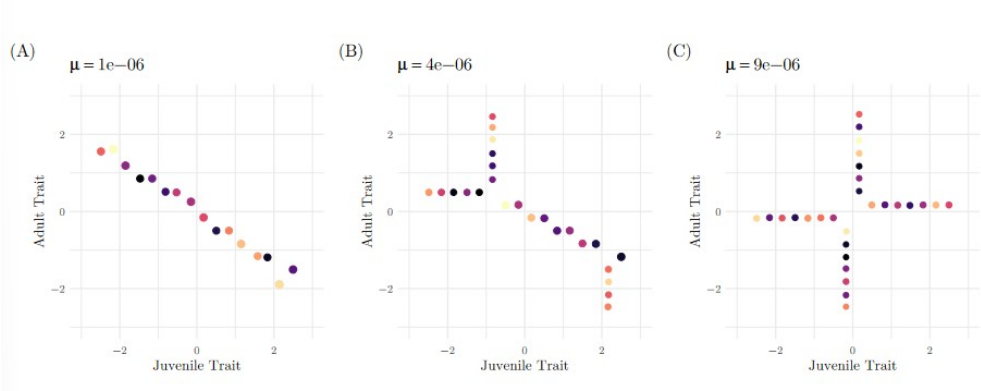
UPPSALA
UNIVERSITET

Relaxation of the separation between ecological and evolutionary timescales has unexpected effects on community assembly of species with complex life cycles.



Izabel Eriksson Reuterborg

Degree project in biology, Master of science (2 years), 2024
Examensarbete i biologi 60 hp till masterexamen, 2024
Biology Education Centre and Biology Education Centre and Department of Ecology and Genetics, Uppsala University
Supervisor: Claus Rueffler
External opponent: Mattias Siljestam

Abstract

A substantial proportion of species have a life cycle with at least two distinct stages. However, explorations of consumer-resource interactions seldom account for stage structure within populations. Understanding the effect of stage structure on community ecology is important, and might help elucidate phenomena that have otherwise been hard to explain. In this thesis the idea that more species can coexist if they possess a complex life cycle is explored using an individual-based discrete time model. I compare the number of coexisting species that result from evolution in communities with species only having simple or only having complex life cycles. Several different niche widths, mutation probabilities and resource distributions are explored. The results substantiate previous findings that complex life cycles can, but do not inherently lead to more coexisting species unless they are not constrained by gradual evolution. The novel result from this model is that when niche width is small and the mutation rate is relatively high, complex life cycles result in more species rich community than simple life cycles.

Keywords:

Complex life cycles, ontogenetic niche shift, coexistence, adaptive diversification, speciation, immigration, gradual evolution.

Introduction

Explorations of consumer-resource interactions often assume that all individuals of a given species are essentially identical (Miller and Rudolf, 2011). In reality, this is not the case as most populations are structured. Many organisms undergo a dramatic change, such that they occupy two separate niches in succession during their ontogeny, known as an *ontogenetic niche shifts*. Organisms who undergo this process have a *complex life cycle* (Wilbur, 1980). I here refer to organisms with complex life cycles as *complex organisms*, and organisms that do not undergo an ontogenetic niche shift as *simple organisms*. This does not refer to complexity in any other aspect of the organism, and is simply used for the sake of brevity. Common representatives of complex organisms include anurans and holometabolous insects, taxa that undergo complete metamorphosis. While understanding the dynamics of these taxa alone could itself be considered a noteworthy pursuit, it has been argued that ontogenetic niche shifts are much more widespread, and not just limited to species that undergo metamorphosis (Werner and Gilliam, 1984). Werner (1988) argued that more than 80% of species experience an ontogenetic niche shift during their lifetime. If 80% of taxa have this structure within populations, then it is important to resolve how this affects predictions from community ecology. How does it affect the stability of food webs, inter- and intraspecific competition, species diversity, and coexistence?

Ontogenetic niche shifts can be due to changes in morphology, behaviour, or habitat (Wilbur, 1980). Generally, it results in an organism that is adapted to at least two individual resources or environments. Importantly, in organisms with metamorphosis these adaptions can be independent, and can therefore evolve separately (Sherratt et al., 2017). Understanding the effect of ontogenetic niche shifts on ecosystems is important as it can, among other things, affect biodiversity loss. A model by Rudolf and Lafferty (2011) argued that a food web containing complex organisms was at increased risk of collapsing due secondary extinctions caused by species loss. Mougi (2017) reached similar conclusions for small, less connected food webs but interestingly also found that the introduction of complex species into large interconnected food webs increased stability. Complex life cycles can also effectively couple ecosystems that might otherwise be considered separate, as an effect on one life stage can affect the ecosystem where the other life stage resides (Schreiber and Rudolf, 2008). It has also been proposed that ontogenetic niche shifts could increase the number of species that can coexist within a community. One argument by Wilbur (1980) was that complex organisms might promote species diversity because stable coexistence of two species can be achieved through niche partitioning in only one of the life stages. For example, if the juvenile stage has a competitive advantage in one of the species, and the adult stage has a competitive advantage in the other. Many authors have investigated this idea since. For example Loreau and Ebenhoh (1994) explored this analytically using a model where two organisms feed on the same resource in each respective stage. They concluded that coexistence in this way was possible. The evolution of

complete metamorphosis has been argued to contribute to the vast diversity among Hexapoda (Rainford et al., 2014), which gives empirical support for the idea that ontogenetic niche shifts can promote species richness. Others have suggested that the evolution of an ontogenetic niche shifts can allow species with an overall weaker competitive ability to coexist with stronger competitors (Bassar et al., 2017). This was explored empirically by Anaya-Rojas et al. (2023) who compared the level of ontogenetic niche shift in killifish and guppies and found partial support for Bassar et al.'s theory.

One of the basic principles of **species coexistence** has long been that intraspecific competition must be greater than interspecific competition. This means that a species must depress its own growth more than others when increasing in density. Chesson (2000) more concretely split the factors promoting species coexistence into equalizing and stabilizing ones. Equalizing factors facilitates coexistence by making species more equal competitors which makes the effect of competition slower. But, stabilizing factors are required to actually make coexistence possible in the long run. A possible stabilizing factor in organisms with an ontogenetic niche shift would be a species mainly being limited by the resource in the stage for which it is a worse competitor.

While discussing coexistence it almost seems impossible not to stumble upon the topic of limiting similarity. The idea is that there is a limit to how similar species that coexist can be. This topic has been revisited many times it is now widely agreed upon that the region of coexistence in parameter space decreases with decreasing niche separation (Abrams, 1983; Meszéna et al., 2006; Abrams and Rueffler, 2009). This means, if species have smaller niches, more species can coexist. This naturally leads to the question if small niche widths or large niche widths are more favourable. Ackermann and Doebeli (2004) created a model where species could evolve a narrower or wider niche width. They made it so that the cost for a species to widen its niche width was a decrease in efficiency for the resources that it could consume. Their results showed that if the total rate of resource uptake is reduced by widening the niche it becomes a detriment for a species to evolve it. While this might not seem too surprising, it exemplifies the point that a species will generally evolve as wide a niche width as "possible" as long as the increased types of resources that it can consume are not less than the loss of the amount of resources it already could consume.

A recent model by Saltini et al. (2023) compared the number of coexisting species between simple- or complex organisms when communities resulted from either gradual evolution or immigration. They show that while complex life cycles could lead to higher species diversity, it did not do so through gradual evolution unless at least one of the stages had a narrower niche width than the simple organism. Thus, complex life cycles did not inherently lead to more coexisting species. While there were more species that could coexist within a "complex" community, these could not be reached when phenotypic change is constrained to be small. If immigration of species with arbitrary phenotypes was used instead, higher species diversity could be reached in communities comprised of complex organisms. It was only when individuals who invaded could be vastly different from individuals already present in the population that the entire trait space could be filled.

Another recent model by Vasconcelos et al. (2022) explored evolutionary branching of a consumer with a complex life cycle where each life stage can feed on two resources. They found that life cycle complexity can favour diversification and that a trimorphic community could be reached when resources where asymmetrically distributed. A trimorphic community is one where three species coexist. Essentially, they reached similar conclusions as Saltini et al. when using a symmetric resource distribution, in their model this represented only having two species in a community. However, the result found under asymmetry did not corroborate Saltini et al. (2023) results, instead they implied that complex life cycle can inherently lead to higher species diversity when resources where no longer symmetric. A lot of research, both theoretical and empirical, has been done since Wilbur published his seminal article exploring possible effects of having a complex life cycle. However, these questions are yet to be fully explored. Therefore, exploring the phenomena of ontogenetic niche shifts is still important to gain a deeper understanding of the mechanisms behind community assembly.

Aims and Hypothesis

In this thesis, I explore how communities of complex organisms whose life stages can evolve independently differ from communities of simple organisms. I revisit the model by Saltini et al. but make several changes that will allow me to investigate arbitrary resource distributions, including asymmetric and multi-modal distributions. My goal is to investigate whether the results of Saltini et al. are robust when no longer constrained to only

using a Gaussian resource distribution and under slightly different mathematical assumptions. Specifically, my model differs in the following ways: (i) discrete time consumer population dynamics instead of continuous time population dynamics. (ii) No logistic resource growth for the resource. Instead, a resource growth model by Schmid et al. (2024) where resources are replenished at the beginning of each season is used. The resources are then distributed among the consumer proportional to their feeding efficiency relative to the feeding efficiency of all individuals in the population. These changes greatly facilitate the numerical analysis of the model.

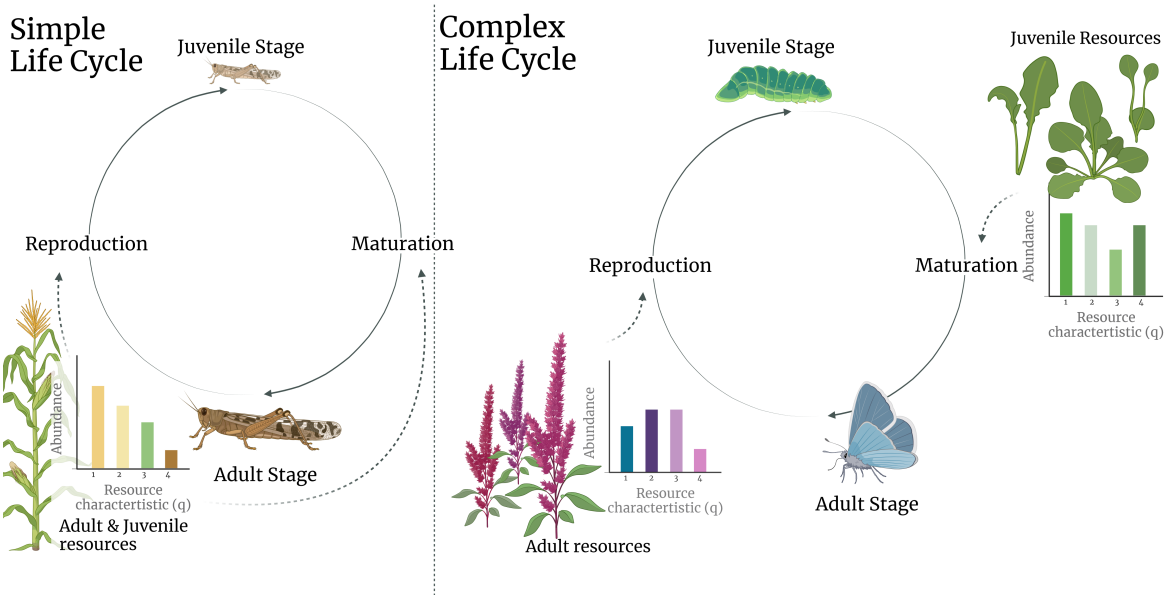


Figure 1: An illustration of the simple life cycle and the complex life cycle. Organisms with a simple life cycle have one set of resources which it can specialize in, while organisms with a complex life cycle can specialize both as a juvenile and as an adult. All resources within the model have a discrete distribution of resource characteristics (q), e.g., seed size or flower shape. An individual's ability to consume a resource depends on how close its trait value z matches the resource characteristic q and how abundant the resource is. For simple life cycles, the amount of resource consumed is linked to both reproduction and maturation of an individual. For complex life cycles, an individual's maturation is linked to the amount of juvenile resource consumed, while the reproduction is linked to the amount of adult resource consumed. The pictures are examples of the type of organisms that have these types of life cycle.

Methods

The Model

I compare community assembly between simple- and complex life cycles and therefore create two equivalent models. The difference between them is that in the complex model individuals within the population are be characterized by an adult trait, influencing fecundity, and a juvenile trait, influencing maturation. In the simple model, there is only one trait that influences both fecundity and maturation. Reproduction in my model is clonal, so there are no hybrids or individuals with intermediate trait values. However, many species of interest are diploid and sexually reproducing. In this paper I refer to individuals that differ in one or both of their trait values as different species. This is a simplification because true speciation also requires the evolution of assortative mating or reproductive isolation. Adaptive diversification is however an important piece of the puzzle and can promote assortative mating to evolve (Weissing et al., 2011; Dieckmann and Doebeli, 1999). To address how community assembly is affected by the life cycle of an organism, I investigate a discrete time model for consumers that live in a seasonal environment and compete for the same resources. Consumer individuals are assumed to complete their life cycle within a single year. Population census occurs just before reproduction of adults, and the population size of adults at time t is $A(t)$. All adult individuals die after reproduction. Reproduction results in $A(t)f$ juvenile individuals, where f denotes the expected number of offspring per individual. Juvenile individuals mature with probability m to become adults one time step later. Thus, the

population dynamics of adults of **one** species is given by

$$A(t+1) = A(t)fm. \tag{1}$$

The expected number of offspring and the probability to successfully mature depend on the amount of resource gathered by adult and juvenile individuals, respectively. Since adult and juvenile individuals with a complex life cycle consume different resources, there are two sets of independent discrete resources (figure 1). All resources are consumed within one time step and replenished at the beginning of the season. Within one time step, adults have offspring and then die, the juveniles then mature to the new generation of adults or die. To describe the dynamics of the resources, I closely follow Schmid et al. (2024).

