y6=6*y1

# Time points to trigger litter fall
FallTime=c(seq(1,15), seq(y1+1,y1+15),seq(y2+1,y2+15),seq(y3+1,y3+15),seq(y4+1,y4+15),seq(y5+1,y5+15),s
```

`model.R`:

```
  # 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))
```

Defining the cycles of scenario simulation

To calculate the decreasing slopes of biomass annually the cycles for the state variables first have to be found through their peaks and valleys. In the original code by *Réveillon et al.* this is done very statically and hardcoded, resulting in large code blocks of 10 lines that are up to 350 characters long! For each scenario this is done for both the state variables, these entire code blocks are thus copied and pasted for each scenario with only the variable names changing.

The new code completely changed the way this is done by implementing dataframe manipulation that is performed inside a dynamic function that is defined once and simply called for each scenario simulation. It can now also be followed which steps and calculations are done since it now more resembles a data pipeline.

```
# 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

# Define litter and Gammarus biomass cycles
t0=2555*0; t1=2555*1; t2=2555*2; t3=2555*3; t4=2555*4

CutL5SD=c(CycleLSD[1,2]:CycleLSD[1,3],CycleLSD[2,2]:CycleLSD[2,3],CycleLSD[3,2]:CycleLSD[3,3],CycleLSD[4
CycleL5SD=c(rep("A",length(CycleLSD[1,2]:CycleLSD[1,3])),rep("B",length(CycleLSD[2,2]:CycleLSD[2,3])),re
CutL10SD=c(CycleLSD[8,2]:CycleLSD[8,3],CycleLSD[9,2]:CycleLSD[9,3],CycleLSD[10,2]:CycleLSD[10,3],CycleLS
CycleL10SD=c(rep("A",length(CycleLSD[8,2]:CycleLSD[8,3])),rep("B",length(CycleLSD[9,2]:CycleLSD[9,3])),r
CutL15SD=c(CycleLSD[15,2]:CycleLSD[15,3],CycleLSD[16,2]:CycleLSD[16,3],CycleLSD[17,2]:CycleLSD[17,3],Cyc
CycleL15SD=c(rep("A",length(CycleLSD[15,2]:CycleLSD[15,3])),rep("B",length(CycleLSD[16,2]:CycleLSD[16,3]
CutL20SD=c(CycleLSD[22,2]:CycleLSD[22,3],CycleLSD[23,2]:CycleLSD[23,3],CycleLSD[24,2]:CycleLSD[24,3],Cyc
CycleL20SD=c(rep("A",length(CycleLSD[22,2]:CycleLSD[22,3])),rep("B",length(CycleLSD[23,2]:CycleLSD[23,3]
CutL25SD=c(CycleLSD[29,2]:CycleLSD[29,3],CycleLSD[30,2]:CycleLSD[30,3],CycleLSD[31,2]:CycleLSD[31,3],Cyc
CycleL25SD=c(rep("A",length(CycleLSD[29,2]:CycleLSD[29,3])),rep("B",length(CycleLSD[30,2]:CycleLSD[30,3]

CutLSD=TestSD[c(CutL5SD,CutL10SD,CutL15SD,CutL20SD,CutL25SD),]
CutLSD$Cycle=c(CycleL5SD,CycleL10SD,CycleL15SD,CycleL20SD,CycleL25SD)
```

simulateScenario.R:

```
    # Define the biomass cycles ####
    ## Find the maximums and minimums and then get all the cycle's times ----
    CycleXSD2 <- scenario_data[
      by = .(Temperature),
      j = .(
        Max = findPeaks(get(col_name)),
        # Set minimum whilst selecting the correct correction for L or G using a switch
        Min = switch(col_name, "L" = findValleys(L)[seq(2,14,2)], "G" = c(findValleys(G), 2555))
      )
    ] %>%
      # Create new column 'Indices' with sequences of all the times in the cycles
      "$<-"(Indices, apply(., 1, function(cur_row) seq(cur_row[[2]], cur_row[[3]])))

    # For each temperature, get the list with indices and cycle identifiers
    createPerTempLists <- function(ind_lists) {
      # Returns a named list containing INDICES and IDENTIFIERS, both in a single array, for the given
      getIdentifiers <- function(ind_lists) {
        # For each cycle create a list repeating the identifying letter for the length of that cycle
        sapply(1:length(ind_lists), function(i) rep( LETTERS[i], length(ind_lists[[i]]) ))
      }
      return( list(Indices = unlist(ind_lists), Identifiers = unlist(getIdentifiers(ind_lists))) )
    }
    per_temp_lists <- tapply(CycleXSD2$Indices, CycleXSD2$Temperature, createPerTempLists)

    ## Combine indices and identifiers of all temperatures a single vector ----
    all_indices <- sapply(1:length(per_temp_lists), function(i) per_temp_lists[[i]]$Indices + 2555 * (i
    all_identifiers <- sapply(1:length(per_temp_lists), function(i) per_temp_lists[[i]]$Identifiers)

