

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	ii
List of Figures	ii
List of Tables	ii
1 Introduction	1
1.1 Theory	2
State variables	2
Parameter equations	3
2 Materials and Methods	5
2.1 The software model	5
2.2 Model configuration	5
Main analysis	7
3 Results	8
3.1 Figures	8
Population dynamics	9
Mean Biomass Slope	10
Persistence Time	11
3.2 R Code	12
Implementation of equations	12
Improvement of simple list creation	13
Defining the cycles of scenario simulation	13
4 Discussion and Conclusion	15
4.1 Discussion	15
4.2 General conclusion and perspective	15
5 References	16
6 Appendices	19
Appendix A: ‘model.R’	19
Appendix B: ‘functions.R’	22
Appendix C: ‘simulateScenario.R’	25
Appendix D: ‘mainAnalysis.R’	28

List of Abbreviations

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

List of Figures

1	Gammarus fossarum	2
2	Diagram showing dynamics of the model (Made by Vincent Talen)	2
3	Population Dynamics SD (made by Réveillon et al.)	9
4	Population Dynamics Reference Scenario (made by Vincent Talen)	9
5	Mean Biomass Slope Scenarios (made by Réveillon et al.)	10
6	Mean Biomass Slope over Temperature (made by Vincent Talen)	10
7	Persistence Time Scenarios (made by Réveillon et al.)	11
8	Persistance Time over Temperature (made by Vincent Talen)	11

List of Tables

1	Model state variables' parameters	3
2	Software and packages	5
3	Global parameter values	6
4	Metabolic Rate (3b) parameter values	6
5	Ingestion Rate parameter values	6
6	Assimilation Efficiency parameter values	6
7	Attack rate formula static parameter values	7
8	Leaf Decomposition Rate parameter values	7
9	Annual persistence time threshold values	7
10	Main analysis initial state variables biomass values	7

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.

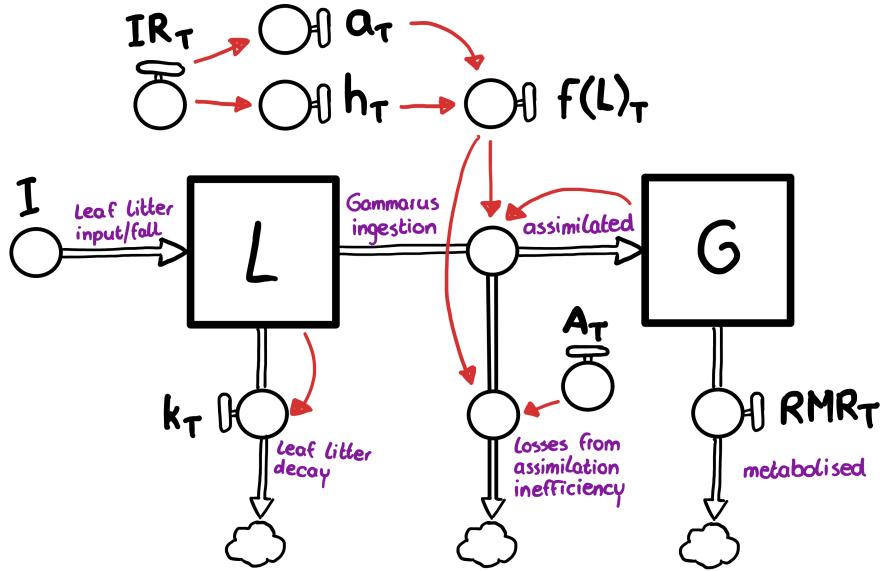


Figure 2: Diagram showing dynamics of the model (Made by Vincent Talen)

State variables

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

$$\frac{dL}{dt} = I - f(L)_T G - k_T L \quad (1a)$$

$$\frac{dG}{dt} = G [f(L)_T A_T - RMR_T] \quad (1b)$$

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
$f(L)_T$	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).

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

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 $DC_T = -\log(1 - IR_T * t/M_L)$. 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 ($a_T = DC_T/t$). The handling time is calculated as the inverse of the ingestion rate ($h_T = 1/IR_T$).

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:

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

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

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.

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

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.

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

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.tables` 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 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	$\mu\text{g C}$	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
$k_{10^\circ\text{C}}$	0.00956	day ⁻¹	Litter decomposition rate at 10°C (283.15°K)
E_a	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	$\text{mg C m}^{-2} \text{year}^{-1}$	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.

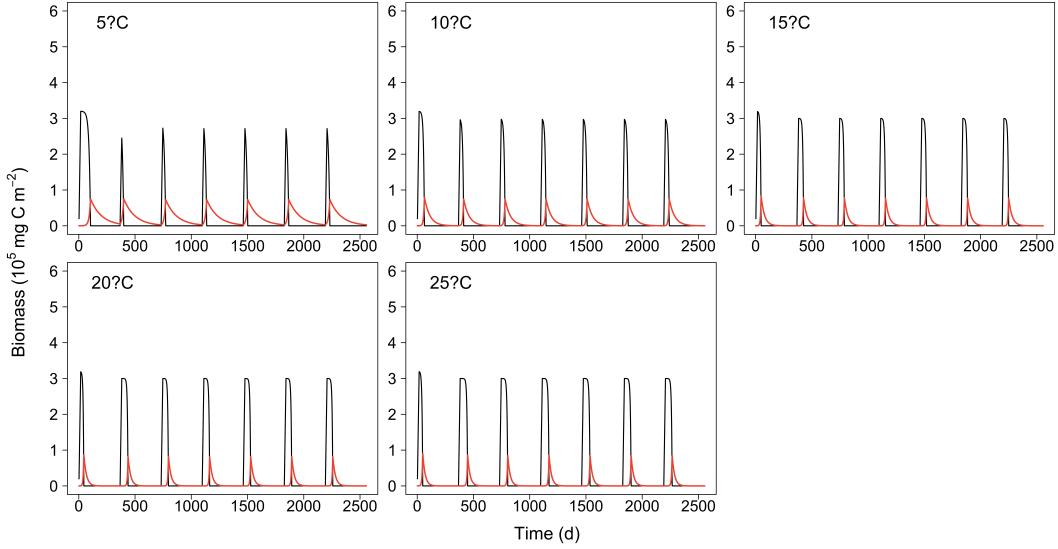


Figure 3: Population Dynamics SD (made by Réveillon et al.)