Resource-consumption dynamics

Adult reproduction and maturation of juveniles depend on the amount of consumed resources at the specific life stage, which, in turn, depend on the amount of available resources and the feeding efficiency for those resources. The efficiency of individual i depends on two things: (1) the individual's trait values $z_{i,s}$, which are a quantitative foraging traits related to the resources, and (2) the characteristic of the resources (q_r). If an individual's trait value ($z_{i,s}$) is equal to a resource characteristic (q_r), then that individual is well adapted for that resource. Each individual in this model has two trait values, one associated to each life stage s, where s is either j or a for juvenile or adult, respectively. For a simple organism this trait value is the same in both life stages, which is equivalent with only having one trait value. I refer to consumers that differ in their trait values ($z_{i,s}$), either as an adult or juvenile, as different species. Each resource (r) is characterized by a resource characteristic (q_r). There are a total of T different resources, so r can take on natural numbers between 1 and T. Both the adult and juvenile resources have a discrete resource characteristic distributions. This allows me to approximate a Gaussian distribution but also implement arbitrary distributions. I assume that the feeding efficiency $\alpha(z_{i,s}, q_r)$ of an individual in a given stage with trait value $z_{i,s}$ on a resource with trait value q_r is monotonically decreasing with increasing distance between $z_{i,s}$ and q_r. Specifically, I use the Gaussian function

$$\alpha(z_{i,s}, q_r) = \frac{1}{\sqrt{2\pi\sigma_s^2}} e^{-\frac{(z_{i,s}-q_r)^2}{2\sigma_s^2}} \tag{2}$$

to describe this relationship. Here, σ_s determines how fast the feeding efficiency decreases with increasing mismatch between $z(z_{i,s}$ and q_r. It can be seen as the extent of generalism held by the consumer, or its niche width. If σ_s is large, individuals can forage efficiently on a broad set of resource. The opposite is true if σ_s is small, then consumers should be considered specialists. Individuals in each life stage feed on different resources, so $\alpha(z_{i,s}, q_r)$ is not only different for each resource but also each life stage. This model takes both density dependence and frequency dependence into account by making an individual's realized feeding efficiency be affected by the feeding efficiency of all other individuals. Simply put, an individual's realized ability to consume a resource is lower if there are many other individuals feeding on the same resource, especially, if they are adapted to feed on that resource. So, the amount of resources obtained depends on the individual's ability relative to the ability of all other individuals in the population. To this end, the feeding efficiency $\alpha(z_{i,s}, q_r)$ of an individual is divided by the summed feeding efficiencies of all other individuals,

$$\frac{\alpha(z_{i,s}, q_r)}{\sum_{n=1}^{N_s} \alpha(z_{n,s}, q_r)}. \tag{3}$$

Here, N_s denotes the total number of individuals in the population in the same life stage. Equation (3) describes the <u>fraction</u> of a resource that **one** individual in **one** life stage consumes of **one** resource. To get the total energy consumed from all resources by an individual during one life stage, equation (3) is summed over all resources T and adjusted for the amount available of each resource. The energy accumulated by an individual during one life stage is

$$E(z_{i,s}, \mathbf{z}_s) = \gamma \sum_{r=1}^{T} R_r \frac{\alpha(z_{i,s}, q_r)}{\sum_{n=1}^{N_s} \alpha(z_{n,s}, q_r)}, \tag{4}$$

where $\mathbf{z}_s$ is a vector with all different individuals traits within the life stage, $\mathbf{z}_s = z_{i,s}, ..., z_{N_s,s}$. Equation (4) describes the total amount of energy consumed by an individual during one life stage. $E(z_{i,s}, \mathbf{z}_s)$ is connected to an individual's stage specific trait value ($z_{i,s}$) as well as all other individuals' trait values ($\mathbf{z}_s$). R_r is the initial abundance of resource r, which is assumed to be depleted during one time step. Schmid et al. (2024) derive equation (4) in detail and show that all resources are depleted as time goes to infinity. The constant γ represents energy content received by a consumer per unit of resource, which for simplicity is equal to one. At this point, recall that in the case of the complex model, each life stage has a separate energy consumption value $E(z_{i,s}, \mathbf{z})$. And these are based on the traits and resources distinct to each life stage. An individual's total energy consumption will therefore depend on its life stage (s) as well. For simple organisms this value will be the same, but for complex organisms it can be different. The reproduction of adults (f) and survival or maturation of juveniles (m) is monotonously increasing with $E(z_{i,s}, \mathbf{z})$. In this model their relationship to energy consumption is also saturating, meaning the relative benefit received from energy acquired will be less and less,

$$f = f_{\max} \frac{E(z_{i,a}, \mathbf{z})}{k_f + E_a}, \tag{5a}$$

$$m = \frac{E(z_{i,j}, \mathbf{z})}{k_m + E_j}. \tag{5b}$$

The half-saturation constants k_f and k_m determine how quickly the functions saturate. The maximum number of offspring an individual can have is $f_{\max}$. This means that the maximum value of equation (5a) is $f_{\max}$. The maximum value of equation (5b) is 1, since maturation, and by extension survival, is a probability. See table 1 for an overview and explanation of all variable, parameters and subscripts.

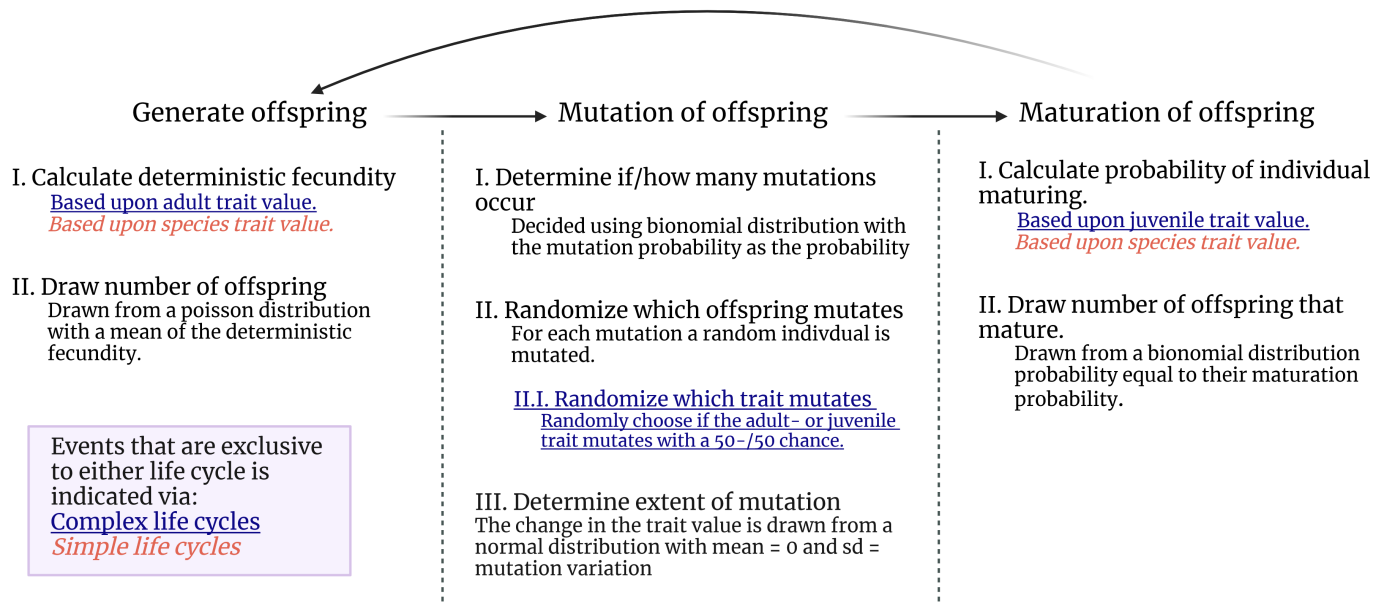


Figure 2: A visual summary of the simulation algorithm.

Model Analysis

This model is analysed with individual-based computer simulations. Since even genetically identical individuals generally do not always have the same number of offspring or equal survival, I include stochasticity in maturation and fecundity. This is done by drawing the number of offspring and maturation from probability distributions. The fecundity value f calculated in equation (5a) is used as the mean in a Poisson distribution. A Poisson distribution describes the probability of observing a certain number of events when the expected number of events is specified in a given time (Otto and Day, 2007). The "expected number of events" is the mean number of offspring an individual produces in one time step and the "certain number of events" the realized number of

offspring. The actual number of offspring surviving and maturing into adults within a time step is drawn from a binomial distribution. Essentially, there is a coin flip made for each individual with a weighted coin. The chance of survival is m and the risk of death is $1 - m$. Survival is the same as maturing, as all individuals who do not mature within one time step die.

The aim of this paper is to study how community assembly is affected by complex life cycles. This is done with the expectation that two independently evolving traits might increase the number of species that can evolve and coexist within a community (Wilbur, 1980; Saltini et al., 2023). To study how gradual evolution of complex organisms compared to simple organisms might differ, organisms need to be able to change within the model. Each newborn individual can mutate with probability μ. Which offspring, and in the case of complex organisms, which trait is mutated, is determined randomly by drawing from a binomial distribution. The actual change in trait is then drawn from a normal distribution with a mean of 0 and a standard deviation of δ. Both μ and δ can be varied to represent different speeds and constraints on mutational change.

These simulations almost inevitably results in having clusters of very similar species at the end of the simulation. Under gradual evolution we expect a discrete set of species at the end of a simulation. However, since each species continuously mutates, we observe a cluster of types that is maintained by a balance of mutation and stabilizing selection. Since we only want to count one species per cluster we use the following algorithm at the end of each simulation. I measure the Euclidean distance between species, any that are less than 0.15 units apart are grouped together into one species. The trait values of the most abundant of those species characterizes the cluster. Only after this, do I calculate the number of species. In addition, I want to remove any species that have just been added, as they have yet to undergo selection during any extended period of time. This is mainly a problem when analysing communities built up through immigration, when there are new individuals of random phenotypes added every generation. During simulations using gradual evolution any species that have just been added will be removed with the grouping algorithm. So, any species with an abundance lower than 0.05% of the total abundance of all species were removed at the last time step.

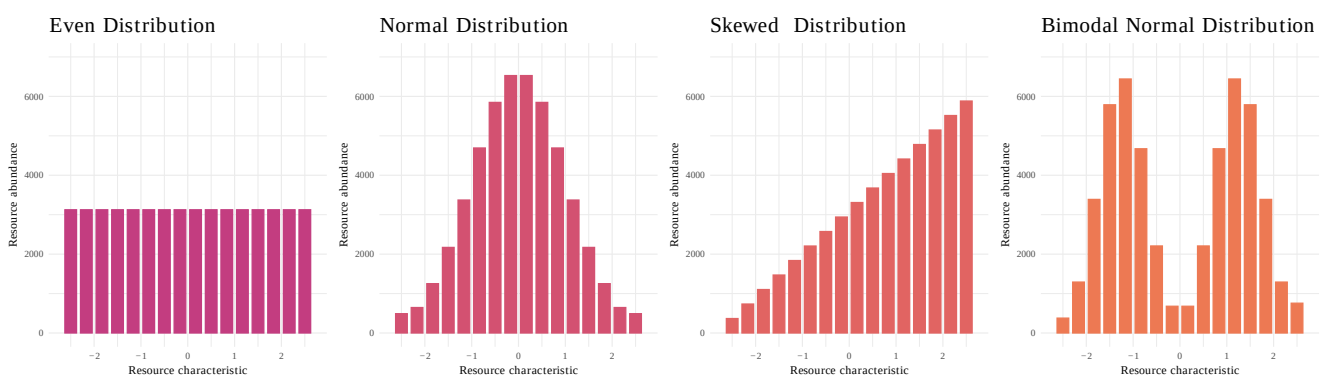


Figure 3: The four different resource distributions used in the model analysis. From left to right, even, normal, skewed and bimodal normal. There are 16 types of resources and their total abundance 50 000. The approximated normal distribution has a mean of 0 and standard deviation of 1. The approximated bimodal normal distribution consists of two normal distributions have a mean of -1.25/1.25 and a standard deviation of 0.5. The bimodal normal distribution is halved when frequency of resources are calculated, so the total area under the two curves is still equal to 1.

By varying the resource characteristic and resource distributions along with generalism of individuals σ_s, I explore how communities emerge and how the endpoint of a community depends on the different parameters used in the model. To get a visual overview and a summary over the model analysis see figure 2. The effect of life cycle complexity on community assembly was explored by comparing the number of species that evolved to coexist gradually within a community of simple organisms contra a community of complex organisms. Furthermore, the position these species occupied within the niche space was analysed. The simulations were run for 50 000 generations. This number was a compromise between having a long enough simulation so that a form of equilibrium could be reached and the time the simulations take. I note that due to stochastic nature of the model the number of species and population size did undergo some fluctuations even after this point (figures12 and 11). The number of species was calculated at the last time step after grouping algorithm was used. Communities built up through immigration were explored by removing mutation from the model.