    ## Subset data using the previously created vector ----
    cut_dt <- scenario_data[unlist(all_indices)] %>%
      # Add a column from the vector with identifiers
      "$<-"( Cycle, unlist(all_identifiers) ) %>%
      # Drop the first cycle for each temperature
      "["(Cycle != "A")
```

4 Discussion and Conclusion

One thing that can confidently be said is that the code written by *Réveillon et al.* is not conform to proper coding practises at all and is very difficult to reproducible, especially for people with limited coding skills and knowledge. The code is very convoluted and hard to follow with most of the scenarios being practically identical copies of each other, meaning a change or fix would need to be made in multiple locations instead of in a single spot. Manually changing variable names and values for each scenario like this can also cause accidental mistakes. During the process of rewriting multiple mistakes or oversights have been found that can really affect the results and credibility of the consumer-resource modelling research.

The formulas for the metabolic- and ingestion rates have been heavily rewritten so they resemble the actual equation more and can be better understood. Doing so revealed what seems to be an incorrect implementation of the equation: whilst mathematically simplifying the equation, the simulation temperature should be subtracted by the reference temperature but it seems to have been swapped around with now the reference temperature being subtracted by the simulation temperature instead.

When cleaning the data for calculating the means, standard deviation and persistence of the biomasses a mistake was made. The first 16 days of each year are when the leaf litter falls, these should be removed since they add noise to the data. The exclusion of these days from the data was not correctly implemented, they likely made an oversight in that the code behaved differently from what they expected it to do. This resulted to the removal of the first 16 days of each year only happening for the temperature of 5 degrees Celsius, instead of for every year and every temperature. Since dataframe manipulation is part of the core of the code rewrite, using `data.table`'s vectorization and grouping the data by year *and* temperature with it the leaf litter fall days were now successfully removed, fixing the problem completely.

To calculate the decreasing slopes, the `quantmod` library's `findPeaks` and `findValleys` functions were used. These however have a known flaw in them, causing the found peaks and valleys to always overshoot by 1 row/datapoint, to fix this every indice was subtracted by 1.

In the end the goal of reproducing and improving on the research done by *Réveillon et al.* on the modelling of the consumer-resource dynamics has been achieved. Multiple mistakes were found, with some of them being fixed causing the results to be more accurate and some of them needing more research to be confident in what needs to be done. Implementing new scenarios to expand the research can now be easily done by using the created functions and verifying the validity of the results can now also be better done. For future the research into the problems and mistakes that have been found but could not be researched enough to be confidently corrected can be continued.

5 References

[1] T. P. Dawson, S. T. Jackson, J. I. House, I. C. Prentice, and G. M. Mace, "Beyond predictions: Biodiversity conservation in a changing climate," *Science*, vol. 332, no. 6025, pp. 53–58, 2011, doi: 10.1126/science.1200303.

[2] G.-R. Walther *et al.*, "Ecological responses to recent climate change," *Nature*, vol. 416, no. 6879, pp. 389–395, Mar. 2002, doi: 10.1038/416389a.

[3] A. I. Dell, S. Pawar, and V. M. Savage, "Systematic variation in the temperature dependence of physiological and ecological traits," *Proc Natl Acad Sci U S A*, vol. 108, no. 26, pp. 10591–10596, May 2011.

[4] U. Sommer, R. Adrian, B. Bauer, and M. Winder, "The response of temperate aquatic ecosystems to global warming: Novel insights from a multidisciplinary project," *Marine Biology*, vol. 159, no. 11, pp. 2367–2377, Nov. 2012, doi: 10.1007/s00227-012-2085-4.

[5] G. Woodward, D. M. Perkins, and L. E. Brown, "Climate change and freshwater ecosystems: Impacts across multiple levels of organization," *Philosophical Transactions of the Royal Society B: Biological Sciences*, vol. 365, no. 1549, pp. 2093–2106, 2010, doi: 10.1098/rstb.2010.0055.

[6] J. H. Brown, J. F. Gillooly, A. P. Allen, V. M. Savage, and G. B. West, "Toward a metabolic theory of ecology," *Ecology*, vol. 85, no. 7, pp. 1771–1789, 2004, doi: 10.1890/03-9000.

[7] M. E. Dillon, G. Wang, and R. B. Huey, "Global metabolic impacts of recent climate warming," *Nature*, vol. 467, no. 7316, pp. 704–706, Oct. 2010, doi: 10.1038/nature09407.

[8] D. J. Marshall and C. D. McQuaid, "Warming reduces metabolic rate in marine snails: Adaptation to fluctuating high temperatures challenges the metabolic theory of ecology," *Proceedings of the Royal Society B: Biological Sciences*, vol. 278, no. 1703, pp. 281–288, 2011, doi: 10.1098/rspb.2010.1414.

[9] B. C. Rall *et al.*, "Universal temperature and body-mass scaling of feeding rates," *Philosophical Transactions of the Royal Society B: Biological Sciences*, vol. 367, no. 1605, pp. 2923–2934, 2012, doi: 10.1098/rstb.2012.0242.

[10] O. Vucic-Pestic, R. B. Ehnes, B. C. Rall, and U. Brose, "Warming up the system: Higher predator feeding rates but lower energetic efficiencies," *Global Change Biology*, vol. 17, no. 3, pp. 1301–1310, 2011, doi: 10.1111/j.1365-2486.2010.02329.x.

[11] D. P. Bickford, J. A. Sheridan, and S. D. Howard, "Climate change responses: Forgetting frogs, ferns and flies?" *Trends in Ecology & Evolution*, vol. 26, no. 11, pp. 553–554, 2011, doi: 10.1016/j.tree.2011.06.016.

[12] M. J. Angilletta, P. H. Niewiarowski, and C. A. Navas, "The evolution of thermal physiology in ectotherms," *Journal of Thermal Biology*, vol. 27, no. 4, pp. 249–268, 2002, doi: https://doi.org/10.1016/S0306-4565(01)00094-8.

[13] C. A. Deutsch *et al.*, "Impacts of climate warming on terrestrial ectotherms across latitude," *Proceedings of the National Academy of Sciences*, vol. 105, no. 18, pp. 6668–6672, 2008, doi: 10.1073/pnas.0709472105.