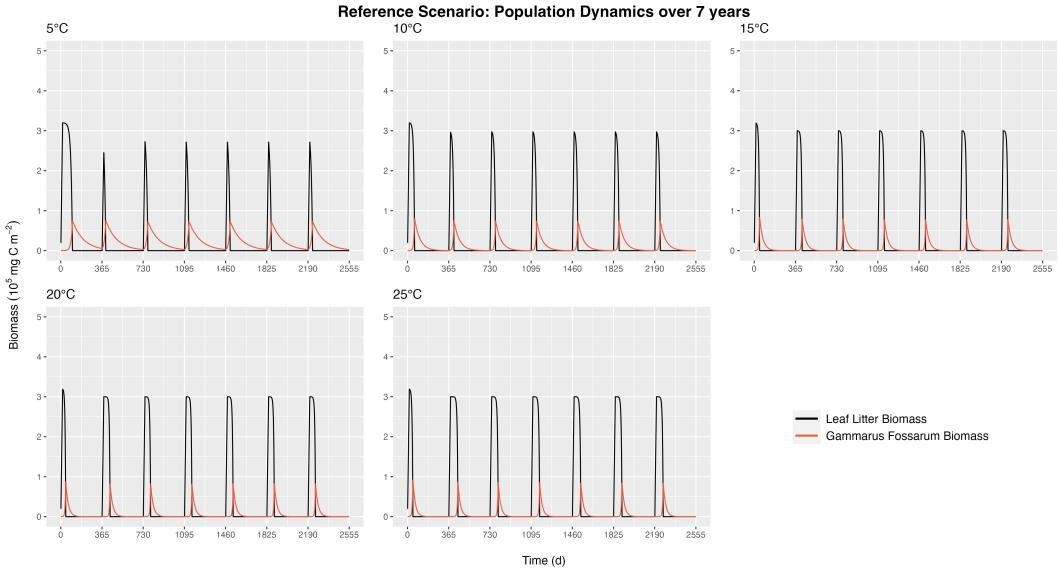


Figure 4: Population Dynamics Reference Scenario (made by Vincent Talen)

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 dependence 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.

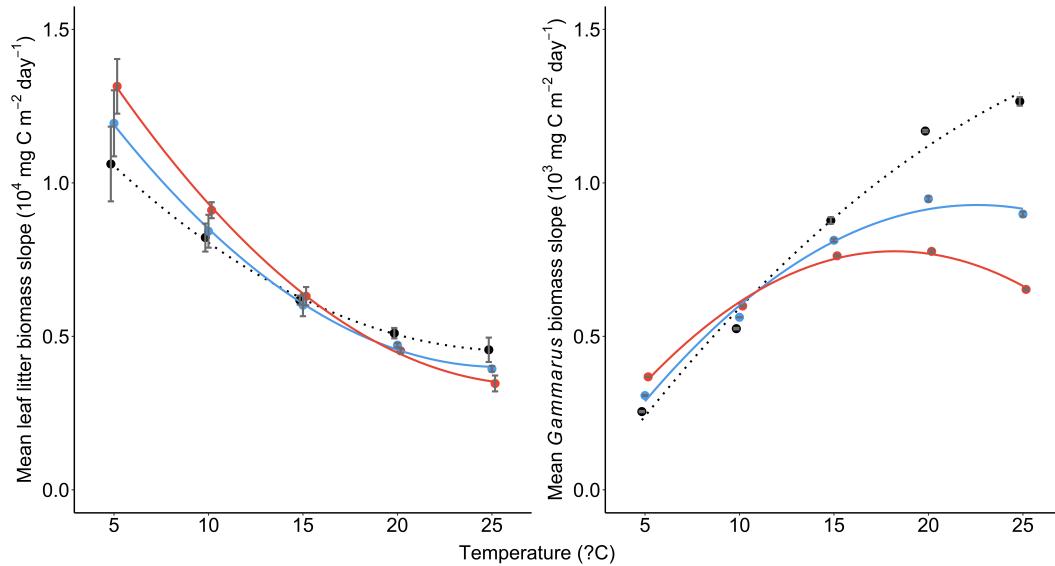


Figure 5: Mean Biomass Slope Scenarios (made by Réveillon et al.)

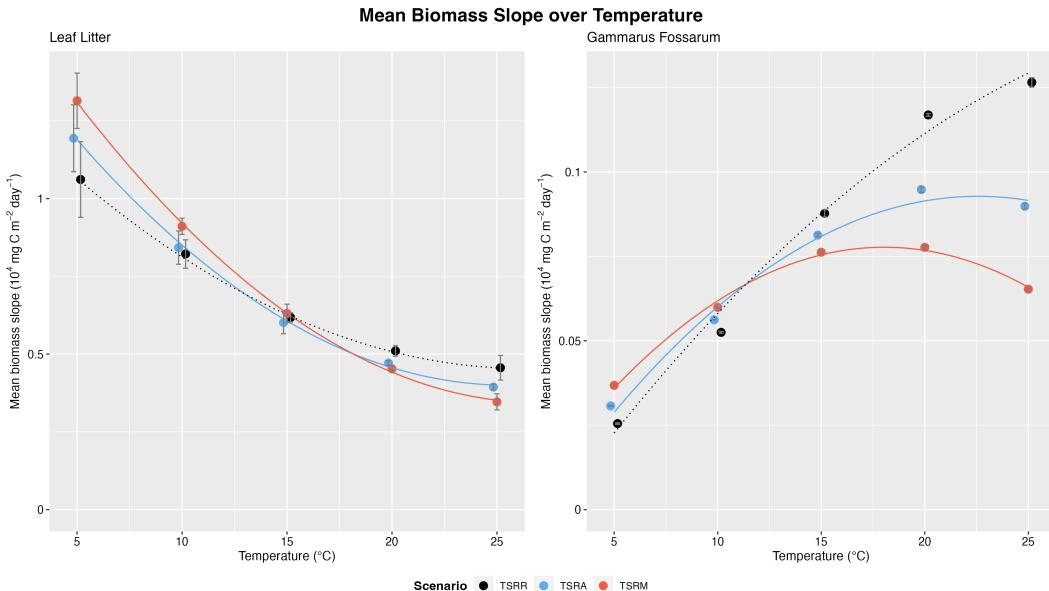


Figure 6: Mean Biomass Slope over Temperature (made by Vincent Talen)

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.

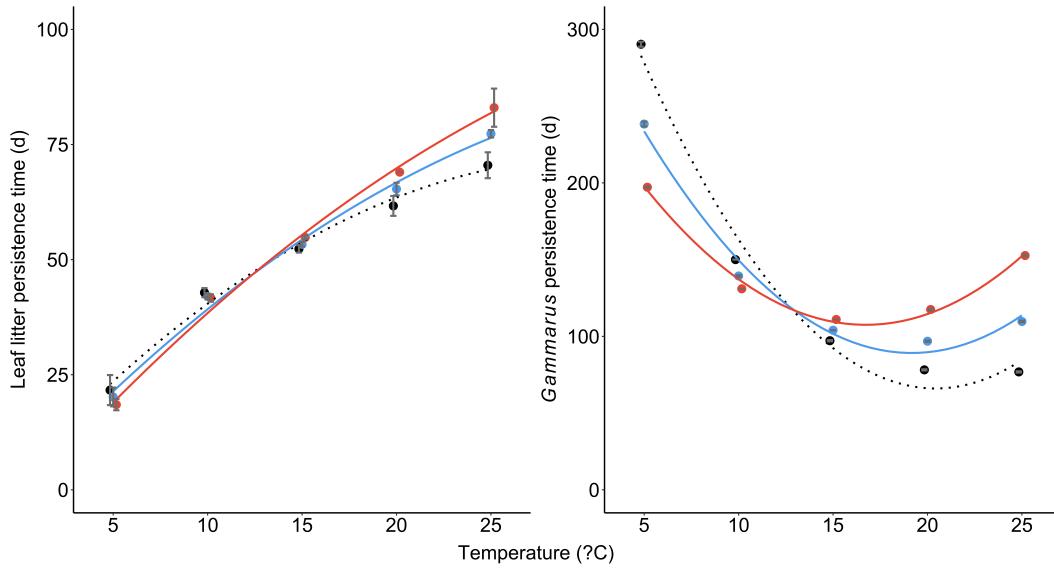


Figure 7: Persistence Time Scenarios (made by Réveillon et al.)