Instead, one individual with random phenotypes between -3 and 3 was added every generation. This is akin to allowing mutations to be of arbitrary size. At the end of the simulations the same grouping and removal of low abundance species that were done during gradual evolution simulations was performed. It is important to note that the timescales between immigration and gradual evolution are not comparable, as the mutation rate and immigration rate are not realistically standardized against each other. Therefore, there are no simulations where both mutation and immigration are used simultaneously.

In order not to confuse effects from differing resource dynamics between stages with the effects caused by having either of the two life history strategies, as many parameters as possible are equal between adults and juveniles (i.e. $k_f = k_m, \sigma_a = \sigma_j$). The only difference between stages is the parameter $f_{\max}$. Maturation and fecundity are saturating functions of the amount of energy consumed but m can take values only between 0 to 1, whereas f can take values between 0 and $f_{\max}$. However, this is still equal between the simple and complex model. Resource distributions for both life cycles and both stages are also identical. The total amount of resources available are equal for simple and complex organisms and set so that the total population of all species in the adult stage is high ($\approx$ 30 000). Each simulation was run with a set of standard conditions unless otherwise specified. The standard conditions are: Both adults and juveniles have an identical resource distribution of 16 resources for which the characteristic q_r goes from -2.5 to 2.5 with a total abundance of all resources combined is 50 000. I explore four different resource distributions, namely even, discrete normal, skewed, and bimodal (see figure 3). The parameters used for simulations are: $\mu = 10^{-5}$, $\delta = 0.05$, $k_A = 0.5$, $k_J = 0.5$ and initial phenotype for both sexes is 0. All simulations were performed using R Statistical Software (R Core Team, 2023, version 4.3.2). If you wish to replicate my analysis the code used can be found in the supplementary Information.

Results

Community richness and niche partitioning in simple and complex organisms

I explore community assembly through gradual evolution by varying niche width (σ_s) of consumers and compare the number of species that evolved to coexist within a community for complex and simple organisms. Lower niche width should result in more coexisting species (Abrams, 1983; Meszéna et al., 2006; Abrams and Rueffler, 2009). Simulations were run 10 times for each combination of σ_s-values and the mean number of species at the end of the simulations was used in the comparison.The results of these simulations are summarized in figure 4.The way my model is constructed a maturation limited community is always formed, because there are always many more juveniles than adults. A maturation limited community is one where the juveniles are experiencing more competition, so the juvenile trait undergoes more phenotypic divergence. This is especially obvious under gradual evolution when σ_s is low and the two species only differentiate in the juvenile trait (figure 9). As a consequence the niche width of the juvenile stage is more of a limiting factor in the number of species that evolved to coexist for complex organisms.

In figure 4 the mean number of species when niche width between complex life cycles ($\sigma_{adult} = \sigma_{juvenile}$) and simple life cycles is shown. It was: equal higher for complex life cycles when niche width was small ($\sigma = 0.05$) and equal or very similar when niche width was moderate or large ($0.2 < \sigma$). There is some difference between different resource distributions, but the general pattern is the same for them all (figure 4).

Some other σ_s values were explored and what could be seen is that when σ_s is below ≈ 0.1 species with a simple life cycle are specialized for one resource each, which is the maximum possible number of species for simple organisms. For my resource distributions this is equal to 16. Species with complex life cycle can reach almost double the amount of species coexisting. When looking at figure 4 one can observe that if at least one of the species have a sufficiently low niche width ($\sigma_s = 0.05$) then the other stages σ_s value does not impact the number of coexisting species. The resulting community that occurs at these low σ_s values can be seen in figure 7 (C). If the initial phenotype in the simulation is at the centre of the resource distribution a community shaped like a cross is often formed. If a different starting phenotype is used the shape differs, but the result is the same, the species arrange themselves along two axes, where approximately half the species share the same juvenile trait and the other half share the same adult trait. This holds true for all four resource distributions. However, this type of community is not the only possible result of simulations at $\sigma_s = 0.05$ therefore standard deviation from

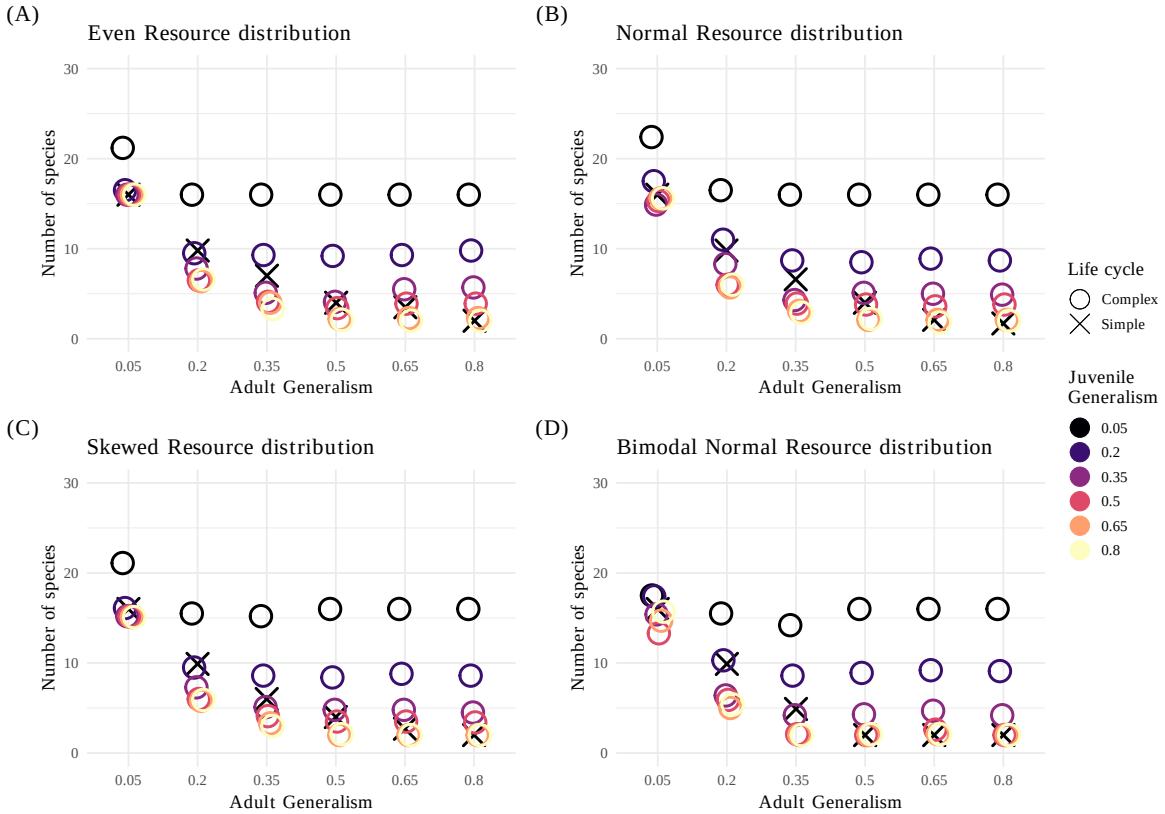


Figure 4: The mean number of coexisting species over ten simulations for a community built up through gradual evolution. The simulations were done for six different σ_s values $(0.05, 0.2, 0.35, 0.5, 0.65, 0.8)$, resulting in respectively 6 and 36 different simulations for simple- and complex life cycles. The type of life cycle is indicated via shape and the juvenile niche width with color. Four different resource distributions were simulated: (A) Even resource distribution, (B) Normal Resource distribution, (C) Skewed Resource distribution, (D) and a Bimodal Normal Resource distribution. Standard parameters were used.

the mean in is much higher(not presented). The alternative to the two axes community is one where each species is arranged in such a way that each juvenile trait value is paired with only one adult trait value. In other words, no two species are specialized for the same set of resources, either as juveniles or as adults. This is sometimes a diagonal but can take on other shapes (figure 6). The latter community is equivalent to the community formed by simple life cycles as the species can be arranged on one axis, and has the same or similar number of species. When the "two axes" pattern was discovered I decided to investigate how mutation probability affects community-wide niche partitioning. With low mutation probability, the resulting communities are almost all arranged along a single axis (figure 7 (A)), while with high mutation probability communities arranged along two axes dominate (figure 7 (C)). At intermediate mutation probabilities there is also a possibility of something in between the two extremes arising, a mixed community, where some of the niche space displays characteristics of the two axes community, while other parts look more like the single axis community (figure 7 (B)). With a mutation probability $\mu = 10^{-5}$, a mutant appears on average every third generation. By lowering μ to $\mu = 10^{-6}$, a mutant appears on average every 30th generation. Hence, essentially only one mutant is invading at a time if the mutation probability is sufficiently low. At higher mutation probability, several possible mutant is in-

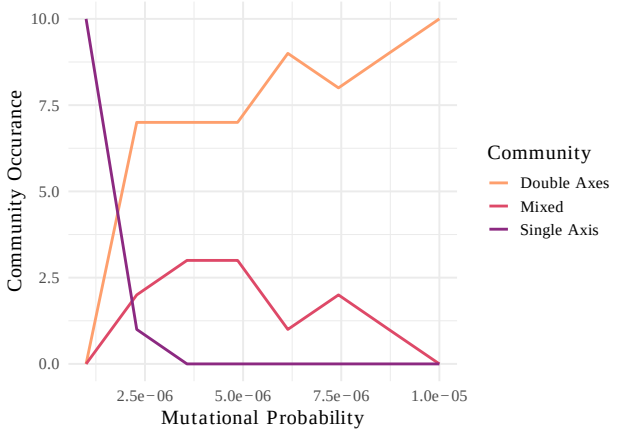


Figure 5: The number of times the three different communities occurred during 10 simulations for eight different mutation probabilities. An even resource distribution was used and $\sigma = 0.05$

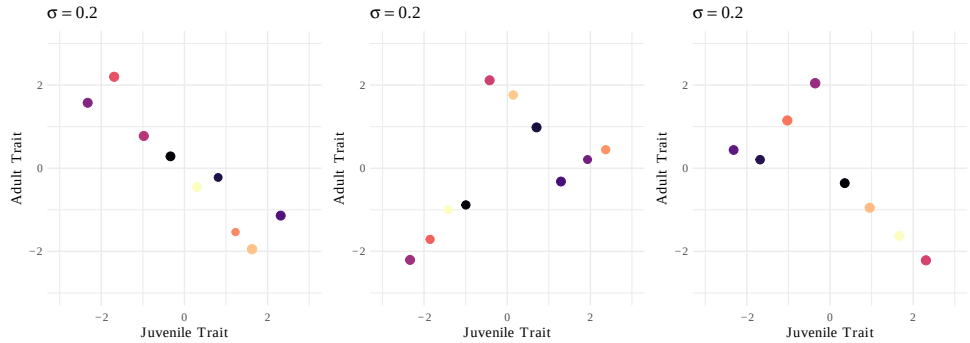


Figure 6: Example of three possible versions of the single axis community, where each species is specialized for one set of resources. Simulations where run with an even resource distribution with standard parameters and $\sigma = 0.2$. The colours of the dots differentiate species and the size of a dot corresponds to the abundance of that species.

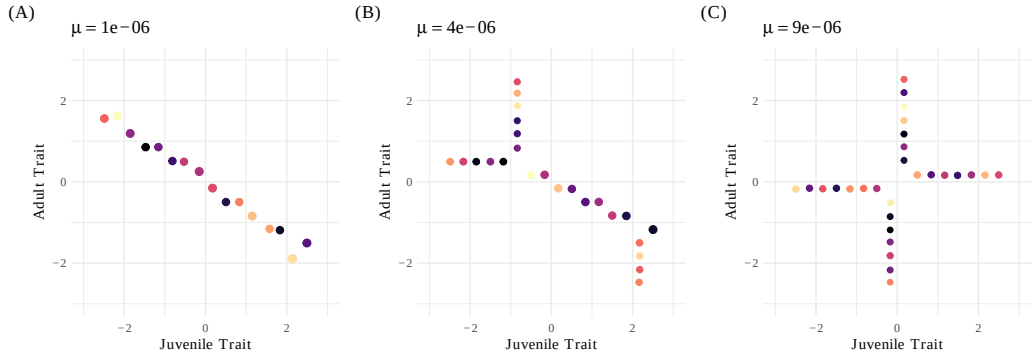


Figure 7: Examples of the three different communities that occurred when σ was low for different mutation probabilities μ. (A) Single axis community, (B) Mixed community, (C) Double axes community. An even resource distribution was used. $\sigma = 0.05$. The colours of the dots differentiate species and the size of a dot corresponds to the abundance of that species.

vading at the same time. This difference is what I believe cause the two different patterns. I believe that the simultaneous or near-simultaneous occurrence of mutants that are located horizontally and vertically from the resident species prevent the establishment of mutants that are positioned on the diagonal. The mutant trying to establish on the diagonal would be competing with both the vertical and horizontal mutant. The chance that a single/mixed/two axes community is formed in my simulation is related to the mutational probability (figure 5). With a single axis community only being formed at very low μ-values and the two axes community becoming more and more pervasive at higher mutation probabilities.