[14] J. F. Gillooly, J. H. Brown, G. B. West, V. M. Savage, and E. L. Charnov, "Effects of size and temperature on metabolic rate," *Science*, vol. 293, no. 5538, pp. 2248–2251, 2001, doi: 10.1126/science.1061967.

[15] G. B. West, V. M. Savage, J. Gillooly, B. J. Enquist, W. H. Woodruff, and J. H. Brown, "Why does metabolic rate scale with body size?" *Nature*, vol. 421, no. 6924, pp. 713–713, Feb. 2003, doi: 10.1038/421713a.

[16] A. P. ALLEN, J. F. GILLOOLY, and J. H. BROWN, "Linking the global carbon cycle to individual metabolism," *Functional Ecology*, vol. 19, no. 2, pp. 202–213, 2005, doi: 10.1111/j.1365-2435.2005.00952.x.

[17] A. Sentis, J.-L. Hemptinne, and J. Brodeur, "Using functional response modeling to investigate the effect of temperature on predator feeding rate and energetic efficiency," *Oecologia*, vol. 169, no. 4, pp. 1117–1125, Aug. 2012, doi: 10.1007/s00442-012-2255-6.

[18] N. P. Lemoine, D. E. Burkepile, and J. D. Parker, "Variable effects of temperature on insect herbivory," *PeerJ*, vol. 2, p. e376, May 2014, doi: 10.7717/peerj.376.

[19] A. Bideault *et al.*, "Thermal mismatches in biological rates determine trophic control and biomass distribution under warming," *Global Change Biology*, vol. 27, no. 2, pp. 257–269, 2021, doi: 10.1111/gcb.15395.

[20] L. Boyero *et al.*, "Global patterns of stream detritivore distribution: Implications for biodiversity loss in changing climates," *Global Ecology and Biogeography*, vol. 21, no. 2, pp. 134–141, 2012, doi: 10.1111/j.1466-8238.2011.00673.x.

[21] B. Wenisch, D. G. Fernández, E. Szöcs, B. G. Mckie, and R. B. Schäfer, "Does the loss of climate sensitive detritivore species alter leaf decomposition?" *Aquatic Sciences*, vol. 79, no. 4, pp. 869–879, Oct. 2017, doi: 10.1007/s00027-017-0538-z.

[22] J. R. Bernhardt, J. M. Sunday, and M. I. O'Connor, "Metabolic theory and the temperature-size rule explain the temperature dependence of population carrying capacity," *The American Naturalist*, vol. 192, no. 6, pp. 687–697, 2018, doi: 10.1086/700114.

[23] M. M. Osmond, M. A. Barbour, J. R. Bernhardt, M. W. Pennell, J. M. Sunday, and M. I. O'Connor, "Warming-induced changes to body size stabilize consumer-resource dynamics," *The American Naturalist*, vol. 189, no. 6, pp. 718–725, 2017, doi: 10.1086/691387.

[24] A. Sentis, A. Binzer, and D. S. Boukal, "Temperature-size responses alter food chain persistence across environmental gradients," *Ecology Letters*, vol. 20, no. 7, pp. 852–862, 2017, doi: 10.1111/ele.12779.

[25] J. Jabiol *et al.*, "Variable temperature effects between heterotrophic stream processes and organisms," *Freshwater Biology*, vol. 65, no. 9, pp. 1543–1554, 2020, doi: 10.1111/fwb.13520.

[26] B. O. L. Demars *et al.*, "Temperature and the metabolic balance of streams," *Freshwater Biology*, vol. 56, no. 6, pp. 1106–1121, 2011, doi: 10.1111/j.1365-2427.2010.02554.x.

[27] G. Yvon-Durocher, J. I. Jones, M. Trimmer, G. Woodward, and J. M. Montoya, "Warming alters the metabolic balance of ecosystems," *Philosophical Transactions of the Royal Society B: Biological Sciences*, vol. 365, no. 1549, pp. 2117–2126, 2010, doi: 10.1098/rstb.2010.0038.

[28] T. Réveillon, T. Rota, É. Chauvet, A. Lecerf, and A. Sentis, "Energetic mismatch induced by warming decreases leaf litter decomposition by aquatic detritivores," *Journal of Animal Ecology*, vol. n/a, no. n/a, 2022, doi: 10.1111/1365-2656.13710.

[29] G. Englund, G. Öhlund, C. L. Hein, and S. Diehl, "Temperature dependence of the functional response," *Ecology Letters*, vol. 14, no. 9, pp. 914–921, 2011, doi: 10.1111/j.1461-0248.2011.01661.x.

[30] U. M. Irlich, J. S. Terblanche, T. M. Blackburn, and S. L. Chown, "Insect rate-temperature relationships: Environmental variation and the metabolic theory of ecology." *The American Naturalist*, vol. 174, no. 6, pp. 819–835, 2009, doi: 10.1086/647904.

[31] B. Lang, R. B. Ehnes, U. Brose, and B. C. Rall, "Temperature and consumer type dependencies of energy flows in natural communities," *Oikos*, vol. 126, no. 12, pp. 1717–1725, 2017, doi: 10.1111/oik.04419.

[32] T. Schneider *et al.*, "Who is who in litter decomposition? Metaproteomics reveals major microbial players and their biogeochemical functions," *The ISME Journal*, vol. 6, no. 9, pp. 1749–1762, Sep. 2012, doi: 10.1038/ismej.2012.11.

[33] R Core Team, *R: A language and environment for statistical computing*. Vienna, Austria: R Foundation for Statistical Computing, 2022. Available: https://www.R-project.org/

[34] K. Soetaert, T. Petzoldt, and R. W. Setzer, "Solving differential equations in R: Package deSolve," *Journal of Statistical Software*, vol. 33, no. 9, pp. 1–25, 2010, doi: 10.18637/jss.v033.i09.

[35] M. Dowle and A. Srinivasan, *Data.table: Extension of 'data.frame'*. 2021. Available: https://CRAN.R-project.org/package=data.table

[36] A. Kassambara, *Ggpubr: 'ggplot2' based publication ready plots*. 2020. Available: https://CRAN.R-project.org/package=ggpubr

[37] H. Wickham, "Reshaping data with the reshape package," *Journal of Statistical Software*, vol. 21, no. 12, pp. 1–20, 2007, Available: http://www.jstatsoft.org/v21/i12/

[38] H. Wickham *et al.*, "Welcome to the tidyverse," *Journal of Open Source Software*, vol. 4, no. 43, p. 1686, 2019, doi: 10.21105/joss.01686.

[39] D. Bates, M. Mächler, B. Bolker, and S. Walker, "Fitting linear mixed-effects models using lme4," *Journal of Statistical Software*, vol. 67, no. 1, pp. 1–48, 2015, doi: 10.18637/jss.v067.i01.

[40] J. A. Ryan and J. M. Ulrich, *Quantmod: Quantitative financial modelling framework*. 2022. Available: https://CRAN.R-project.org/package=quantmod

[41] J. J. Follstad Shah *et al.*, "Global synthesis of the temperature sensitivity of leaf litter breakdown in streams and rivers," *Glob. Chang. Biol.*, vol. 23, no. 8, pp. 3064–3075, Aug. 2017.

[42] D. Atkinson, "Temperature and organism size—a biological law for ectotherms?" M. Begon and A. H. Fitter, Eds., in Advances in ecological research, vol. 25. Academic Press, 1994, pp. 1–58. doi: 10.1016/S0065-2504(08)60212-3.

[43] J. Forster, A. G. Hirst, and D. Atkinson, "Warming-induced reductions in body size are greater in aquatic than terrestrial species," *Proceedings of the National Academy of Sciences*, vol. 109, no. 47, pp. 19310–19314, 2012, doi: 10.1073/pnas.1210460109.

6 Appendices

Appendix A: 'model.R'