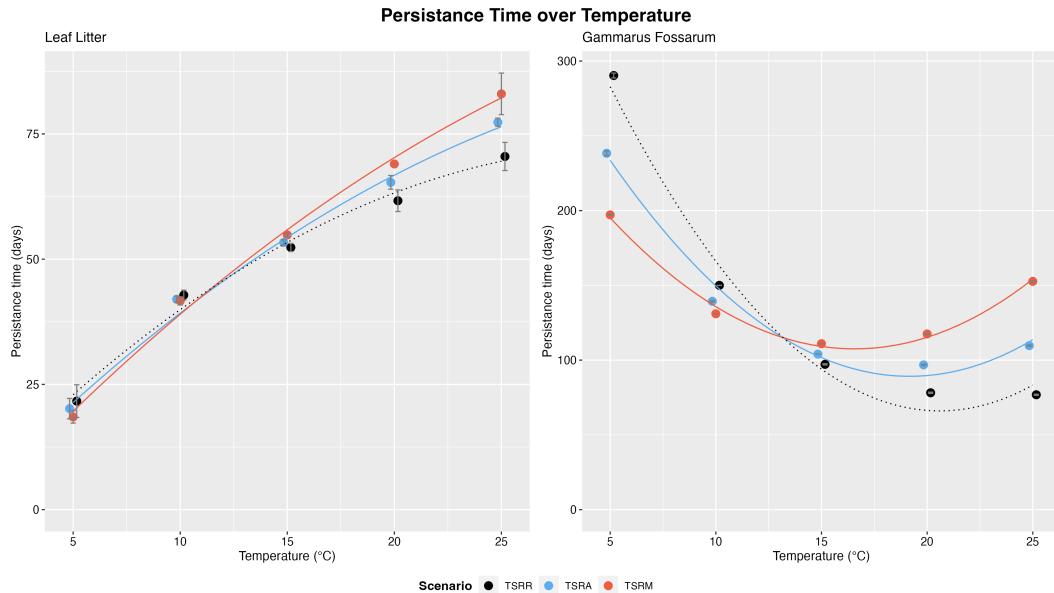


Figure 8: Persistance Time over Temperature (made by Vincent Talen)

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 advised 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.

```
333 AssimQuadra=function(Temp){  
334   (exp(-0.84730)*exp(0.16400*((Temp+273.15)-285.65)/(Boltz*285.65*(Temp+273.15))))  
335   /(1+(exp(-0.84730)*exp(0.16400*((Temp+273.15)-285.65)/(Boltz*285.65*(Temp+273.15)))))}  
  
model.R:  
  
75 calcAssimEff <- function(T.C) {  
76   alpha <- exp(-0.84730)      # normalization constant of assimilation efficiency  
77   Ea <- 0.16400                # activation energy  
78   T.0 <- 285.65                 # reference temperature of 12.5 degrees Celsius in Kelvin  
79   T.K <- T.C + 273.15          # convert temperature from Celsius to Kelvin  
80  
81   mte_equation <- alpha * exp( Ea * (T.K - T.0) / (boltz_const * T.0 * T.K) )  
82   return(mte_equation / (1 + mte_equation))  
83 }
```

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.

```
407 # Define the years
408 y1=365; y2=2*y1; y3=3*y1; y4=4*y1; y5=5*y1; y6=6*y1
409
410 # Time points to trigger litter fall
411 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:
134 # Get time points to trigger litter fall event (first 15 days of the year)
135 getFallTimesYearX <- function(year) { seq(year * 365 + 1, year * 365 + 15) }
136 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.

```
595 # Find biomass maximums and minimums
596 CycleLSD/as.data.frame(setDT(TestSD)[, .(MaxLSD=findPeaks(L), MinLSD=findValleys(L)[seq(2,14,2)]), by=1]
597 CycleGSD/as.data.frame(setDT(TestSD)[, .(MaxGSD=findPeaks(G), MinGSD=c(findValleys(G),2555)), by=list(T
598
599 # Define litter and Gammarus biomass cycles
600 t0=2555*0; t1=2555*1; t2=2555*2; t3=2555*3; t4=2555*4
601
602 CutL5SD=c(CycleLSD[1,2]:CycleLSD[1,3],CycleLSD[2,2]:CycleLSD[2,3],CycleLSD[3,2]:CycleLSD[3,3],CycleLSD[4,
603 CycleL5SD=c(rep("A",length(CycleLSD[1,2]:CycleLSD[1,3])),rep("B",length(CycleLSD[2,2]:CycleLSD[2,3])),r
604 CutL10SD=c(CycleLSD[8,2]:CycleLSD[8,3],CycleLSD[9,2]:CycleLSD[9,3],CycleLSD[10,2]:CycleLSD[10,3],CycleL
605 CycleL10SD=c(rep("A",length(CycleLSD[8,2]:CycleLSD[8,3])),rep("B",length(CycleLSD[9,2]:CycleLSD[9,3])),r
606 CutL15SD=c(CycleLSD[15,2]:CycleLSD[15,3],CycleLSD[16,2]:CycleLSD[16,3],CycleLSD[17,2]:CycleLSD[17,3],Cy
607 CycleL15SD=c(rep("A",length(CycleLSD[15,2]:CycleLSD[15,3])),rep("B",length(CycleLSD[16,2]:CycleLSD[16,3]),r
608 CutL20SD=c(CycleLSD[22,2]:CycleLSD[22,3],CycleLSD[23,2]:CycleLSD[23,3],CycleLSD[24,2]:CycleLSD[24,3],Cy
609 CycleL20SD=c(rep("A",length(CycleLSD[22,2]:CycleLSD[22,3])),rep("B",length(CycleLSD[23,2]:CycleLSD[23,3]),r
610 CutL25SD=c(CycleLSD[29,2]:CycleLSD[29,3],CycleLSD[30,2]:CycleLSD[30,3],CycleLSD[31,2]:CycleLSD[31,3],Cy
611 CycleL25SD=c(rep("A",length(CycleLSD[29,2]:CycleLSD[29,3])),rep("B",length(CycleLSD[30,2]:CycleLSD[30,3]),r
612
613 CutLSD=TestSD[c(CutL5SD,CutL10SD,CutL15SD,CutL20SD,CutL25SD),]
614 CutLSD$Cycle=c(CycleL5SD,CycleL10SD,CycleL15SD,CycleL20SD,CycleL25SD)
```

simulateScenario.R:

```
55 # Define the biomass cycles #####
56 ## Find the maximums and minimums and then get all the cycle's times ----
57 CycleXSD2 <- scenario_data[
58   by = .(Temperature),
59   j = .(
60     Max = findPeaks(get(col_name)),
61     # Set minimum whilst selecting the correct correction for L or G using a switch
62     Min = switch(col_name, "L" = findValleys(L)[seq(2,14,2)], "G" = c(findValleys(G), 2555))
63   )
64 ] %>%
65   # Create new column 'Indices' with sequences of all the times in the cycles
66   "$<-"(Indices, apply(., 1, function(cur_row) seq(cur_row[[2]], cur_row[[3]])))
67
68 # For each temperature, get the list with indices and cycle identifiers
69 createPerTempLists <- function(ind_lists) {
70   # Returns a named list containing INDICES and IDENTIFIERS, both in a single array, for the given
71   getIdentifiers <- function(ind_lists) {
72     # For each cycle create a list repeating the identifying letter for the length of that cycle
73     sapply(1:length(ind_lists), function(i) rep( LETTERS[i], length(ind_lists[[i]])) )
74   }
75   return( list(Indices = unlist(ind_lists), Identifiers = unlist(getIdentifiers(ind_lists))) )
76 }
77 per_temp_lists <- tapply(CycleXSD2$Indices, CycleXSD2$Temperature, createPerTempLists)
78
79 ## Combine indices and identifiers of all temperatures a single vector ----
80 all_indices <- sapply(1:length(per_temp_lists), function(i) per_temp_lists[[i]]$Indices + 2555 * (i - 1))
81 all_identifiers <- sapply(1:length(per_temp_lists), function(i) per_temp_lists[[i]]$Identifiers)
82
83 ## Subset data using the previously created vector ----
84 cut_dt <- scenario_data[unlist(all_indices)] %>%
85   # Add a column from the vector with identifiers
86   "$<-"( Cycle, unlist(all_identifiers) ) %>%
87   # Drop the first cycle for each temperature
88   "["(Cycle != "A")
```

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.
- 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 so these need to be removed as to not taint the results. This was then coded in the literal way as it was just said, however, the code behaves different than they likely expected so only 16 days were removed per year. Thus the removal only happened each year for the temperature of 5 degrees Celsius, this was easily fixed by grouping by year *and* temperature.
- To calculate the decreasing slopes the `quantmod` library's `findPeaks` and `findValleys` functions were used. These however have a flaw in them causing the peaks and valleys to always overshoot by 1 row, to fix this all indices were subtracted by 1.

4.2 General conclusion and perspective

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](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. Hemptonne, 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’

```
1 ## Copyright (c) 2023 Vincent Talen.
2 ## Licensed under GPLv3. See LICENSE file.
3 ## -----
4 ##
5 ## Script name: model.R
6 ##
7 ## Purpose of script: Implements the biological model of consumer-resource dynamics
8 ##
9 ## Author: Vincent Talen
10 ##
11 ## Date Created: 09 Jan 2023
12 ##
13 ## Email: v.k.talen@st.hanze.nl
14 ##
15 ## -----
16 ##
17 ## Notes:
18 ## - Goal: formulate formula functions in a more readable/recognizable manner
19 ##
20 ## -----
21
22
23 # #####
24 # Libs #
25 # #####
26 library(deSolve)
27
28
29 # #####
30 # Code #
31 # #####
32 # Define Boltzmann term (°K)
33 boltz_const <- 8.62 * 10^-5
34 # Define mean inverse temperature (calculated from Data_Mismatch.txt)
35 mean_inverse_temp <- 40.5941593143742
36
37
38 ## ---- mte formulations ----
39 # Quadratic function for metabolic rate ( $\mu\text{g C/day}$ )
40 calcMetabolicRate <- function(T.C, M) {
41   alpha <- exp(2.41599)      # metabolic expression level at reference temperature
42   b <- 0.62308              # mass-scaling exponent
43   p <- 0.66731              # curve steepness (of the relationship)
44   q <- 0.21153              # quadratic term
45   T.K <- T.C + 273.15       # convert temperature from Celsius to Kelvin
46
47   # Repeating part with temperatures
48   temp_dependency_part <- (1 / (T.K * boltz_const)) - mean_inverse_temp
```

```

49
50 # Calculate metabolic rate with full formula
51 metabolic_rate <- alpha * M^b * exp(-p * temp_dependency_part) * exp(-q * temp_dependency_part^2)
52 return(metabolic_rate)
53 }

54
55 # Quadratic function for ingestion rate ( $\mu\text{g C/day}$ )
56 calcIngestionRate <- function(T.C, M) {
57   alpha <- exp(5.26814)           # ingestion expression level at reference temperature
58   b <- 0.81654                  # mass-scaling exponent
59   p <- 0.31876                  # curve steepness (of the relationship)
60   q <- 0.18909                  # quadratic term
61   T.K <- T.C + 273.15          # convert temperature from Celsius to Kelvin
62
63   # Repeating part with temperatures
64   temp_dependency_part <- (1 / (T.K * boltz_const)) - mean_inverse_temp
65
66   # Calculate ingestion rate with full formula
67   ingestion_rate <- alpha * M^b * exp(-p * temp_dependency_part) * exp(-q * temp_dependency_part^2)
68   return(ingestion_rate)
69 }

70
71
72 ## ---- assimilation efficiency ----
73 # Assimilation efficiency function based on exponential decay (quadratic model)
74 # Is a logistic equation with the MTE equation both at the numerator and the denominator
75 calcAssimEff <- function(T.C) {
76   alpha <- exp(-0.84730)        # normalization constant of assimilation efficiency
77   Ea <- 0.16400                # activation energy
78   T.0 <- 285.65                 # reference temperature of 12.5 degrees Celsius in Kelvin
79   T.K <- T.C + 273.15          # convert temperature from Celsius to Kelvin
80
81   mte_equation <- alpha * exp( Ea * (T.K - T.0) / (boltz_const * T.0 * T.K) )
82   return(mte_equation / (1 + mte_equation))
83 }

84
85
86 ## ---- attack rate parameter ----
87 # Attack rate function based on exponential decay (quadratic model)
88 calcAttackRate <- function(temp, mass) {
89   t <- 2                         # Experiment duration
90   M.L <- 4612.5                  # Mean initial C mass of leaf discs in microcosms ( $\mu\text{g C}$ )
91   decay_rate <- -log(1 - calcIngestionRate(temp, mass) * t / M.L)
92   return( decay_rate / t )
93 }