Community assembly through immigration

When studying the effect of immigration, mutation was removed from the model and an individual with randomly chosen trait values was added each generation. By removing the constraints of gradual evolution a more saturated community could be reached with more of the niche space being filled for complex organisms (figure 9). When comparing the equivalent species between simple and complex communities (i.e. $\sigma_a = \sigma_j$ and both life cycle types) the complex community always had more species, for most resource distributions around at least 1.5x more species (figure 8). The number of species for complex communities remains higher compared to simple communities even when comparing niche widths that are wider. For example, when complex communities have $\sigma = 0.2$ they have on average 20 species coexisting while simple communities with $\sigma = 0.05$ have 16 species coexisting.

For simple organisms the niche space that can be filled is one-dimensional, and the maximum number of

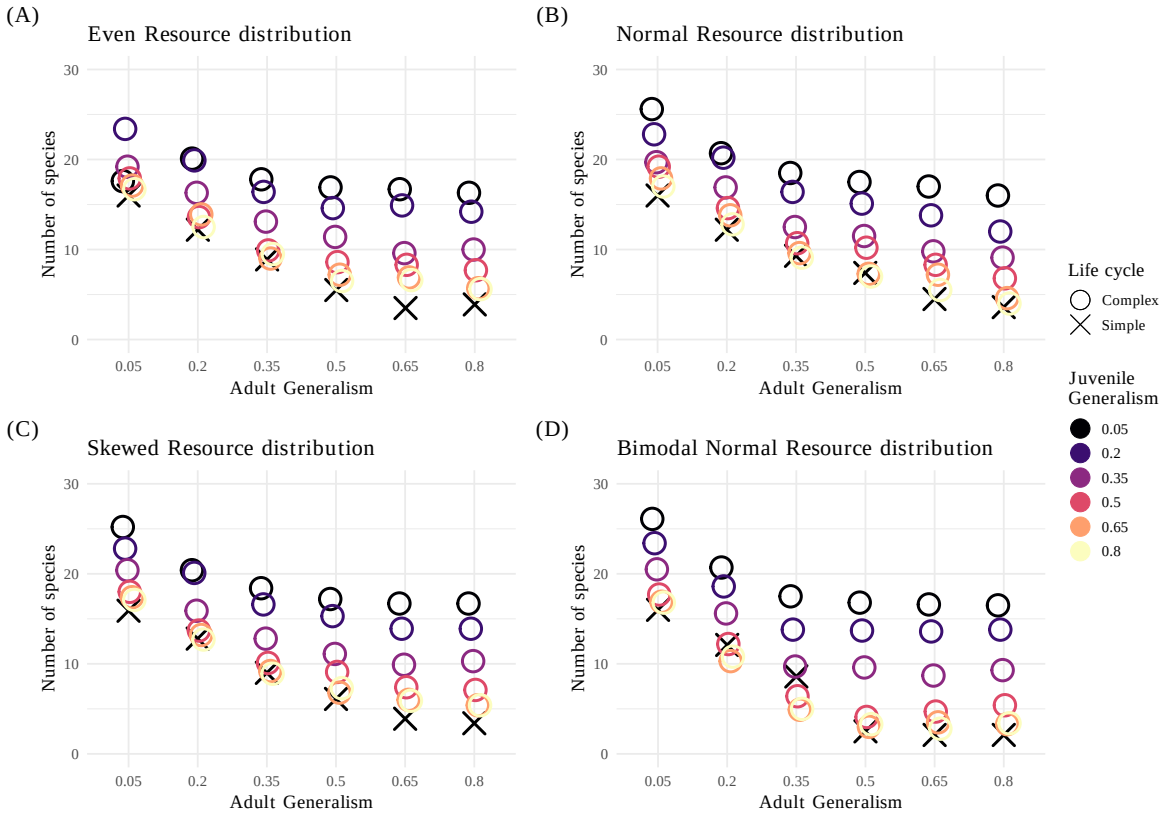


Figure 8: The mean number of coexisting species over ten simulations for a community built up through immigration. The simulations were done for six different σ_s values $(0.05, 0.2, 0.35, 0.5, 0.65, 0.8)$, resulting in 6 and 36 different simulations, respectively, for simple- and complex life cycles. The type of life cycle is indicated via shape and the juvenile niche width with colour. Four different resource distributions were used: (A) Even resource distribution, (B) Normal Resource distribution, (C) Skewed Resource distribution, (D) and a Bimodal Normal Resource distribution. Standard parameters were used.

species reached is the number of resources, i.e. 16. Complex organisms have a two-dimensional niche space, which means there is more space for species to coexist. The exact placement of the species within this space is not uniform in my model (figure 9). I believe this is mainly due to the limitation that the abundance of resources put on the number of species that can coexist. Populations do not persist if they are not abundant enough because of drift. Generally the more species there are, the less abundant they become, because the combined abundance of all species within my model is fairly constant and depends on the total abundance of resources. Therefore, while there might be places in niche space species can still occupy, they cannot occupy these because the maximum number of species have been reached. Nevertheless, it is clear that communities built up through immigration are more species rich for complex life cycles. Not only is the simple community less species rich than the complex community during immigration simulations but the mean number of species in the immigration community is higher compared to the number of species that evolved to coexist under gradual evolution. Another difference between the immigration and gradual evolution community assembly is the adult trait is more phenotypically diverged under immigration than it was under gradual evolution (figure 9. The community is not **as** affected by being maturation limited.

Effect of Resource Distribution

While there was no significant difference between the number of coexisting species for different resource distributions, there were slight differences in the partitioning, visualized in figure 10. The patterns are quite intuitive. Species are generally more closely clustered around the peak of the resource distribution, and species closer to the peak have a higher abundance. The even resource distribution causes species arrange themselves so that no two species compete for the same set of resources. The normal resource distribution causes species to cluster closer to the centre of the resource distribution at $q_R = 0$. The skewed resource distribution causes species to

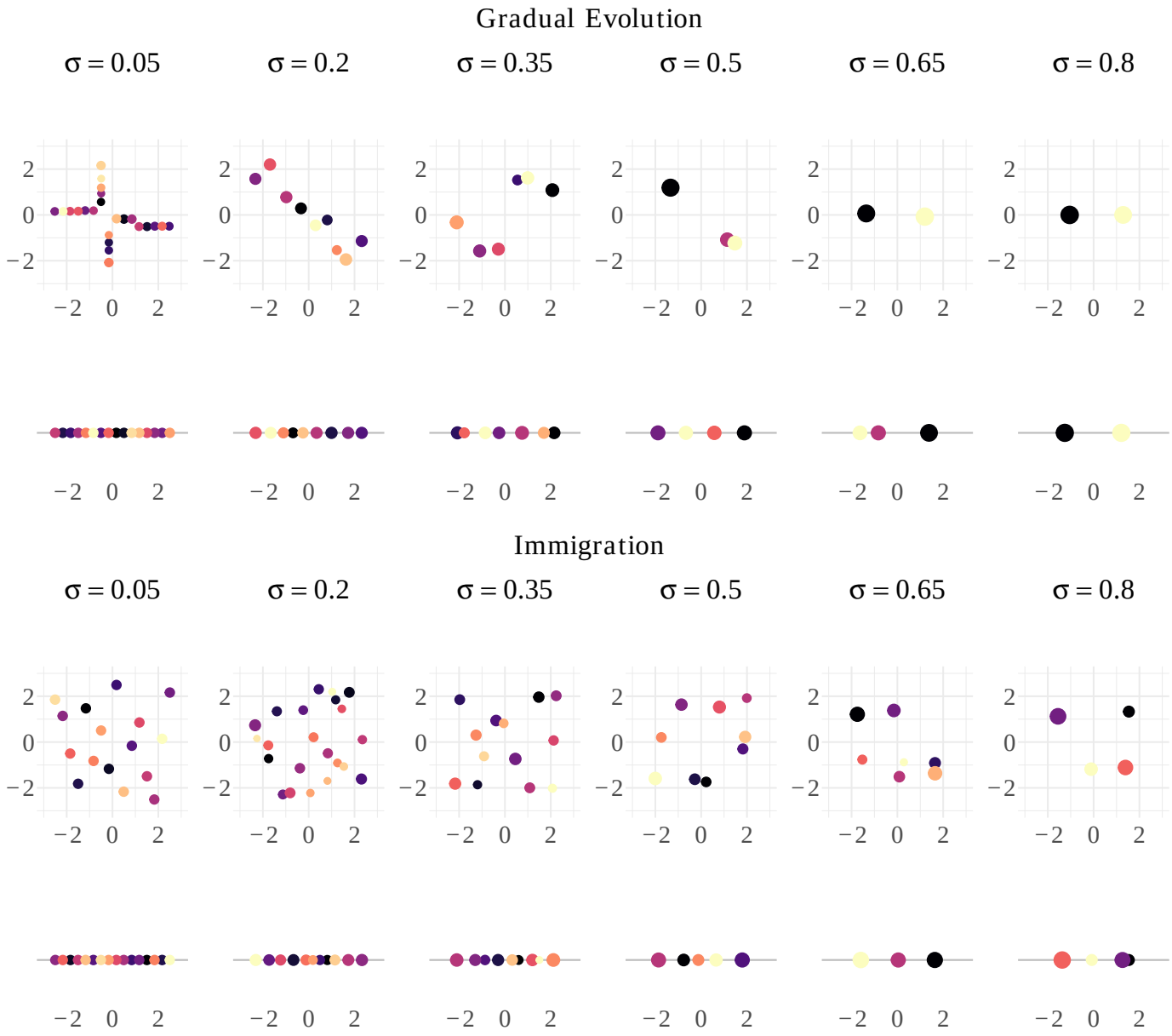


Figure 9: A phase-plane plot showing the phenotypes of species at the end of a simulation for six different σ-values for both gradual evolution and immigration. The top rows are simulations for species with a complex life cycle and the bottom rows simulations for species with a simple life cycle. For complex life cycles the x-axis is juvenile trait and the y-axis adult trait. Standard parameters and an even resource distribution were used. The colours of the dots differentiate species and the size of a dot corresponds to the abundance of that species.

cluster around the peak of $q_r = 2.5$ and avoid the area where there are the least resources $q_r = -2.5$. For the bimodal resource distribution there are two different peaks, and therefore two clusters centred on them (figure 10). At $\sigma_s = 0.05$ and high mutation probability the two axes community appeared regardless of resource distribution.

Trimorphic Community in a two resource system.

To compare the results of this model with that of Vasconcelos et al. (2022), simulations where consumers had two resources available were done. Only even and skewed resource distributions were used in these simulations, in other words, a symmetric and an asymmetric resource distribution, with resources characteristics equal to -1 and 1. The resource abundances were either 25 000 for both (even) or 10 000 and 40 000 (skewed). A trimorphic community was sometimes reached for skewed resource distributions, but never for even resource

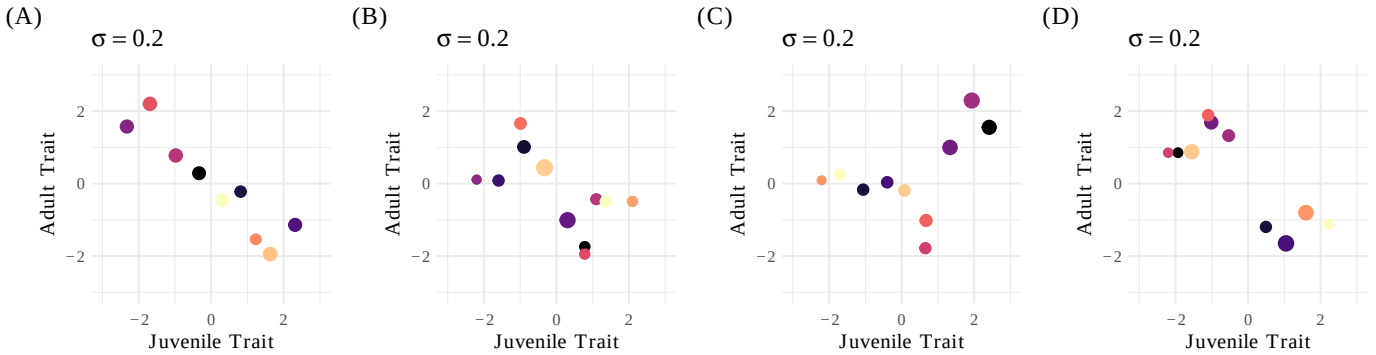


Figure 10: Final community composition under gradual evolution for the four different resource distributions. (A) Even resource distribution, (B) Normal Resource distribution, (C) Skewed Resource distribution, (D) and a Bimodal Normal Resource distribution. Standard parameters where used and $\sigma = 0.2$. The colours of the dots differentiate species and the size of a dot corresponds to the abundance of that species.

distributions. There was a lack of consistency in when the trimorphic community arose, and it was therefore hard to explore for which σ_s-values it was most common. This led me to also explore a simulation where four species were placed in each corner of the phenotype space without the possibility of mutations arising. Standard resource distribution where used for these simulations. They revealed that a skewed distribution supported a trimorphic community at higher niche widths than any other distribution. Even resource distributions supported a trimorphic community until $\sigma = 0.35$, and normal and bimodal normal resource distributions until $\sigma = 0.20$. Skewed resource distributions supported the trimorphism at even the highest niche width explored in these simulations, $\sigma = 0.8$.