```r
## Copyright (c) 2023 Vincent Talen.
## Licensed under GPLv3. See LICENSE file.
## ~~~~~~~~~~~~~~~~~~~~~~~~~~
##
## Script name: model.R
##
## Purpose of script: Implements the biological model of consumer-resource dynamics
##
## Author: Vincent Talen
##
## Date Created: 09 Jan 2023
##
## Email: v.k.talen@st.hanze.nl
##
## ~~~~~~~~~~~~~~~~~~~~~~~~~~
##
## Notes:
##   - Goal: formulate formula functions in a more readable/recognizable manner
##
## ~~~~~~~~~~~~~~~~~~~~~~~~~~


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


# ######### #
#   Code    #
# ######### #
# Define Boltzmann term (°K)
boltz_const <- 8.62 * 10^-5
# Define mean inverse temperature (calculated from Data_Mismatch.txt)
mean_inverse_temp <- 40.5941593143742


## ---- mte formulations ----
# 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                # 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
```

```r

  # 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                # 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(T.C) {
  alpha <- exp(-0.84730)    # normalization constant of assimilation efficiency
  Ea <- 0.16400             # activation energy
  T.0 <- 285.65             # reference temperature of 12.5 degrees Celsius in Kelvin
  T.K <- T.C + 273.15       # convert temperature from Celsius to Kelvin

  mte_equation <- alpha * exp( Ea * (T.K - T.0) / (boltz_const * T.0 * T.K) )
  return(mte_equation / (1 + mte_equation))
}