94
95
96 ## ---- handling time parameter ----
97 # Handling time function based on exponential decay (quadratic model)
98 calcHandlingTime <- function(temp, mass) {1 / (calcIngestionRate(temp, mass) / 1000) }
99
100 ## ---- leaf decomposition- and respiration rate ----
101 # Function for the leaf litter microbial decomposition rate (Arrhenius equation)
102 calcLeafDecomp <- function(T.C) {

```

```

103 k.10C <- 0.00956          # litter decomposition rate at 10°C (283.15°K)
104 Ea <- 0.37000            # activation energy
105 T.K <- T.C + 273.15      # convert temperature from Celsius to Kelvin
106 # return( k.10C * exp(-Ea * (1 / (boltz_const * T.K - 283.15 * boltz_const))) )
107 return( k.10C * exp(-Ea * (1 / (boltz_const * T.K) - 10)) )
108 }
109
110
111 ## ---- consumer-resource model ----
112 GammLeafModel <- function(temp, gamm_indv_mass, leaf_fall, gamm_start_biomass, tsr_model) {
113   # Apply TSR Model to Gammarus individual mass
114   if (!is.null(tsr_model)) {
115     gamm_indv_mass <- tsr_model(temp, gamm_indv_mass)
116   }
117
118   Nutri <- function(time, state, parms) {
119     with(as.list(c(state, parms)), {
120       fL <- a * L / (1 + a * h * L)           # Holling type II functional response
121       dL <- -fL * G - k * L                  # Biomass changes of leaf litter stock
122       dG <- G * (fL * A - M)                 # Biomass changes of Gammarus population
123       list(c(dL, dG))
124     })
125   }
126
127   # Leaf litter fall event function
128   leafFallEvent <- function(time, state, parms) {
129     with(as.list(c(state, parms)), {
130       return(c(L + leaf_fall, G))
131     })
132   }
133
134   # Get time points to trigger litter fall event (first 15 days of the year)
135   getFallTimesYearX <- function(year) { seq(year * 365 + 1, year * 365 + 15) }
136   leaf_fall_times <- unlist(lapply(seq(0, 6), getFallTimesYearX))
137
138   # Model parameters
139   parameters <- c(
140     M = calcMetabolicRate(temp, gamm_indv_mass) / 1000,    # Gammarus metabolic rate (in mgC/day)
141     a = calcAttackRate(temp, gamm_indv_mass),               # Gammarus attack rate (in mgC/day)
142     h = calcHandlingTime(temp, gamm_indv_mass),             # Gammarus handling time (in 1/day)
143     A = calcAssimEff(temp),                                # Gammarus assimilation efficiency
144     k = calcLeafDecomp(temp)                               # Leaf microbial decomposition (in 1/day)
145   )
146
147   # Times and starting conditions
148   times <- seq(0, 365 * 7, by = 1)                      # Times in days for 7 years
149   state <- c(L = leaf_fall, G = gamm_start_biomass)      # Starting biomasses (in g/m2)
150
151   # Model output
152   out <- ode(time = times, func = Nutri, y = state, parms = parameters,
153              events = list(func = leafFallEvent, time = leaf_fall_times))
154
155   # Turn deSolve class object into dataframe and change very low and negative values to 0

```

```

156     data_table <- as.data.table(out) %>% mutate(across(c(L, G), ~ ifelse(.x < 10^-3, 0, .x)))
157     return(data_table)
158 }
159
160
161 ## ---- temperature-size rule models ----
162 # Average TSR response
163 calcTSR.Avg <- function(temp, mass) {
164   conv_fact <- 6.5
165   change_slope <- -3.90 - 0.53 * log10(mass)
166   change_prop <- log(1 + change_slope / 100)
167   change_const <- exp(log(mass) - 12.5 * change_prop)
168
169   dry_mass <- change_const * exp(change_prop * (temp))
170   fresh_mass <- dry_mass / conv_fact
171   return(dry_mass)
172 }
173
174 # Maximum TSR response
175 calcTSR.Max <- function(temp, mass) {
176   conv_fact <- 6.5
177   change_slope <- -8.0
178   change_prop <- log(1 + change_slope / 100)
179   change_const <- exp(log(mass) - 12.5 * change_prop)
180
181   dry_mass <- change_const * exp(change_prop * (temp))
182   fresh_mass <- dry_mass / conv_fact
183   return(dry_mass)
184 }
```

Appendix B: ‘functions.R’

```

1 ## Copyright (c) 2023 Vincent Talen.
2 ## Licensed under GPLv3. See LICENSE file.
3 ## -----
4 ##
5 ## Script name: functions.R
6 ##
7 ## Purpose of script: Functions
8 ##
9 ## Author: Vincent Talen
10 ##
11 ## Date Created: 09 Jan 2023
12 ##
13 ## Email: v.k.talen@st.hanze.nl
14 ##
15 ## -----
16 ##
17 ## Notes:
18 ##      - X
19 ##
```

```

20 ## ~~~~~
21
22
23 # #####
24 # Libs #
25 # #####
26 library(data.table)
27 library(ggpubr)
28 library(reshape2)
29 library(tidyverse)
30 source("src/model.R")
31
32
33 # #####
34 # Functions #
35 # #####
36 # ---- Scenario data gathering and preparations ----
37 getScenarioDataList <- function(gamm_indv_mass, leaf_fall, gamm_start_biomass, tsr_model) {
38   # Get data for given values for each temperature using the model function that performs an ode
39   data_list <- lapply(temperatures, GammLeafModel, gamm_indv_mass, leaf_fall, gamm_start_biomass, tsr_model)
40   setNames(data_list, temperatures)
41   return(data_list)
42 }
43
44 createLongDataFrame <- function(df_list, tsr_model) {
45   if (is.null(tsr_model)) { tsr_model <- function(temp, mass) {return(mass)} }
46
47   # Function to get the population metabolism for a temperature with the population biomass
48   getPopMetabolism <- function(cur_temp, gamm_pop_biomass) {
49     # Get metabolic rate for current temperature
50     meta_rate <- calcMetabolicRate(cur_temp, tsr_model(cur_temp, gamm_indv_mass)) / 1000 # Gammarus metabolism
51     # Calculate population metabolism
52     pop_metabolism <- meta_rate * gamm_pop_biomass
53     return(fifelse(pop_metabolism < 0, 0, pop_metabolism))
54   }
55   # Function to get the population ingestion for a temperature with the population- and leaf biomasses
56   getPopIngestion <- function(cur_temp, leaf_biomass, gamm_pop_biomass) {
57     # Get ingestion- and attack rates for current temperature
58     ingest_rate <- calcIngestionRate(cur_temp, tsr_model(cur_temp, gamm_indv_mass)) / 1000 # Gammarus ingestion
59     attack_rate <- calcAttackRate(cur_temp, tsr_model(cur_temp, gamm_indv_mass)) # Gammarus attack
60     # Calculate population leaf ingestion
61     pop_ingestion <- (attack_rate * leaf_biomass / (1 + attack_rate * 1 / ingest_rate * leaf_biomass))
62     return(fifelse(pop_ingestion < 0, 0, pop_ingestion))
63   }
64
65   # Bind all dataframes from list to single big one and
66   # drop the last days to have 2555 days/rows left per temperature
67   big_df <- rbindlist(df_list)[!time == 2555] %>%
68     # Rename 'time' column to conform to naming scheme
69     setnames("time", "Time") %>%
70     # Add temperature and year columns to facilitate future calculations
71     "$<-(Temperature, rep(temperatures, each = 2555)) %>%
72     "$<-(Year, rep(rep(1:7, each = 365), 5)) %>%

```