Discussion

I used an individual based model to explore if community assembly of species with an ontogenetic niche shift could allow more species to coexist. To this end I simulate the creation of communities under both gradual evolution and immigration with different resource distributions, niche widths and mutation probabilities.

One of the reasons I created this model was to see if the results reached by Saltini et al. (2023) changed if looking at other resource distributions. The hypothesis is that a skewed resource distribution might change the results, similarly to how they did in the model by Vasconcelos et al. (2022). Using my model, I could look at the effect of different resource distributions. I explored even-, normal-, skewed- and bimodal normal-resource distributions. While there were some differences between both the number of species that evolved to coexist and niche partitioning, these differences were slight and very similar patterns arose across all the different type of distributions. The differences were also unsurprising, for example a skewed resource distribution having a higher proportion of the species within the niche space where the resources are more abundant. So using different resource distributions I could reach the same conclusions as Saltini et al.. Some of the major conclusions reached by Saltini et al. (2023) is that if mutations are constrained to have a small phenotypic effect, a "diagonal community" is formed, that is, a community where each species competes for a different set of resources. I refer to this as a single axis community, as it is equivalent to the community reached by the simple life cycle when comparing the number of species found. Their results indicate that complex life cycles only lead to higher species richness if the community is built up from sequential immigrations. Essentially, communities with complex life cycles could not reach higher species richness under gradual evolution, because they were "trapped" in the diagonal community, surrounded by fitness valleys, resulting in any nearby mutants

having a negative invasion fitness. For an in-depth explanation consult Saltini et al. (2023).

The differences in our results instead arose when looking at the effect of mutational rate. My model showed that having a complex life cycle could inherently lead to more species evolving under gradual evolution when niche width was small. However, the pervasiveness of this among the simulations depended on the mutation probability. In Saltini et al.'s model and most other similar models exploring this subject there is a separation between ecological and evolutionary time scales. This means that the population dynamics reach an equilibrium before a new mutant is added. This is one of the core assumptions of adaptive dynamics, which is a mathematical framework often used to study evolutionary diversification of this type analytically (Geritz et al., 1998). The separation of the evolutionary and ecological is not inherently present in my model. While something similar can be achieved by lowering the mutation probability, there is still the chance of several mutants establishing at the same time. I believe the major differences found in my results compared to Saltini et al. (2023) stems from the fact that several mutants can invade at the same time. If there are mutants invading vertically and horizontally from the established species, they hinder a mutant on the diagonal from invading, even though this mutant actually competes less with the established species. During simulations in my model the chance of a diagonal community being formed is related to the mutation probability as well as the value of σ_s. The combination of low niche width and high mutation probability lead to what I refer to as two axes community. Where half of the species compete for the same juvenile resource and the other half for the same adult resource, causing two axes to form along niche space. The actual shape of this depended on the initial phenotypes, but similar patterns were found for different initial phenotypes.

By simulating immigration rather than gradual evolution communities with complex life cycles attained higher species richness compared to communities with simple life cycles. This mirrors the result found by Saltini et al. (2023). The only difference found in the uniformity of the resulting communities placement within niche space, with my results not being as uniform. I attribute this slight difference to the stochasticity in my model. Immigration also leads to higher species richness when comparing it to complex communities resulting from gradual evolution.

I explored the difference between an even resource distribution and a skewed resource distribution more closely in a model where each life stage feed on resource distribution with only two resource types to compare the results to those found by Vasconcelos et al. (2022). These results seem to match each other as a trimorphic community could both be sustained at higher niche widths in a static community without mutation, and arose more often through gradual evolution for a skewed resource distribution. Vasconcelos et al. (2022) found that the asymmetry in resources allowed for one species that fed on a less abundant resource as a juvenile could make up for this growth by feeding on a more abundant resource as an adult, as opposed to only having two species, each being specialized for the same resource at both stages. However, the skewed resource distribution did not markedly change the results found when looking at communities that arose under gradual evolution or immigration when there were more than two resources.

Can complex life cycles result in more species coexisting in a community?

Results from this study substantiate earlier studies showing that complex life cycles can lead to higher species richness under certain conditions (Saltini et al., 2023). But one of those conditions, the separation of the evolutionary and ecological timescale, has not been previously explored.

The results from this model suggest that complex life cycles do allow more species to coexist within a community when either (1) one of the stages has a lower niche width than the species with a simple life cycle, or (2) a community is built of through sequential immigrations, i.e., evolution is not constrained to be gradual, or (3) the mutation probability is large and niche width is small. Condition (1) and (2) where also found by Saltini et al. (2023), while condition (3) is novel. There are two facts to question with this condition: Is a very small σ realistic? and, is the relaxation between the separation of ecological and evolutionary time realistic? To address the first question one must first consider what small niche width implies. Species that have a small niche width are generally thought of as specialized and highly efficient on one or a few types of resources. Ackermann and Doebeli (2004) created a model looking at the evolution of niche width and found that species would generally evolve to widen their niche width as long as the total rate of resource uptake was not lowered. From a biological standpoint having an extremely narrow niche would often times result in the evolution of a wider niche width

being favoured. Unless widening once niche results in a net loss of resources. As the values of the parameters in my model are arbitrary it is of course hard to tell if $\sigma = 0.05$ is actually realistic. However, there are definitely cases of highly specialized species thriving in nature. So the next question is: is it plausible that two different mutants with functional differences are invading a population at the same time? I certainly think this could be the case for large populations and this is something that should be explored further. One of the reasons most models choose to separate evolutionary and ecological time scales is because it is difficult to analytically study a system where they are not. However, if there is an impact on the results it is important to challenge that assumption.

Another big assumption in my model is that phenotypic divergence results in separate species. In sexually reproducing organism adaptive diversification is hindered by random mating. There are many possible outcomes of phenotypic divergence in a sexually reproducing organism, of which sympatric speciation is only one (Rueffler et al., 2006). Nevertheless, previous explorations on the subject have shown that selection for different environments or niches is one of the forces that can promote the evolution of assortative mating and speciation (Butlin et al., 2009; Weissing et al., 2011). Therefore, the results of this model are still relevant for sexually reproducing organisms, as they might inform us on one of the mechanisms that results in sympatric speciation.

Conclusion

The results of this model mainly mimic the ones reached by Saltini et al. (2023), the study that inspired this thesis. Generally, the number of species that evolve to coexist within a community of complex species will not be higher than for simple species under gradual evolution, unless at least one of the life stages has a lower niche width than the species with a simple life cycle. When comparing communities that are built up through immigration, complex communities can have many more coexisting species even for higher niche widths than the simple communities. The novel conclusion of my study is that complex life cycles can inherently lead to higher species richness even through gradual evolution. This happens when the separation of ecological and evolutionary time is relaxed. When the mutation probability is high, almost twice as many species can evolve in complex communities compared to simple communities. Most models exploring gradual evolution assume there is a separation between ecological and evolutionary time-scales, so the knowledge that this could affect results is important to consider when drawing conclusions from these.

Acknowledgements

I would like to offer thanks to all my friends who kept my spirits high and let me vent when me and my thesis where on the outs. I would also like to thank our hacky sack group which probably did more for my good mood than anything else during the last part of the year. Lastly, I would like to thank my supervisor Claus Rüffler for his immense support during my time writing this thesis.

Model Variables, Parameters, Subscripts, and their Meanings

Variables	
A(t)	Number of adults at time t.
$\alpha(z_{i,s}, q_r)$	The feeding efficiency of an individual of a given life stage for a specific resource.
$E_s(z_i, \mathbf{z})$	Total amount of energy consumed by an individual consumer during one life stage.
Parameters	
f	The number of offspring per adult, i.e. fecundity.
m	The probability that a juvenile will mature into an adult.
z	Represents the trait value quantitatively. In this model an individual can have several trait values associated with it. The subscripts are used to clarify which individual the trait value is associated with and which life stage or resource character it corresponds to.
q	Characteristic of a resource. This is compared against an individual's trait value to determine how well individuals can forage on this resource.
σ	Represents the extent of generalism held by the consumer. This can be varied between life stages.
T	Total number of resources.
$\mathbf{z}$	Does not have an explicit number of value associated with it, but indicates that a variable is affected by all individuals' trait value.
γ	Energy content received by consumer per unit of resource.
R_r	Initial abundance of a resource.
N	The total number of individuals within the population.
f_{max}	The maximum number of offspring per individual.
k_m, k_f	Half saturation constants. Determines how energy content received by consumers and fecundity/maturation are related to each other.
δ	The amount of change that a mutation can create in a trait.
μ	The chance of a mutation happening in an individual.
Subscripts	
r	Used to indicate which resource is being considered.
i	As each individual is considered separately in this model this subscript is used to indicate an individual.
s	This subscript is used to determine which stage is being referred to for an individual. It can be either juvenile or adult.

Table 1: Table of the variables, parameters and subscripts used in this paper and their associated meaning.

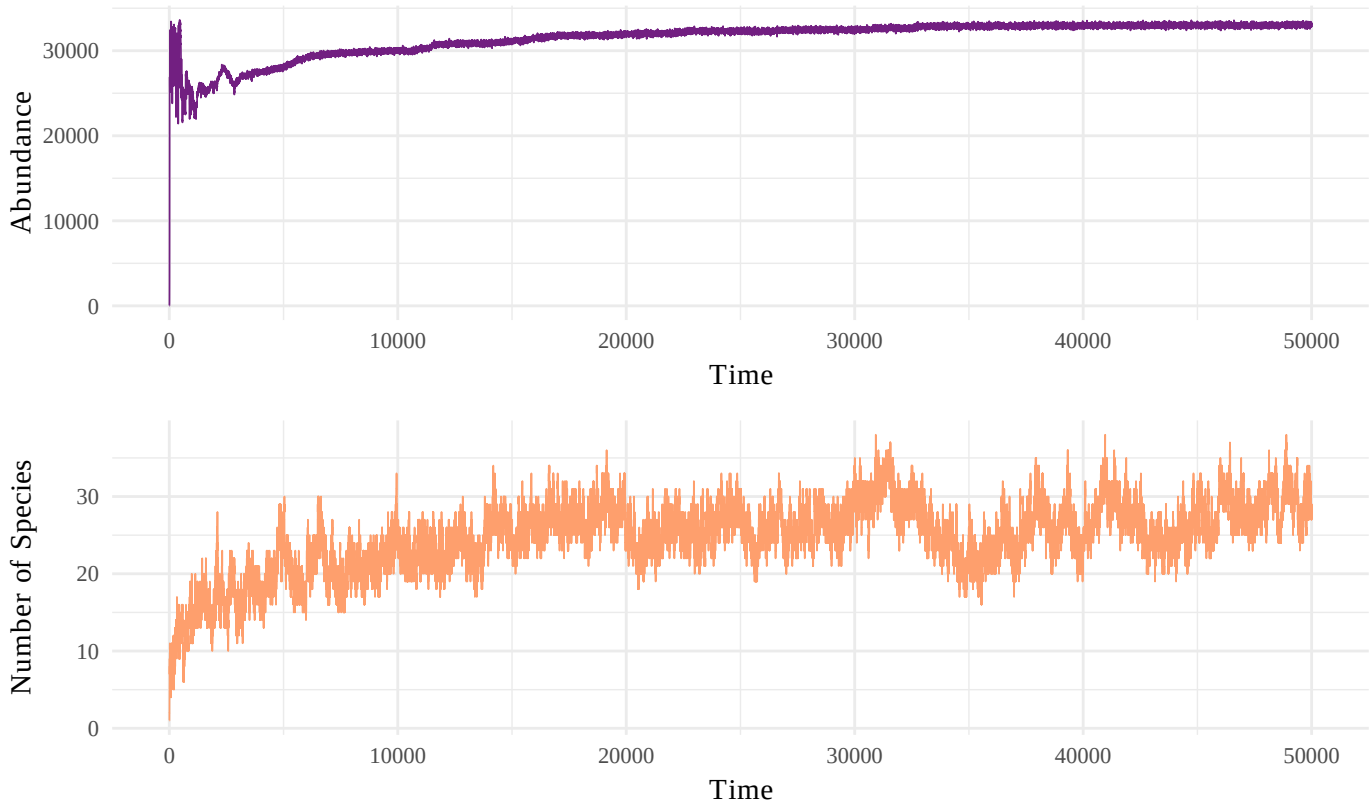


Figure 11: Abundance of all species combined as well as the number of species over time.