## ---- attack rate parameter ----
# Attack rate function based on exponential decay (quadratic model)
calcAttackRate <- function(temp, mass) {
  t <- 2         # Experiment duration
  M.L <- 4612.5  # Mean initial C mass of leaf discs in microcosms (μg C)
  decay_rate <- -log(1 - calcIngestionRate(temp, mass) * t / M.L)
  return( decay_rate / t )
}


## ---- 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 ----
# Function for the leaf litter microbial decomposition rate (Arrhenius equation)
calcLeafDecomp <- function(T.C) {
```

```r
  k.10C <- 0.00956          # litter decomposition rate at 10°C (283.15°K)
  Ea <- 0.37000             # activation energy
  T.K <- T.C + 273.15       # convert temperature from Celsius to Kelvin
  # return( k.10C * exp(-Ea * (1 / (boltz_const * T.K - 283.15 * boltz_const))) )
  return( k.10C * exp(-Ea * (1 / (boltz_const * T.K) - 10)) )
}


## ---- consumer-resource model ----
GammLeafModel <- function(temp, gamm_indv_mass, leaf_fall, gamm_start_biomass, tsr_model) {
  # Apply TSR Model to Gammarus individual mass
  if (!is.null(tsr_model)) {
    gamm_indv_mass <- tsr_model(temp, gamm_indv_mass)
  }

  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)
  )

  # 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
  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
```

```r
  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
## Copyright (c) 2023 Vincent Talen.
## Licensed under GPLv3. See LICENSE file.
## ~~~~~~~~~~~~~~~~~~~~~~~~~~
##
## Script name: functions.R
##
## Purpose of script: Functions
##
## Author: Vincent Talen
##
## Date Created: 09 Jan 2023
##
## Email: v.k.talen@st.hanze.nl
##
## ~~~~~~~~~~~~~~~~~~~~~~~~~~
##
## Notes:
##   - x
##
```

```r
## ~~~~~~~~~~~~~~~~~~~~~~~~~~

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


# ######### #
# Functions #
# ######### #
# ---- Scenario data gathering and preparations ----
getScenarioDataList <- function(gamm_indv_mass, leaf_fall, gamm_start_biomass, tsr_model) {
  # 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, tsr_m
    setNames(temperatures)
  return(data_list)
}

createLongDataFrame <- function(df_list, tsr_model) {
  if (is.null(tsr_model)) { tsr_model <- function(temp, mass) {return(mass)} }

  # 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, tsr_model(cur_temp, gamm_indv_mass)) / 1000  # Gammarus me
    # 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, tsr_model(cur_temp, gamm_indv_mass)) / 1000  # Gammarus
    attack_rate <- calcAttackRate(cur_temp, tsr_model(cur_temp, gamm_indv_mass))           # Gammarus
    # Calculate population leaf ingestion
    pop_ingestion <- (attack_rate * leaf_biomass / (1 + attack_rate * 1 / ingest_rate * leaf_biomass)) *
    return(fifelse(pop_ingestion < 0, 0, pop_ingestion))
  }

  # Bind all dataframes from list to single big one and
  # drop the 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
    "$<-"(Temperature, rep(temperatures, each = 2555)) %>%
    "$<-"(Year, rep(rep(1:7, each = 365), 5)) %>%
```

```
    # Add population metabolism and leaf ingestion columns
    "$<-"(M, getPopMetabolism(.$Temperature, .$G)) %>%
    "$<-"(I, getPopIngestion(.$Temperature, .$L, .$G)) %>%
    # Set column order to a nicer one
    setcolorder(c("Time", "L", "G", "M", "I", "Temperature", "Year"))
  return(big_df)
}

# ---- Plot 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), # offset title inside the plot
      plot.margin=margin(0.1, 0.25, 0, 0, "cm"), # enlarge plots and decrease gaps between them
      legend.text=element_text(size=12), # for better readability increase legend font size
      legend.key.width=unit(1,"cm") # widen the legend keys
    ) +
    scale_color_manual(
      name = "",
      values = c("black","tomato2"),
      limits = c("Leaf Litter Biomass", "Gammarus Fossarum Biomass")
    ) +
    guides(colour=guide_legend(override.aes=list(size=1))) # Increase legend entries' line width
  return(plot)
}

plotScenarioDynamics <- 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, size=16, face="bold"),
      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(paste("figures/reproduced_plots/", file_out, sep=""), bg = "white", width=15, height=8, units=
  dev.off()
}

# ---- Create prediction data ----
createPredictionData <- function(huidige_data, scenario_names) {
  # Get vector from lowest temperature to highest temperature with steps of 0.5
  Temperature <- seq(head(temperatures, n=1), tail(temperatures, n=1), by=0.5)
  # Create vector with scenario names times the number of temperature steps
  Scenario <- rep(scenario_names, each=length(Temperature))
  # Create dataframe
  Pred <- data.table(Temperature, Scenario)