```

73  # Add population metabolism and leaf ingestion columns
74  "$<-"(M, getPopMetabolism(.Temperature, .G)) %>%
75  "$<-"(I, getPopIngestion(.Temperature, .L, .G)) %>%
76  # Set column order to a nicer one
77  setcolorder(c("Time", "L", "G", "M", "I", "Temperature", "Year"))
78  return(big_df)
79 }
80
81 # ---- Plot list of scenario dataframes ----
82 createPlotForTemp <- function(cur_temp, cur_data) {
83   # Divide L & G values to create a better readable plot
84   divided_data <- copy(cur_data)
85   set(divided_data, i = NULL, "L", divided_data$L / 10^5)
86   set(divided_data, i = NULL, "G", divided_data$G / 10^5)
87
88   # Create plot
89   plot <- ggplot(divided_data, aes(x = time, y = value)) +
90     # Set axis limits and step size
91     scale_x_continuous(breaks = seq(0, 7 * 365, 365)) +
92     ylim(NA, ceiling(max(divided_data[, -1])) + 1) +
93     # Add the data (lines)
94     geom_line(aes(y = L, color = "Leaf Litter Biomass")) +
95     geom_line(aes(y = G, color = "Gammarus Fossarum Biomass")) +
96     # Add styling
97     labs(title = sprintf("%s°C", cur_temp), x = "", y = "") +
98     theme(
99       # plot.title=element_text(hjust=0.075, vjust=-11), # offset title inside the plot
100      plot.margin=margin(0.1, 0.25, 0, 0, "cm"), # enlarge plots and decrease gaps between them
101      legend.text=element_text(size=12), # for better readability increase legend font size
102      legend.key.width=unit(1,"cm") # widen the legend keys
103    ) +
104    scale_color_manual(
105      name = "",
106      values = c("black", "tomato2"),
107      limits = c("Leaf Litter Biomass", "Gammarus Fossarum Biomass")
108    ) +
109    guides(colour=guide_legend(override.aes=list(size=1))) # Increase legend entries' line width
110  return(plot)
111 }
112
113 plotScenarioDynamics <- function(data, image_title, file_out) {
114   # Use lapply to create plots for each temperature in the list and collect the legend from a plot
115   plot_list <- lapply(seq_along(data), function(i) { createPlotForTemp(names(data)[i], data[[i]]) })
116   plot_legend <- get_legend(plot_list[[1]])
117
118   # Remove legends from the plots and add extracted legend to end of the list
119   plot_list <- lapply(plot_list, function(cur_plot) { cur_plot + theme(legend.position = "none") }) %>%
120     "[[<-(length(plot_list) + 1, plot_legend)
121
122   # Place plots and legend in an arranged grid
123   col_num <- 3
124   my.grid <- ggarrange(plotlist = plot_list, ncol = col_num, nrow = ceiling(length(plot_list) / col_num),
125     annotate_figure(

```

```

126     top = text_grob(image_title, size=16, face="bold"),
127     bottom = text_grob("Time (d)"),
128     left = text_grob(bquote("Biomass "(10^5~ mg~ C~ m^-2)), rot = 90))
129
130 # Save the created arranged grid with the lossless 'lzw' compression that greatly reduces file size
131 ggsave(paste("figures/reproduced_plots/", file_out, sep=""), bg = "white", width=15, height=8, units=
132 dev.off()
133 }
134
135 # ---- Create prediction data ----
136 createPredictionData <- function(huidige_data, scenario_names) {
137   # Get vector from lowest temperature to highest temperature with steps of 0.5
138   Temperature <- seq(head(temperatures, n=1), tail(temperatures, n=1), by=0.5)
139   # Create vector with scenario names times the number of temperature steps
140   Scenario <- rep(scenario_names, each=length(Temperature))
141   # Create dataframe
142   Pred <- data.table(Temperature, Scenario)
143
144   # Each column with 'Mean' should get their predicted values for smooth lines
145   for(cname in colnames(huidige_data)) {
146     if (startsWith(cname, "Mean")) {
147       new_col_name <- gsub("Mean", "Pred", cname)
148       predicted_data <- predict(lmList(get(cname) ~ poly(Temperature, 2)|Scenario, data=huidige_data),
149       Pred[ , (new_col_name) := predicted_data]
150     }
151   }
152   return(Pred)
153 }
```

Appendix C: ‘simulateScenario.R’

```

1 ## Copyright (c) 2023 Vincent Talen.
2 ## Licensed under GPLv3. See LICENSE file.
3 ##
4 ##
5 ## Script name: simulateScenario.R
6 ##
7 ## Purpose of script: Make simulating a scenario easy with this one function
8 ##
9 ## Author: Vincent Talen
10 ##
11 ## Date Created: 29 Jun 2023
12 ##
13 ## Email: v.k.talen@st.hanze.nl
14 ##
15 ##
16 ##
17 ## Notes:
18 ## - x
19 ##
20 ##
```

```

21
22
23 # ##### #
24 #   Libs   #
25 # ##### #
26 library(quantmod)
27 library(lme4)
28
29
30 # ##### #
31 #   Code   #
32 # ##### #
33 simulateScenario <- function(scenario_data, scenario_name) {
34   simulateColumnScenario <- function(scenario_data, scenario_name, col_name, threshold) {
35     # Exclude the first year and the first 16 days with leaf fall for all years for each temperature
36     cut_df <- scenario_data[!Year == "1", tail(.SD, -16), by = .(Year)] %>%
37       setcolorder(colnames(scenario_data)) # Restore column order
38
39   # Biomass mean for each temperature #####
40   Biom <- cut_df %>%
41     # Calculate mean annual biomass per temperature, per year
42     "["(j = .(all_means = mean(get(col_name))), by = .(Temperature, Year)) %>%
43     # Calculate biomass mean and deviation over 6 years per temperature
44     "["(j = .(MeanBiom = mean(all_means), SdBiom = sd(all_means)), by = Temperature)
45
46   # Calculate persistence time for each temperature with the threshold #####
47   PersTime <- cut_df %>%
48     # Only keep days above the threshold
49     "["(i = .[[col_name]] > threshold) %>%
50     # Count the days above the threshold per temperature, per year
51     "["(j = .(days_above = .N), by = .(Temperature, Year)) %>%
52     # Calculate persistence time mean and deviation over 6 years per temperature
53     "["(j = .(MeanPersTime = mean(days_above), SdPersTime = sd(days_above)), by = Temperature)
54
55   # Define the biomass cycles #####
56   ## Find the maximums and minimums and then get all the cycle's times ----
57   CycleXSD2 <- scenario_data[
58     by = .(Temperature),
59     j = .(
60       Max = findPeaks(get(col_name)), #-1,
61       # Set minimum whilst selecting the correct correction for L or G using a switch
62       #Min = switch(col_name, "L" = findValleys(L)[seq(2,14,2)]-1, "G" = c(findValleys(G)-1, 2555))
63       Min = switch(col_name, "L" = findValleys(L)[seq(2,14,2)], "G" = c(findValleys(G), 2555))
64     )
65   ] %>%
66     # Create new column 'Indices' with sequences of all the times in the cycles
67     "$<"(Indices, apply(., 1, function(cur_row) seq(cur_row[[2]], cur_row[[3]])))
68
69   # For each temperature, get the list with indices and cycle identifiers
70   createPerTempLists <- function(ind_lists) {
71     # Returns a named list containing INDICES and IDENTIFIERS, both in a single array, for the given
72     getIdentifiers <- function(ind_lists) {
73       # For each cycle create a list repeating the identifying letter for the length of that cycle
74       sapply(1:length(ind_lists), function(i) rep( LETTERS[i], length(ind_lists[[i]])) )

```