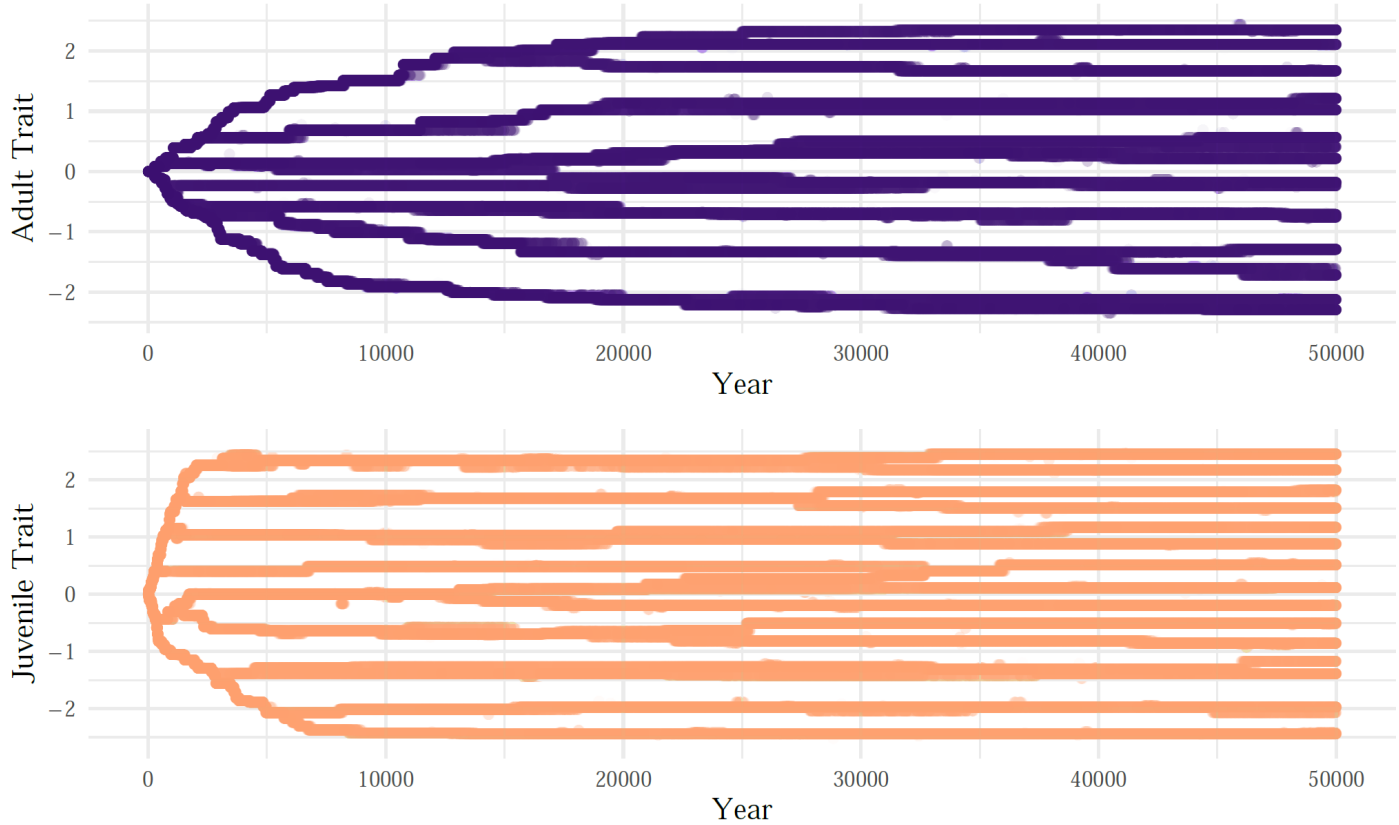


Figure 12: Simulated phenotypic divergence starting from 10 individuals of one species with $\sigma = 0.05$ and standard parameters over the course of 50000 generations. The transparency of colour corresponds to the number of individuals.

References

Abrams, P. 1983. The Theory of Limiting Similarity. Annual Review of Ecology and Systematics 14:359–376.

Abrams, P. A., and C. Rueffler. 2009. Coexistence and limiting similarity of consumer species competing for a linear array of resources. Ecology 90:812–822.

Ackermann, M., and M. Doebeli. 2004. Evolution of niche width and adaptive diversification. Evolution 58:2599–2612.

Anaya-Rojas, J. M., R. D. Bassar, B. Matthews, J. F. Goldberg, L. King, D. Reznick, and J. Travis. 2023. Does the evolution of ontogenetic niche shifts favour species coexistence? An empirical test in Trinidadian streams. Journal of Animal Ecology 92:1601–1612.

Bassar, R. D., J. Travis, and T. Coulson. 2017. Predicting coexistence in species with continuous ontogenetic niche shifts and competitive asymmetry. Ecology 98:2823–2836.

Butlin, R., J. Bridle, and D. Schluter. 2009. Speciation and Patterns of Diversity. Cambridge University Press, Cambridge, United Kingdom.

Chesson, P. 2000. Mechanisms of Maintenance of Species Diversity. Annual Review of Ecology and Systematics 31:343–366.

Dieckmann, U., and M. Doebeli. 1999. On the origin of species by sympatric speciation. Nature 400:354–357.

Geritz, S. A. H., E. Kisdi, G. Meszéna, and J. A. J. Metz. 1998. Evolutionarily singular strategies and the adaptive growth and branching of the evolutionary tree. Evolutionary ecology 12:35–57.

Loreau, M., and W. Ebenhoh. 1994. Competitive Exclusion and Coexistence of Species with Complex Life Cycles. Theoretical Population Biology 46:58–77.

Meszéna, G., M. Gyllenberg, L. Pásztor, and J. A. J. Metz. 2006. Competitive exclusion and limiting similarity: A unified theory. Theoretical Population Biology 69:68–87.

Miller, T. E., and V. H. Rudolf. 2011. Thinking inside the box: community-level consequences of stage-structured populations. Trends in Ecology & Evolution 26:457–466.

Mougi, A. 2017. Persistence of a stage-structured food-web. Scientific Reports 7:11055.

Otto, S. P., and T. Day. 2007. A biologist's guide to mathematical modeling in ecology and evolution. Princeton University Press, Princeton.

R Core Team. 2023. R: A Language and Environment for Statistical Computing. R Foundation for Statistical Computing, Vienna, Austria.

Rainford, J. L., M. Hofreiter, D. B. Nicholson, and P. J. Mayhew. 2014. Phylogenetic Distribution of Extant Richness Suggests Metamorphosis Is a Key Innovation Driving Diversification in Insects. PLoS ONE 9:e109085.

Rudolf, V. H. W., and K. D. Lafferty. 2011. Stage structure alters how complexity affects stability of ecological networks. Ecology Letters 14:75–79.

Rueffler, C., T. J. M. Van Dooren, O. Leimar, and P. A. Abrams. 2006. Disruptive selection and then what? Trends in Ecology & Evolution 21:238–245.

Saltini, M., P. Vasconcelos, and C. Rueffler. 2023. Complex life cycles drive community assembly through immigration and adaptive diversification. Ecology letters 26:1084–1094.

Schmid, M., C. Rueffler, L. Lehmann, and C. Mullon. 2024. Resource variation within and between patches: Where exploitation competition, local adaptation and kin selection meet. The American Naturalist 203:E19–E34.

Schreiber, S., and V. H. W. Rudolf. 2008. Crossing habitat boundaries: coupling dynamics of ecosystems through complex life cycles. Ecology Letters 11:576–587.

Sherratt, E., M. Vidal-García, M. Anstis, and J. S. Keogh. 2017. Adult frogs and tadpoles have different macroevolutionary patterns across the Australian continent. Nature ecology & evolution 1:1385–1391.

Vasconcelos, P., M. Saltini, and C. Rueffler. 2022. Consequences of life-cycle complexity to the potential for evolutionary branching.

Weissing, F. J., P. Edelaar, and G. S. van Doorn. 2011. Adaptive speciation theory: a conceptual review. Behavioral Ecology and Sociobiology 65:461–480.

Werner, E. E. 1988. Size, Scaling, and the Evolution of Complex Life Cycles. Pages 60–81 *in* B. Ebenman and L. Persson, eds. Size-Structured Populations. Springer, Berlin, Heidelberg.

Werner, E. E., and J. F. Gilliam. 1984. The Ontogenetic Niche and Species Interactions in Size-Structured Populations. Annual review of ecology and systematics 15:393–425.

Wilbur, H. M. 1980. Complex Life Cycles. Annual review of ecology and systematics 11:67–93.

Supplementary Information

Relaxation of the separation between ecological and evolutionary timescales has unexpected effects on community assembly of species with complex life cycles.

Izabel Eriksson

R-Code

The code is divided into five parts for easier handling. The first two parts are the functions used during simulations. The third part is how I define my resource distributions and parameters and the fourth part and example simulation. The last part is how I would plot the simulation example. I run simulations in many different iterations during my thesis, with different parameters and replicates and different plotting techniques. If you are interested in recreating something that is not in the simulation example please consult my thesis for the different change needed in resource distribution or parameter values.

I ran these simulations using R studio in R Statistical Software (R Core Team, 2023, version 4.3.2). The packages used where:

```
library(job)
library(tidyverse)
library(viridisLite)
library(viridis)
library(ggplot2)
library(extrafont)
library(patchwork)
library(dplyr)
library(FamilyRank)
```

(Lindeløv, 2021; Garnier et al., 2023; Wickham, 2016; Wickham et al., 2019; Wickham, 2016; Chang, 2023; Wickham et al., 2023; Saul, 2021; Pedersen, 2024).