  # Each column with 'Mean' should get their predicted values for smooth lines
  for(cname in colnames(huidige_data)) {
    if (startsWith(cname, "Mean")) {
      new_col_name <- gsub("Mean", "Pred", cname)
      predicted_data <- predict(lmList(get(cname) ~ poly(Temperature, 2)|Scenario, data=huidige_data), I
      Pred[ , (new_col_name) := predicted_data]
    }
  }
  return(Pred)
}
```

Appendix C: 'simulateScenario.R'

```
## Copyright (c) 2023 Vincent Talen.
## Licensed under GPLv3. See LICENSE file.
## ~~~~~~~~~~~~~~~~~~~~~~~~~~
##
## Script name: simulateScenario.R
##
## Purpose of script: Make simulating a scenario easy with this one function
##
## Author: Vincent Talen
##
## Date Created: 29 Jun 2023
##
## Email: v.k.talen@st.hanze.nl
##
## ~~~~~~~~~~~~~~~~~~~~~~~~~~
##
## Notes:
##   - x
##
## ~~~~~~~~~~~~~~~~~~~~~~~~~~
```

```r


# ######### #
#   Libs    #
# ######### #
library(quantmod)
library(lme4)


# ######### #
#   Code    #
# ######### #
simulateScenario <- function(scenario_data, scenario_name) {
  simulateColumnScenario <- function(scenario_data, scenario_name, col_name, threshold) {
    # Exclude the first year and the first 16 days with leaf fall for all years for each temperature
    cut_df <- scenario_data[!Year == "1", tail(.SD, -16), by = .(Year)] %>%
      setcolorder(colnames(scenario_data)) # Restore column order

    # Biomass mean for each temperature ####
    Biom <- cut_df %>%
      # Calculate mean annual biomass per temperature, per year
      "["(j = .(all_means = mean(get(col_name))), by = .(Temperature, Year)) %>%
      # Calculate biomass mean and deviation over 6 years per temperature
      "["(j = .(MeanBiom = mean(all_means), SdBiom = sd(all_means)), by = Temperature)

    # Calculate persistence time for each temperature with the threshold ####
    PersTime <- cut_df %>%
      # Only keep days above the threshold
      "["(i = .[[col_name]] > threshold) %>%
      # Count the days above the threshold per temperature, per year
      "["(j = .(days_above = .N), by = .(Temperature, Year)) %>%
      # Calculate persistence time mean and deviation over 6 years per temperature
      "["(j = .(MeanPersTime = mean(days_above), SdPersTime = sd(days_above)), by = Temperature)

    # Define the biomass cycles ####
    ## Find the maximums and minimums and then get all the cycle's times ----
    CycleXSD2 <- scenario_data[
      by = .(Temperature),
      j = .(
        Max = findPeaks(get(col_name)),#-1,
        # Set minimum whilst selecting the correct correction for L or G using a switch
        #Min = switch(col_name, "L" = findValleys(L)[seq(2,14,2)]-1, "G" = c(findValleys(G)-1, 2555))
        Min = switch(col_name, "L" = findValleys(L)[seq(2,14,2)], "G" = c(findValleys(G), 2555))
      )
    ] %>%
      # Create new column 'Indices' with sequences of all the times in the cycles
      "$<-"(Indices, apply(., 1, function(cur_row) seq(cur_row[[2]], cur_row[[3]])))

    # For each temperature, get the list with indices and cycle identifiers
    createPerTempLists <- function(ind_lists) {
      # Returns a named list containing INDICES and IDENTIFIERS, both in a single array, for the given
      getIdentifiers <- function(ind_lists) {
        # For each cycle create a list repeating the identifying letter for the length of that cycle
        sapply(1:length(ind_lists), function(i) rep( LETTERS[i], length(ind_lists[[i]]) ))
```

```
      }
      return( list(Indices = unlist(ind_lists), Identifiers = unlist(getIdentifiers(ind_lists))) )
    }
    per_temp_lists <- tapply(CycleXSD2$Indices, CycleXSD2$Temperature, createPerTempLists)

    ## Combine indices and identifiers of all temperatures a single vector ----
    all_indices <- sapply(1:length(per_temp_lists), function(i) per_temp_lists[[i]]$Indices + 2555 * (i
    all_identifiers <- sapply(1:length(per_temp_lists), function(i) per_temp_lists[[i]]$Identifiers)

    ## Subset data using the previously created vector ----
    cut_dt <- scenario_data[unlist(all_indices)] %>%
      # Add a column from the vector with identifiers
      "$<-"( Cycle, unlist(all_identifiers) ) %>%
      # Drop the first cycle for each temperature
      "["(Cycle != "A")

    # Calculate litter and Gammarus mean maximum biomasses and decreasing slopes ----
    ## Get the first time each cycle per temperature where the biomass is below the threshold
    first_below_threshold <- cut_dt[get(col_name) < threshold, head(.SD, 1), by = .(Temperature, Cycle)]
    ## Also get the highest biomqss (first) value for each cycle per temperature
    maximum_values <- cut_dt[, head(.SD, 1), by = .(Temperature, Cycle)]