```

75     }
76     return( list(Indices = unlist(ind_lists), Identifiers = unlist(getIdentifiers(ind_lists))) )
77   }
78   per_temp_lists <- tapply(CycleXSD2$Indices, CycleXSD2$Temperature, createPerTempLists)
79
80   ## Combine indices and identifiers of all temperatures a single vector ----
81   all_indices <- sapply(1:length(per_temp_lists), function(i) per_temp_lists[[i]]$Indices + 2555 * (i - 1))
82   all_identifiers <- sapply(1:length(per_temp_lists), function(i) per_temp_lists[[i]]$Identifiers)
83
84   ## Subset data using the previously created vector ----
85   cut_dt <- scenario_data[unlist(all_indices)] %>%
86     # Add a column from the vector with identifiers
87     "$<-"(Cycle, unlist(all_identifiers)) %>%
88     # Drop the first cycle for each temperature
89     "["(Cycle != "A")
90
91   # Calculate litter and Gammarus mean maximum biomasses and decreasing slopes ----
92   ## Get the first time each cycle per temperature where the biomass is below the threshold
93   first_below_threshold <- cut_dt[get(col_name) < threshold, head(.SD, 1), by = .(Temperature, Cycle)]
94   ## Also get the highest biomqss (first) value for each cycle per temperature
95   maximum_values <- cut_dt[, head(.SD, 1), by = .(Temperature, Cycle)]
96
97   ## Calculate the slope for each cycle using the maximum and threshold times
98   SlopLSD3 <- rbind(first_below_threshold, maximum_values) %>%
99     "["(j = .(Slope = coef(lm(get(col_name) ~ Time))[2])), by = .(Cycle, Temperature) )
100
101  Maxi <- maximum_values[, .(MeanMax = mean(get(col_name)), SdMax = sd(get(col_name))), by = Temperature]
102  Slop <- SlopLSD3[, .(MeanSlope = mean(Slope), SdSlope = sd(Slope)), by = Temperature]
103
104  # Put all the data tables into a single one
105  final_df <- setDT(c(Biom, PersTime, Maxi, Slop)) %>%
106    # Remove duplicate 'Temperature' columns
107    "["(j = which(duplicated(names(.))) := NULL)
108    # Add L or G (current column argument) to column names
109    colnames(final_df)[2:9] <- sapply(colnames(final_df)[2:9], function(cur_col) paste(cur_col, col_name))
110  return(final_df)
111}
112
113 # Use above function to get data
114 Leaf <- simulateColumnScenario(scenario_data, scenario_name, "L", 60000)
115 Gamm <- simulateColumnScenario(scenario_data, scenario_name, "G", 5000)
116
117 # Put all the information in a single data.table
118 combined_df <- setDT(c(Leaf, Gamm)) %>%
119  # Remove duplicate 'Temperature' columns
120  "["(j = which(duplicated(names(.))) := NULL)
121
122  # Add scenario column
123  combined_df <- add_column(combined_df, Scenario=rep(scenario_name, 5), .after="Temperature")
124  return(combined_df)
125}

```

Appendix D: ‘mainAnalysis.R’

```
1 ## Copyright (c) 2023 Vincent Talen.
2 ## Licensed under GPLv3. See LICENSE file.
3 ##
4 ##
5 ## Script name: mainAnalysis.R
6 ##
7 ## Purpose of script: Perform all scenario simulations for the main analysis
8 ##
9 ## Author: Vincent Talen
10 ##
11 ## Date Created: 29 Jun 2023
12 ##
13 ## Email: v.k.talen@st.hanze.nl
14 ##
15 ##
16 ##
17 ## Notes:
18 ## - x
19 ##
20 ##
21
22
23 # ##### #
24 # Libs #
25 # #####
26 source("src/functions.R")
27 source("src/simulateScenario.R")
28
29
30 # #####
31 # Code #
32 # #####
33 # SETTINGS FOR ALL SCENARIOS #####
34 # Temperatures to do simulations of
35 temperatures <- c(5, 10, 15, 20, 25)
36
37 # Duration of the leaf fall in days
38 fall_duration_in_days <- 15
39
40 # Gammarus mean body mass = 4.26 mgDM
41 gamm_indv_mass <- 4.26
42 # Annual leaf fall = 300 gC/m2/an = 300 000 mgC/m2/an
43 leaf_fall <- 300000 / fall_duration_in_days
44 # Gammarus density = 30 mgDM/m2 = 15 mgC/m2
45 gamm_start_biomass <- 15
46
47
48 # SCENARIO 0: REFERENCE SCENARIO #####
49 # Get data for temperatures with values of current scenario
50 df_list.TSRR <- getScenarioDataList(gamm_indv_mass, leaf_fall, gamm_start_biomass, NULL)
```