Resource competition functions

This is the the actual simulation algorithm where competition and evolution occurs.

```
# Resource competition Models:

# Simple life cycle -------------


resourceCompetitionSLC <- function(popSize, resProp, resFreq, resGen=matrix(
   c(0.2,0.2),ncol=1, nrow=2), im = 0,
                                      fmax = 2, kA = 0.5, kJ = 0.5, mutProb
                                         =0.0005, mutVar=0.05, time.steps
                                         =50000, iniP=0,
                                      threshold = 0.0005, nmorphs = 1, maxTr =
                                         3, minTr = -3){
```

```
pop <- matrix(data = NA, ncol = 3, nrow = nmorphs)
                              # Each column in this matrix is one species.

pop[,1] <- popSize
                                                                  # Number of
   individuals per species is first row.
pop[,2] <- iniP
                                                                    #
   Phenotype for species is second row.


colnames(pop) <- c("Number␣of␣indivduals", "Trait", "Proxy")
                     # Third row is where f and m are stored to calculate
   fecundity and maturation.

stats         <- matrix(data = c(0, sum(pop[,1]), 0, nrow(pop), mean(pop
   [,2]), var(pop[,2]))
                        , nrow = 1, ncol = 6)
                                                          # Where we will
                            eventually save our data on number of species
                           etc
phenotypes <- matrix(data = c(0, popSize, iniP), nrow = 1, ncol = 3)
            # Where information on each different species is stored
colnames(phenotypes) <- c("Year", "Number␣of␣indivduals", "Trait")

epsilon <- .Machine$double.eps^10
                                                 # Added when there is risk
   of r rounding a number down to 0, very small number

for (t in 1:time.steps){

  # Deterministic fecundity proxy alpha
     ------------------------------------

  adults    <- pop
                                                                  # The
     adults matrix is only created so it easier to mentally seperate adult
      and juvenile step, also if the adult population wants to be
     extracted at end of timestep
  alphaA    <- NULL
                                                                 # Will
     create a matrix with all alpha values, is nullified so previous time
     step is overwritten
  alphaSumA <- NULL


  resPropAduMatrix <- matrix(data = resProp, ncol = length(resProp),
             # Resource property is made into a matrix here so that
     matrix calculatsion can be used.
                              nrow = nrow(adults), byrow = T)
  aduTrait <- adults[, 2]
```

```
aduTraitMatrix <- matrix(data = rep(aduTrait, each = length(resProp)),
        # Made into matrix so matrix calculcations can be used
                        ncol = length(resProp), nrow = nrow(adults),
                        byrow = T)

alphaA           <- (1/(sqrt(2*pi*resGen[1,1]^2)))*exp(-(((
   aduTraitMatrix-resPropAduMatrix)^2)/(2*resGen[1,1])^2)) + epsilon
                  # Calculation of individual alpha values, equation 2
adultAbund       <- adults[,1]
adultAbundMatrix <- matrix(data = rep(adultAbund, each = length(resProp)
   ), ncol = length(resProp), nrow = nrow(adults), byrow = T)  #
   Creation of a matrix with population size of each type in the rows
alphaSumA        <- colSums((alphaA*adultAbundMatrix))
                       # Creates the denominator for equation 3. Which
   is each INDIVIDUALS alphaA added together for each resource type, so
   each column is one resource, each row is one species.


RdivAlphaSumA       <- resFreq/alphaSumA
RdivAlphaSumATrans  <- matrix(data = RdivAlphaSumA)
                          # Is transformed so that matrix caluclations
   can be done

Fec <- alphaA%*%RdivAlphaSumATrans
                                             # The result is a vector
   with each species total energy consumption, equation 4.
adults[,3] <- fmax*(Fec/(kA+Fec))
                                              # Gives f (fecundity)
   equation 5a



# Spawning of offspring
   ------------------------------------------------

juveniles <- adults
                                                            # Create a
   matrix were we will add juveniles into

juveniles[,1] <- rpois(n = nrow(juveniles),
                                   # The number of juveniles is drawn
   from a poisson distribution with mean f x species abundance
                       lambda = juveniles[,1]*juveniles[,3])
                                     # Since each individual of a
                          species has offspring

juvenile.pop <- c()
juvenile.pop <- sum(juveniles[,1])
                                             # To extract number of
   juveniles if that wants to be analysed
```

```
# Mutation of offspring
   ---------------------------------------------------

probs <- juveniles[,1]/sum(juveniles[, 1])    # Generates probability of
    morph being mutated based upon number of individuals.
N.mut <- as.numeric(rbinom(n = 1, size = sum(juveniles[, 1]), prob =
   mutProb)) # Draws the number of mutations this generation


if(N.mut > 0){
  random.choice <- c()
  mutation.pos <- c()

  for (m in 1:length(N.mut)){
    random.choice <- rmultinom(n = 1 , size = 1, prob = probs)
                    # Randomly chooses which morphs to mutate based on
       probs
    mutation.pos[m] <- as.numeric(which(random.choice == 1))
  }

  for (i in  1:length(mutation.pos)){

    mutChange <- rnorm(n=1, mean=0, sd=mutVar)
    juveniles[mutation.pos[i], 1] <- juveniles[mutation.pos[i], 1] - 1
             # Removes the mutated individual from the morph

    new.morph <- matrix(data = c(1, juveniles[mutation.pos[i], 2] +
       mutChange, 0),  #Changes trait to a new trait and adds it to the
       juveniles
                        ncol = ncol(juveniles), nrow = 1)
    juveniles <- rbind(juveniles, new.morph)

  }
}



# Maturation off offspring
   ------------------------------------------------------

# The calculation of alphaJ is the same as the caluclation of alphaA see
    above comments for clarifications

alphaSumJ <- NULL
alphaJ    <- NULL

resPropJuvMatrix <- matrix(data = resProp, ncol = length(resProp), nrow
   = nrow(juveniles), byrow = T)
juvTrait <- juveniles[,2]
juvTraitMatrix <- matrix(data = rep(juvTrait, each = length(resProp)),
   ncol = length(resProp), nrow = nrow(juveniles), byrow = T)
```

```
alphaJ            <- (1/(sqrt(2*pi*resGen[2,1]^2)))*exp(-(((
   juvTraitMatrix-resPropJuvMatrix)^2)/(2*resGen[2,1]^2))) + epsilon
juvenAbund        <- juveniles[,1]
juvenAbundMatrix  <- matrix(data = rep(juvenAbund, each = length(resProp
   )), ncol = length(resProp), nrow = nrow(juveniles), byrow = T)
alphaSumJ         <- colSums(alphaJ*juvenAbundMatrix)


RdivAlphaSumJ     <- resFreq/alphaSumJ
RdivAlphaSumJTrans   <- matrix(data = RdivAlphaSumJ)


Sur <- alphaJ%*%RdivAlphaSumJTrans
juveniles[,3] <- (Sur/(kJ+Sur))
                                                # Gives m, equation 5b




juveniles[,1] <- rbinom(n = nrow(juveniles) , size = juveniles[,1], prob
    = juveniles[,3])


pop <- juveniles[juveniles[, 1] != 0, , drop = FALSE]
                           # all adults die after reproducing, so the new
    generation is only juveniles, and all rows with zero individuals are
    removed.

# Adding immigrants
   ---------------------------------------------------------------------

# Only done when im = 1
if(im == 1) {

    trait <- runif(1, min = minTr, max = maxTr)
                                    # Creates phenotype of immigrant

    if(sum(pop[,2] == trait) == 0) {
                                                # Checks whether a exact
       match of immigrant already exists
      pop <- rbind(pop, c(1, trait, NA))
                                              # If no duplicate is found,
         a new species is added to the pop matrix
    } else{
      same <- which(pop[,2] == trait)
      pop[same,1] <- pop[same,1]+1
                                                   # If so it is just
         added to that species
    }
  }
```

```
  # extract stats and phenotype
     ---------------------------------------------

  if(nrow(pop) == 0){
                                                                  # Checks
     whether population has reached zero, then it breaks the for loop.
    print("Population␣extinction")
    break
  }

  if(sum(which(is.na(pop[,1])) != 0)){
                                                # Checks whether population
     has reached zero, then it breaks the for loop.
    print("Population␣extinction")
    break
  }

  # Extracts the results from this generation into the stats and
     phenotypes matrices.

    stats <- rbind(stats, c(t, sum(adults[,1]), juvenile.pop, nrow(pop),
       mean(pop[,2]), var(pop[,2])))

    pStats <- cbind(rep(t, nrow(pop)), pop[,1], pop[,2])
    phenotypes <- rbind(phenotypes, pStats)


}

# Removing last time step

stats <- stats[stats[, 1] != time.steps, , drop = FALSE]
phenotypes <- phenotypes[phenotypes[,1] != time.steps, , drop = FALSE]

# Removing any morphs of very low abundance for last time step
pop <- pop[pop[, 1] > threshold*stats[nrow(stats), 2], , drop = FALSE]


# Re-adding modified last time step

stats <- rbind(stats, c(t, sum(adults[,1]), juvenile.pop, nrow(pop), mean(
   pop[,2]), var(pop[,2])))

pStats <- cbind(rep(t, nrow(pop)), pop[,1], pop[,2])
phenotypes <- rbind(phenotypes, pStats)

#return output
   ------------------------------------------------------------
colnames(stats) <- c("year", "Adult␣Population␣size", "Juvenile␣Population
   ␣Size", "Number␣of␣morphs", "mean␣trait", "var␣trait")
rownames(phenotypes) <- NULL
```

```
  return(list(stats=stats, phenotypes=phenotypes))
                                    #returns both the stats and the
     phenotype


}



# Complex life cycle -------------------------

# The simple life cycle function is more thoroughly commented, the same
   methods are used here, please consult above for explanations.
# Only code that differs is explained here.

resourceCompetitionCLC <- function(popSize, resProp, resFreq, resGen=matrix(
   c(0.2,0.2),ncol=1, nrow=2), fmax = 2,
                                  kA = 0.5, kJ = 0.5,mutProb=0.0005, mutVar
                                     =0.05, time.steps=50000, iniPA=0,
                                     iniPJ=0,
                                  threshold = 0.0005, nmorphs = 1, im = 0,
                                     maxTr = 3, minTr = -3){


  pop <- matrix(data = NA, ncol = 4, nrow = nmorphs)
                                # Each column in this matrix is one
     phenotype combination.

  pop[,1] <- popSize
  pop[,2] <- iniPA
                                                                    # Adult
     trait is stored in second row and
  pop[,3] <- iniPJ
                                                                    # juvenile
      trait in third row.

  colnames(pop) <- c("Number␣of␣indivduals", "Adult␣trait", "Juvenile␣trait"
     , "Proxy")

  stats         <- matrix(data = c(0, sum(pop[,1]), 0, nrow(pop), mean(pop
     [,2]), var(pop[,2]),
                                  mean(pop[,3]), var(pop[,3])), nrow = 1,
                                     ncol = 8)

                                     #Where we will eventually save our
                                     stats and phenotypes
  phenotypes <- matrix(data = c(0, popSize, iniPA, iniPJ), nrow = 1, ncol =
     4)
  colnames(phenotypes) <- c("Year", "Number␣of␣indivduals", "Adult␣trait", "
     Juvenile␣trait")
```

```
epsilon <- .Machine$double.eps^10
                                                    # Added when some number
   become zero, very small number



for (t in 1:time.steps){
  # Deterministic fecundity proxy alpha
     -----------------------------------

  adults     <- pop
  alphaA     <- NULL
                                                                  # Will
     create a matrix with all alpha values
  alphaSumA  <- NULL


  resPropAduMatrix <- matrix(data = resProp[1,], ncol = ncol(resProp),
     nrow = nrow(adults), byrow = T)
  aduTrait <- adults[, 2]
  aduTraitMatrix <- matrix(data = rep(aduTrait, each = ncol(resProp)),
     ncol = ncol(resProp), nrow = nrow(adults), byrow = T)

  alphaA           <- (1/(sqrt(2*pi*resGen[1,1]^2)))*exp(-(((
     aduTraitMatrix-resPropAduMatrix)^2)/(2*resGen[1,1])^2)) + epsilon
                      # Calculation of individual alpha
  adultAbund       <- adults[,1]
  adultAbundMatrix <- matrix(data = rep(adultAbund, each = ncol(resProp)),
      ncol = ncol(resProp), nrow = nrow(adults), byrow = T)  # Creation of
      a matrix with population size of each type in the rows
  alphaSumA        <- colSums((alphaA*adultAbundMatrix))

     # Creation of matrix that reflects both the trait but also number of
     individuals in type


  RdivAlphaSumA       <- resFreq[1,]/alphaSumA
  RdivAlphaSumATrans  <- matrix(data = RdivAlphaSumA)

  Fec <- alphaA%*%RdivAlphaSumATrans
  adults[,4] <- fmax*(Fec/(kA+Fec))

  # Spawning of offspring
     ---------------------------------------------------

  juveniles <- adults
                                                                # Create a
     matrix were we will add juveniles into

  juveniles[,1] <- rpois(n = nrow(juveniles), lambda = juveniles[,1]*
     juveniles[,4])
```

```
juvenile.pop <- c()
juvenile.pop <- sum(juveniles[,1])   # To extract number of juveniles


# Mutation of offspring
   ------------------------------------------------

probs <- juveniles[,1]/sum(juveniles[, 1])    # Generates probability of
    morph being mutated based upon number of individuals.
N.mut <- as.numeric(rbinom(n = 1, size = sum(juveniles[, 1]), prob =
   mutProb))


if(N.mut > 0){
  random.choice <- c()
  mutation.pos <- c()

  for (m in 1:length(N.mut)){
    random.choice <- rmultinom(n = 1 , size = 1, prob = probs)
                    # Randomly chooses which morphs to mutate based on
       probs
    mutation.pos[m] <- as.numeric(which(random.choice == 1))
  }

  for (i in  1:length(mutation.pos)){

    mutChange <- rnorm(n=1, mean=0, sd=mutVar)
    juveniles[mutation.pos[i], 1] <- juveniles[mutation.pos[i], 1] - 1
            # Removes the mutated individual from the morph

    if(rbinom(n = 1, size = 1, prob = 0.5) == 0){
                                # Randomly choose whether adult or
       juvenile trait gets morphed.


      new.morph <- matrix(data = c(1, juveniles[mutation.pos[i], 2] +
         mutChange, #Changes adult trait to a new trait and adds it to
         the juveniles
                                juveniles[mutation.pos[i], 3], 0),
                        ncol = ncol(juveniles), nrow = 1)
      juveniles <- rbind(juveniles, new.morph)
    }
    else {

      new.morph <- matrix(data = c(1, juveniles[mutation.pos[i], 2] ,
                #Changes juvenile trait to a new trait and adds it to
         the juveniles
                                juveniles[mutation.pos[i], 3]+
                                   mutChange, 0),
                        ncol = ncol(juveniles), nrow = 1)
      juveniles <- rbind(juveniles, new.morph)
```

```
    }

  }

}



# Maturation off offspring
   ----------------------------------------------------

alphaSumJ <- NULL
alphaJ    <- NULL
# Survival of juveniles also depends on resource availability
resPropJuvMatrix <- matrix(data = resProp[2,], ncol = ncol(resProp),
   nrow = nrow(juveniles), byrow = T)
juvTrait <- juveniles[,3]
juvTraitMatrix <- matrix(data = rep(juvTrait, each = ncol(resProp)),
   ncol = ncol(resProp), nrow = nrow(juveniles), byrow = T)

alphaJ            <- (1/(sqrt(2*pi*resGen[2,1]^2)))*exp(-(((
   juvTraitMatrix-resPropJuvMatrix)^2)/(2*resGen[2,1]^2))) + epsilon
juvenAbund        <- juveniles[,1]
juvenAbundMatrix  <- matrix(data = rep(juvenAbund, each = ncol(resProp))
   , ncol = ncol(resProp), nrow = nrow(juveniles), byrow = T)
alphaSumJ         <- colSums(alphaJ*juvenAbundMatrix)


RdivAlphaSumJ     <- resFreq[2,]/alphaSumJ
RdivAlphaSumJTrans   <- matrix(data = RdivAlphaSumJ)


Sur <- alphaJ%*%RdivAlphaSumJTrans
juveniles[,4] <- (Sur/(kJ+Sur))




juveniles[,1] <- rbinom(n = nrow(juveniles) , size = juveniles[,1], prob
    = juveniles[,4])

pop <- juveniles[juveniles[, 1] != 0, , drop = FALSE]
                          # all adults die after reproducing, so the new
    generation is only juveniles, and all rows with zero individuals are
    removed.

# Adding immigrants
   ----------------------------------------------------------------------



if(im == 1) {
```

```
    Atrait  <- runif(1, min = minTr, max = maxTr)
    Jtrait  <- runif(1, min = minTr, max = maxTr)

    if(sum(pop[,2] == Atrait & pop[,3] == Jtrait) == 0) {
                                # Checks whether a exact match of immigrant
       already exists, both traits are checked in complex
         pop <- rbind(pop, c(1, Atrait, Jtrait, NA))
        } else{
          same <- which(pop[,2] == Atrait & pop[,3] == Jtrait)
          pop[same,1] <- pop[same,1]+1
          }
    }




  # extract stats and phenotype
     ---------------------------------------------

  if(nrow(pop) == 0){
                                                                  # Checks
     whether population has reached zero, then it breaks the for loop.
    print("Population␣extinction")
    break
  }

  if(sum(which(is.na(pop[,1])) != 0)){
                                                # Checks whether population
     has reached zero, then it breaks the for loop.
    print("Population␣extinction")
    break
  }


  stats <- rbind(stats, c(t, sum(adults[,1]), juvenile.pop, nrow(pop),
     mean(pop[,2]), var(pop[,2]),
                                mean(pop[,3]), var(pop[,3])))

  pStats <- cbind(rep(t, nrow(pop)), pop[,1], pop[,2], pop[,3])
  phenotypes <- rbind(phenotypes, pStats)




}

#Removing last time step

stats <- stats[stats[, 1] != time.steps, , drop = FALSE]
phenotypes <- phenotypes[phenotypes[, 1] != time.steps, , drop = FALSE]
```

```
  # Removing any morphs of very low abundance for last time step
  pop <- pop[pop[, 1] > threshold*stats[nrow(stats), 2], , drop = FALSE]

  # Readding modified last time step
  stats <- rbind(stats, c(t, sum(adults[,1]), juvenile.pop, nrow(pop), mean(
     pop[,2]), var(pop[,2]),
                          mean(pop[,3]), var(pop[,3])))

  pStats <- cbind(rep(t, nrow(pop)), pop[,1], pop[,2], pop[,3])
  phenotypes <- rbind(phenotypes, pStats)

  #return output
     ---------------------------------------------------------------
  colnames(stats) <- c("year", "Adult␣population␣size","Juvenile␣Population␣
     Size", "Number␣of␣morphs", "mean␣A␣trait", "var␣A", "mean␣J␣trait", "
     var␣J")
  rownames(phenotypes) <- NULL

  return(list(stats=stats, phenotypes=phenotypes))   #returns both the stats
      and the phenotype


}
```