    ## Calculate the slope for each cycle using the maximum and threshold times
    SlopLSD3 <- rbind(first_below_threshold, maximum_values) %>%
      "["( j = .(Slope = coef(lm(get(col_name) ~ Time))[[2]]), by = .(Cycle, Temperature) )

    Maxi <- maximum_values[, .(MeanMax = mean(get(col_name)), SdMax = sd(get(col_name))), by = Temperatu
    Slop <- SlopLSD3[, .(MeanSlope = mean(Slope), SdSlope = sd(Slope)), by = Temperature]

    # Put all the data tables into a single one
    final_df <- setDT(c(Biom, PersTime, Maxi, Slop)) %>%
      # Remove duplicate 'Temperature' columns
      "["(j = which(duplicated(names(.))) := NULL)
    # Add L or G (current column argument) to column names
    colnames(final_df)[2:9] <- sapply(colnames(final_df)[2:9], function(cur_col) paste(cur_col, col_name
    return(final_df)
  }

  # Use above function to get data
  Leaf <- simulateColumnScenario(scenario_data, scenario_name, "L", 60000)
  Gamm <- simulateColumnScenario(scenario_data, scenario_name, "G", 5000)

  # Put all the information in a single data.table
  combined_df <- setDT(c(Leaf, Gamm)) %>%
    # Remove duplicate 'Temperature' columns
    "["(j = which(duplicated(names(.))) := NULL)

  # Add scenario column
  combined_df <- add_column(combined_df, Scenario=rep(scenario_name, 5), .after="Temperature")
  return(combined_df)
}
```

Appendix D: 'mainAnalysis.R'

```r
## Copyright (c) 2023 Vincent Talen.
## Licensed under GPLv3. See LICENSE file.
## ~~~~~~~~~~~~~~~~~~~~~~~~~~
##
## Script name: mainAnalysis.R
##
## Purpose of script: Perform all scenario simulations for the main analysis
##
## Author: Vincent Talen
##
## Date Created: 29 Jun 2023
##
## Email: v.k.talen@st.hanze.nl
##
## ~~~~~~~~~~~~~~~~~~~~~~~~~~
##
## Notes:
##   - x
##
## ~~~~~~~~~~~~~~~~~~~~~~~~~~


# ######### #
#   Libs    #
# ######### #
source("src/functions.R")
source("src/simulateScenario.R")


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

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

# Gammarus mean body mass = 4.26 mgDM
gamm_indv_mass <- 4.26
# Annual leaf fall = 300 gC/m2/an = 300 000 mgC/m2/an
leaf_fall <- 300000 / fall_duration_in_days
# Gammarus density = 30 mgDM/m2 = 15 mgC/m2
gamm_start_biomass <- 15


# SCENARIO 0: REFERENCE SCENARIO ############################################
# Get data for temperatures with values of current scenario
df_list.TSRR <- getScenarioDataList(gamm_indv_mass, leaf_fall, gamm_start_biomass, NULL)

```

```r
# Create plots in an arranged grid
file_out.TSRR <- "Population Dynamics Reference Scenario.png"
image_title.TSRR <- "Reference Scenario: Population Dynamics over 7 years"
plotScenarioDynamics(df_list.TSRR, image_title.TSRR, file_out.TSRR)

# Combine dataframes into one and add temperature, year, population metabolism- and ingestion columns
combined_df.TSRR <- createLongDataFrame(df_list.TSRR, NULL)
# Simulate scenario and get final dataframes for both types of masses
data.TSRR <- simulateScenario(combined_df.TSRR, "TSRR")


# SCENARIO 1: AVERAGE TSR RESPONSE ########################################
# Get data for temperatures with values of current scenario
df_list.TSRA <- getScenarioDataList(gamm_indv_mass, leaf_fall, gamm_start_biomass, calcTSR.Avg)

# Create plots in an arranged grid
file_out.TSRA <- "Population Dynamics Average TSR Scenario.png"
image_title.TSRA <- "Average Temperature-Size Rule Response: Population Dynamics over 7 years"
plotScenarioDynamics(df_list.TSRA, image_title.TSRA, file_out.TSRA)

# Combine dataframes into one and add temperature, year, population metabolism- and ingestion columns
combined_df.TSRA <- createLongDataFrame(df_list.TSRA, calcTSR.Avg)
# Simulate scenario and get final dataframes for both types of masses
data.TSRA <- simulateScenario(combined_df.TSRA, "TSRA")


# SCENARIO 2: MAXIMUM TSR RESPONSE ########################################
# Get data for temperatures with values of current scenario
df_list.TSRM <- getScenarioDataList(gamm_indv_mass, leaf_fall, gamm_start_biomass, calcTSR.Max)

# Create plots in an arranged grid
file_out.TSRM <- "Population Dynamics Maximum TSR Scenario.png"
image_title.TSRM <- "Maximum Temperature-Size Rule Response: Population Dynamics over 7 years"
plotScenarioDynamics(df_list.TSRM, image_title.TSRM, file_out.TSRM)

# Combine dataframes into one and add temperature, year, population metabolism- and ingestion columns
combined_df.TSRM <- createLongDataFrame(df_list.TSRM, calcTSR.Max)
# Simulate scenario and get final dataframes for both types of masses
data.TSRM <- simulateScenario(combined_df.TSRM, "TSRM")