```

52 # Create plots in an arranged grid
53 file_out.TSRR <- "Population Dynamics Reference Scenario.png"
54 image_title.TSRR <- "Reference Scenario: Population Dynamics over 7 years"
55 plotScenarioDynamics(df_list.TSRR, image_title.TSRR, file_out.TSRR)
56
57 # Combine dataframes into one and add temperature, year, population metabolism- and ingestion columns
58 combined_df.TSRR <- createLongDataFrame(df_list.TSRR, NULL)
59 # Simulate scenario and get final dataframes for both types of masses
60 data.TSRR <- simulateScenario(combined_df.TSRR, "TSRR")
61
62
63 # SCENARIO 1: AVERAGE TSR RESPONSE #####
64 # Get data for temperatures with values of current scenario
65 df_list.TSRA <- getScenarioDataList(gamm_indev_mass, leaf_fall, gamm_start_biomass, calcTSR.Avg)
66
67 # Create plots in an arranged grid
68 file_out.TSRA <- "Population Dynamics Average TSR Scenario.png"
69 image_title.TSRA <- "Average Temperature-Size Rule Response: Population Dynamics over 7 years"
70 plotScenarioDynamics(df_list.TSRA, image_title.TSRA, file_out.TSRA)
71
72 # Combine dataframes into one and add temperature, year, population metabolism- and ingestion columns
73 combined_df.TSRA <- createLongDataFrame(df_list.TSRA, calcTSR.Avg)
74 # Simulate scenario and get final dataframes for both types of masses
75 data.TSRA <- simulateScenario(combined_df.TSRA, "TSRA")
76
77
78 # SCENARIO 2: MAXIMUM TSR RESPONSE #####
79 # Get data for temperatures with values of current scenario
80 df_list.TSRM <- getScenarioDataList(gamm_indev_mass, leaf_fall, gamm_start_biomass, calcTSR.Max)
81
82 # Create plots in an arranged grid
83 file_out.TSRM <- "Population Dynamics Maximum TSR Scenario.png"
84 image_title.TSRM <- "Maximum Temperature-Size Rule Response: Population Dynamics over 7 years"
85 plotScenarioDynamics(df_list.TSRM, image_title.TSRM, file_out.TSRM)
86
87 # Combine dataframes into one and add temperature, year, population metabolism- and ingestion columns
88 combined_df.TSRM <- createLongDataFrame(df_list.TSRM, calcTSR.Max)
89 # Simulate scenario and get final dataframes for both types of masses
90 data.TSRM <- simulateScenario(combined_df.TSRM, "TSRM")
91
92
93 ## PLOTTING FUNCTIONS #####
94 createPlotForStateVariable <- function(state_variable_name, statistic_info, data_column_names, all_scenario_data)
95   # Calculate the y-axis limits
96   max_lim <- round(max(all_scenario_data[,get(data_column_names["Mean"])]) + statistic_info$correction)
97   break_by <- ifelse(max_lim <= 4, 1, 2)
98
99   plot <- ggplot(all_scenario_data, aes(x=Temperature, y=abs (!!sym(data_column_names["Mean"])), group=Scenario))
100  # Add the actual scenario simulation points to the plot
101  geom_point(
102    aes(color=Scenario),
103    size=3,
104    position=position_dodge(0.5)

```

```

105
106 ) +
107 # Add the prediction data lines to the plot
108 geom_line(
109   data=prediction_data,
110   aes(x=Temperature, y=abs(!!sym(data_column_names["Pred"])), color=Scenario, linetype=Scenario),
111   show.legend=F
112 ) +
113 # Add the standard deviation error bar to the plot
114 geom_errorbar(
115   color="grey50",
116   width=0.75,
117   position=position_dodge(0.5),
118   aes(ymin=(abs(!!sym(data_column_names["Mean"])) - !!sym(data_column_names["Sd"]))),
119   ymax=(abs(!!sym(data_column_names["Mean"])) + !!sym(data_column_names["Sd"])))
120 ) +
121 # Add the plot title and axis labels
122 labs(x = "Temperature (\u00b0C)", y=statistic_info$y_label, title=state_variable_name) +
123 # Style the plot title, axis and legend texts
124 theme(
125   axis.text.y=element_text(size=10, colour="black"),
126   axis.text.x=element_text(size=10, colour="black"),
127   plot.title = element_text(size=12),
128   legend.title=element_text(face="bold")
129 ) +
130 # Set the y-axis limits
131 scale_y_continuous(labels=function(x) {x / statistic_info$correction}) +
132 expand_limits(y = 0) +
133 # Color and types of plotted data elements
134 scale_color_manual(values=c(TSRR="black", TSRA="steelblue2", TSRM="tomato2")) +
135 scale_linetype_manual(values=c("TSRR"="dotted", "TSRA"="solid", "TSRM"="solid"))
136 return(plot)
137 }
138
139 plotGridOfStatistic <- function(statistic_info, all_scenario_data, prediction_data) {
140   # Define state variables
141   state_variable_list <- list(
142     list(id="L", full_name="Leaf Litter"),
143     list(id="G", full_name="Gammarus Fossarum")
144   )
145   # Create the plots for the state variables in a list using lapply
146   state_variable_plots <- lapply(state_variable_list, function(state_variable) {
147     # Get the full names of the data columns for the current statistic + state variable combination
148     data_column_names <- sapply(
149       c("Mean", "Sd", "Pred"), # for each data column
150       paste, # perform the paste function
151       state_variable$id, # put the id of the state variable at the end
152       sep=statistic_info$name # and the name of the statistic in between
153     )
154     # Create the plot of the data columns for the current statistic + state variable combination
155     createPlotForStateVariable(state_variable$full_name, statistic_info, data_column_names, all_scenario_data)
156   })
157
158   # Return the plots in an arranged grid with the legend at the bottom

```

```

158     arranged_plots <- ggpubr::ggarrange(plotlist=state_variable_plots, ncol=2, nrow=1, common.legend=TRUE)
159     return(arranged_plots)
160   }
161
162 createFigureForStatistic <- function(statistic_info, all_scenario_data) {
163   # Get prediction data that is used for the continuous lines in the plots
164   prediction_data <- createPredictionData(all_scenario_data, c("TSRR", "TSRA", "TSRM"))
165   # Create the arranged grid with the two plots of the state variables for the statistic and annotate them
166   plotGridOfStatistic(statistic_info, all_scenario_data, prediction_data) %>%
167     annotate_figure(top=text_grob(statistic_info$image_title, size=16, face="bold"))
168   # Save the final figure for the column
169   ggsave(
170     filename=paste("figures/reproduced_plots/", statistic_info$image_title, ".png", sep=""),
171     bg="white",
172     width=3840,
173     height=2160,
174     units="px",
175     dpi=300
176   )
177   dev.off()
178 }
179
180
181 ## CREATING PLOTS #####
182 # Combine the data frames into a single one
183 combined_data <- rbind(data.TSRR, data.TSRA, data.TSRM)
184
185 # Create a list with the statistics that a figure needs to be made for
186 statistic_info_list <- list(
187   list(name = "Biom",
188     correction = 10^4,
189     y_label = expression('Mean biomass'~'(*10^4~mg~C~m^-2*)'),
190     image_title = "Mean Biomass over Temperature"),
191   list(name = "PersTime",
192     correction = 1,
193     y_label = "Persistance time (days)",
194     image_title = "Persistance Time over Temperature"),
195   list(name = "Slope",
196     correction = 10^4,
197     y_label = expression('Mean biomass slope'~'(*10^4~mg~C~m^-2~day^-1*)'),
198     image_title = "Mean Biomass Slope over Temperature")
199 )
200
201 # Create a figures for each statistic defined above
202 for (statistic_info in statistic_info_list) {
203   createFigureForStatistic(statistic_info, combined_data)
204 }

```