## PLOTTING FUNCTIONS ##########################################################
createPlotForStateVariable <- function(state_variable_name, statistic_info, data_column_names, all_scena
  # Calculate the y-axis limits
  max_lim <- round( max(all_scenario_data[,get(data_column_names["Mean"])]) + statistic_info$correction
  break_by <- ifelse(max_lim <= 4, 1, 2)

  plot <- ggplot(all_scenario_data, aes(x=Temperature, y=abs(!!sym(data_column_names["Mean"])), group=Sc
    # Add the actual scenario simulation points to the plot
    geom_point(
      aes(color=Scenario),
      size=3,
      position=position_dodge(0.5)
```

```
    ) +
    # Add the prediction data lines to the plot
    geom_line(
      data=prediction_data,
      aes(x=Temperature, y=abs(!!sym(data_column_names["Pred"])), color=Scenario, linetype=Scenario),
      show.legend=F
    ) +
    # Add the standard deviation error bar to the plot
    geom_errorbar(
      color="grey50",
      width=0.75,
      position=position_dodge(0.5),
      aes(ymin=(abs(!!sym(data_column_names["Mean"]) - !!sym(data_column_names["Sd"]))),
          ymax=(abs(!!sym(data_column_names["Mean"]) + !!sym(data_column_names["Sd"]))))
    ) +
    # Add the plot title and axis labels
    labs(x = "Temperature (\u00B0C)", y=statistic_info$y_label, title=state_variable_name) +
    # Style the plot title, axis and legend texts
    theme(
      axis.text.y=element_text(size=10, colour="black"),
      axis.text.x=element_text(size=10, colour="black"),
      plot.title = element_text(size=12),
      legend.title=element_text(face="bold")
    ) +
    # Set the y-axis limits
    scale_y_continuous(labels=function(x) {x / statistic_info$correction}) +
    expand_limits(y = 0) +
    # Color and types of plotted data elements
    scale_color_manual(values=c(TSRR="black", TSRA="steelblue2", TSRM="tomato2")) +
    scale_linetype_manual(values=c("TSRR"="dotted", "TSRA"="solid", "TSRM"="solid"))
  return(plot)
}

plotGridOfStatistic <- function(statistic_info, all_scenario_data, prediction_data) {
  # Define state variables
  state_variable_list <- list(
    list(id="L", full_name="Leaf Litter"),
    list(id="G", full_name="Gammarus Fossarum")
  )
  # Create the plots for the state variables in a list using lapply
  state_variable_plots <- lapply(state_variable_list, function(state_variable) {
    # Get the full names of the data columns for the current statistic + state variable combination
    data_column_names <- sapply(
      c("Mean", "Sd", "Pred"),  # for each data column
      paste,                    # perform the pasta function
      state_variable$id,        # put the id of the state variable at the end
      sep=statistic_info$name   # and the name of the statistic in between
    )
    # Create the plot of the data columns for the current statistic + state variable combination
    createPlotForStateVariable(state_variable$full_name, statistic_info, data_column_names, all_scenari
  })

  # Return the plots in an arranged grid with the legend at the bottom
```

```r
  arranged_plots <- ggpubr::ggarrange(plotlist=state_variable_plots, ncol=2, nrow=1, common.legend=TRUE
  return(arranged_plots)
}

createFigureForStatistic <- function(statistic_info, all_scenario_data) {
  # Get prediction data that is used for the continuous lines in the plots
  prediction_data <- createPredictionData(all_scenario_data, c("TSRR", "TSRA", "TSRM"))
  # Create the arranged grid with the two plots of the state variables for the statistic and annotate t
  plotGridOfStatistic(statistic_info, all_scenario_data, prediction_data) %>%
    annotate_figure(top=text_grob(statistic_info$image_title, size=16, face="bold"))
  # Save the final figure for the column
  ggsave(
    filename=paste("figures/reproduced_plots/", statistic_info$image_title, ".png", sep=""),
    bg="white",
    width=3840,
    height=2160,
    units="px",
    dpi=300
  )
  dev.off()
}


## CREATING PLOTS ##############################################################
# Combine the data frames into a single one
combined_data <- rbind(data.TSRR, data.TSRA, data.TSRM)

# Create a list with the statistics that a figure needs to be made for
statistic_info_list <- list(
  list(name = "Biom",
       correction = 10^4,
       y_label = expression('Mean biomass'~'('*10^4~mg~C~m^-2*')'),
       image_title = "Mean Biomass over Temperature"),
  list(name = "PersTime",
       correction = 1,
       y_label = "Persistance time (days)",
       image_title = "Persistance Time over Temperature"),
  list(name = "Slope",
       correction = 10^4,
       y_label = expression('Mean biomass slope'~'('*10^4~mg~C~m^-2~day^-1*')'),
       image_title = "Mean Biomass Slope over Temperature")
)

# Create a figures for each statistic defined above
for (statistic_info in statistic_info_list) {
  createFigureForStatistic(statistic_info, combined_data)
}
```