Grouping functions

```
# Grouping functions
# These functions are used at the end of a simulation to ensure
# very closely clustered species
# are grouped into one.

# SLC -------------------------


slc.groups <- function(output = outputSLC, threshold = 0.15){
  outputSLC <- output

  # Prepare the data

  phenodataSLC <- data.frame(
  Year = outputSLC$phenotypes[, 1],
  Trait = outputSLC$phenotypes[, 3],
  Num_Individuals = outputSLC$phenotypes[, 2]
      )

  last_year_dataSLC <- phenodataSLC[phenodataSLC$Year == max(phenodataSLC$
     Year), ]


  last_year_dataS <- subset(last_year_dataSLC, select = -Year)
  last_year_dataS <- subset(last_year_dataS, select = -Num_Individuals)
```

```
rownames(last_year_dataS) <- NULL
rownames(last_year_dataSLC) <- NULL

# Creates a matrix where each entry indicates distances between two
   species.

distance_matrix <- as.matrix(dist(last_year_dataS[, 1, drop = FALSE],
   method = "euclidean"))

# Removes duplicates in lower diagnoal

distance_matrix[lower.tri(distance_matrix)] <- NA


if(sum(which(distance_matrix < threshold)) == 0){

  return(last_year_dataSLC)
} # Checks if there are zero individuals who are alike, then the function
   is stopped.


# Find indices of individuals to keep
same <- which(distance_matrix < threshold, arr.ind = T) # Creates groups
   of species that are clustered togerther


same <- same[same[, 1]-same[,2] != 0, , drop = FALSE]   # Removes the rows
    indicating a species itself is to similar to itself.
rownames(same) <- NULL


if(sum(same) == 0){
  return(last_year_dataSLC)
}  # Checks if there are zero individuals who are alike, then the function
    is stopped.



# Initialize an empty list to store groups
groups <- list()

# Function to find group index for a species
find_group <- function(species_id) {
  for (g in seq_along(groups)) {
    if (species_id %in% unlist(groups[[g]])) {
      return(g)
    }
  }
  return(0)
}

# Iterate over rows in the matrix
```

```
  for (s in 1:nrow(same)) {
    species1 <- same[s, 1]
    species2 <- same[s, 2]

    # Find groups for each species
    group1 <- find_group(species1)
    group2 <- find_group(species2)

    if (group1 == 0 & group2 == 0) {
      # Create a new group
      groups <- c(groups, list(c(species1, species2)))
    } else if (group1 == 0) {
      # Add species1 to the group containing species2
      groups[[group2]] <- c(groups[[group2]], species1)
    } else if (group2 == 0) {
      # Add species2 to the group containing species1
      groups[[group1]] <- c(groups[[group1]], species2)
    } else if (group1 != group2) {
      # Merge two groups
      groups[[group1]] <- c(groups[[group1]], groups[[group2]])
      groups <- groups[-group2]
    }
  }




  # Filter out duplicate species in each group
  groups <- lapply(groups, function(group) unique(group))

  rownames(last_year_dataSLC) <- NULL
  final_data <- last_year_dataSLC         # Place to store filtered data
  total.sub <- c()                      # Place to store subspecies

  #Add population count of "subspecies" to main species

  for(q in seq_along(groups)){
    combo <- NULL
    combo <- groups[[q]]
    main <- combo[which.max(final_data[combo,3])]
    sub <- combo[-which.max(final_data[combo,3])]
    final_data[main,3] <- final_data[main,3] + sum(final_data[sub,3])
    total.sub <- rbind(c(total.sub, sub))

  }
  # Remove subspecies
  final_data <- final_data[-total.sub, ,drop = FALSE]
  return(final_data)


}
```

```
# CLC -------------------------------------------------------


clc.groups <- function(output = outputCLC, threshold = 0.15){

  outputCLC <- output

  # Prepare the data

  phenodataCLC <- data.frame(
    Year = outputCLC$phenotypes[, 1],
    Adult_Trait = outputCLC$phenotypes[, 3],
    Juvenile_Trait = outputCLC$phenotypes[, 4],
    Num_Individuals = outputCLC$phenotypes[, 2]
  )

  last_year_dataCLC <- phenodataCLC[phenodataCLC$Year == max(phenodataCLC$
     Year), ]
  last_year_dataC <- subset(last_year_dataCLC, select = -Year)
  last_year_dataC <- subset(last_year_dataC, select = -Num_Individuals)
  rownames(last_year_dataCLC) <- NULL
  rownames(last_year_dataC) <- NULL

  # Creates a matrix where each entry indicates distances between two
     species.

  distance_matrix_adult <- as.matrix(dist(last_year_dataC[, 1, drop = FALSE
     ], method = "euclidean"))
  distance_matrix_juvenile <- as.matrix(dist(last_year_dataC[, 2, drop =
     FALSE], method = "euclidean"))

  # The lower diagonal is removed as it is a duplicate

  distance_matrix_adult[lower.tri(distance_matrix_adult)] <- NA
  distance_matrix_juvenile[lower.tri(distance_matrix_juvenile)] <- NA




  if(sum(which(distance_matrix_adult < threshold & distance_matrix_juvenile
     < threshold)) == 0){
    return(last_year_dataCLC)
  }   # Checks if there are zero individuals who are alike.

  # Checks which individuals are closer than the threshold distance,
     indicating they are too similar.

  same <- which(distance_matrix_adult < threshold & distance_matrix_juvenile
      < threshold, arr.ind = T)
  same <- same[same[, 1]-same[,2] != 0, , drop = FALSE]  # Removes the rows
     indicating a species is to similar to itself.
  rownames(same) <- NULL
```

```
if(sum(same) == 0){
  return(last_year_dataCLC)
}
# Check is there are zero species that are the same and stops function if
   that is the case.



# Initialize an empty list to store groups
groups <- list()

# Function to find group index for a species
find_group <- function(species_id) {
  for (g in seq_along(groups)) {
    if (species_id %in% unlist(groups[[g]])) {
      return(g)
    }
  }
  return(0)
}

# Iterate over rows in the matrix
for (s in 1:nrow(same)) {
  species1 <- same[s, 1]
  species2 <- same[s, 2]

  # Find groups for each species
  group1 <- find_group(species1)
  group2 <- find_group(species2)

  if (group1 == 0 & group2 == 0) {
    # Create a new group
    groups <- c(groups, list(c(species1, species2)))
  } else if (group1 == 0) {
    # Add species1 to the group containing species2
    groups[[group2]] <- c(groups[[group2]], species1)
  } else if (group2 == 0) {
    # Add species2 to the group containing species1
    groups[[group1]] <- c(groups[[group1]], species2)
  } else if (group1 != group2) {
    # Merge two groups
    groups[[group1]] <- c(groups[[group1]], groups[[group2]])
    groups <- groups[-group2]
  }
}


# Filter out duplicate species in each group
groups <- lapply(groups, function(group) unique(group))

final_data <- last_year_dataCLC         # Place to store filtered data
```

```
  total.sub <- c()                      # Place to store subspecies

  #Add population count of "subspecies" to main species

  for(q in seq_along(groups)){
    combo <- NULL
    combo <- groups[[q]]
    main <- combo[which.max(final_data[combo,4])]
    sub <- combo[-which.max(final_data[combo,4])]
    final_data[main,4] <- final_data[main,4] + sum(final_data[sub,4])
    total.sub <- rbind(c(total.sub, sub))

  }
  # Remove subspecies
  final_data <- final_data[-total.sub, , drop = FALSE]

  return(final_data)
}
```

Resource and Parameter initialization

```
# Resource initializations -------------------


Num.Res <- 16                   # Number of resources
res.Abund <-  50000             # Abundance of resources

# Evenly distributed Resources

# SLC:
resource.freq.even.slc <- rep(1/Num.Res, times = Num.Res)
                         # res. freq.
resource.prop.even.slc <- c(seq(from = -2.5, to = 2.5, length.out = Num.Res)
   )   # res. property
resource.freq.even.slc <- res.Abund*resource.freq.even.slc


# CLC:

resource.property.even.clc <- c(seq(from = -2.5, to = 2.5, length.out = Num.
   Res))

resource.frequency.even.clc <- rep(1/Num.Res, times = Num.Res)

resource.abundance.adults.even.clc     <- res.Abund
                                # res. abundance of adults and juveniles
resource.abundance.juveniles.even.clc  <- res.Abund

resFreqMatrix.even.clc <- matrix(resource.frequency.even.clc, nrow=2, ncol=
   length(resource.frequency.even.clc ), byrow = TRUE)
resFreqMatrix.even.clc [1, ] <- resFreqMatrix.even.clc [1, ]*resource.
   abundance.adults.even.clc
```

```
resFreqMatrix.even.clc [2, ] <- resFreqMatrix.even.clc [2, ]*resource.
   abundance.juveniles.even.clc

resPropMatrix.even.clc <- matrix(resource.property.even.clc, nrow=2, ncol=
   length(resource.property.even.clc ), byrow = TRUE)

rownames(resFreqMatrix.even.clc) <- c("Adult", "Juvenile")
colnames(resFreqMatrix.even.clc)  <- paste0("Resource␣", 1:ncol(
   resFreqMatrix.even.clc))

rownames(resPropMatrix.even.clc)<-c("Adult", "Juvenile")
colnames(resPropMatrix.even.clc)  <- paste0("Resource␣", 1:ncol(
   resPropMatrix.even.clc))

# Normal resources:

m <- 0     # mean
s <- 1     # standard deviation
N.resource.frequency <- c()
N.resource.property<- c(seq(from = -2.5, to = 2.5, length.out = Num.Res))

mid.add <- c()
midpoint <- c()

# Loop to create an approximation of normal distribution for discrete
   resources.
for(i in 1:(length(N.resource.property))){
  mid.add <- (N.resource.property[i+1]-N.resource.property[i])/2
  high.midpoint <- N.resource.property[(i)]+mid.add
  low.midpoint <- N.resource.property[(i)]-mid.add
  if(i == 1){
    N.resource.frequency[i] <- pnorm(high.midpoint, mean = m, sd = s)
  }else if(i == length(N.resource.property)){
    low.midpoint <- N.resource.property[(i-1)] + (N.resource.property[i]-N.
       resource.property[i-1])/2
    N.resource.frequency[i] <- pnorm(low.midpoint, mean = m, sd = s, lower.
       tail = FALSE)
  }else{
    N.resource.frequency[i] <- pnorm(high.midpoint, mean = m, sd = s) -
       pnorm(low.midpoint, mean = m, sd = s)
  }
}



resource.abundance.adults.norm.clc      <- res.Abund
resource.abundance.juveniles.norm.clc   <- res.Abund

# SLC:

resource.prop.norm.slc  <- N.resource.property
resource.freq.norm.slc  <- res.Abund*N.resource.frequency
```

```
# CLC:

resFreqMatrix.norm.clc  <- matrix(N.resource.frequency, nrow=2, ncol=length(
   N.resource.frequency), byrow = TRUE)

resFreqMatrix.norm.clc [1, ] <- resFreqMatrix.norm.clc [1, ]*resource.
   abundance.adults.norm.clc
resFreqMatrix.norm.clc [2, ] <- resFreqMatrix.norm.clc [2, ]*resource.
   abundance.juveniles.norm.clc

rownames(resFreqMatrix.norm.clc ) <- c("Adult", "Juvenile")
colnames(resFreqMatrix.norm.clc )  <- paste0("Resource␣", 1:ncol(
   resFreqMatrix.norm.clc ))


resPropMatrix.norm.clc  <- matrix(N.resource.property, nrow=2, ncol=length(N
   .resource.property), byrow = TRUE)


rownames(resPropMatrix.norm.clc )<-c("Adult", "Juvenile")
colnames(resPropMatrix.norm.clc )  <- paste0("Resource␣", 1:ncol(
   resPropMatrix.norm.clc))


# Skewed resource distribution

# SLC

tot <- (Num.Res*(Num.Res+1))/2

x <- 1/tot
resource.freq.skew.slc <- c()

for (i in 1:Num.Res){
  resource.freq.skew.slc[i] <- i*x
}

resource.prop.skew.slc <- c(seq(from = -2.5, to = 2.5, length.out = Num.Res)
   )
resource.freq.skew.slc <- res.Abund*resource.freq.skew.slc


# CLC:

resource.property.skew.clc <- c(seq(from = -2.5, to = 2.5, length.out = Num.
   Res))


resource.frequency.skew.clc <- c()
for (i in 1:Num.Res){
```

```
  resource.frequency.skew.clc[i] <- i*x
}


resource.abundance.adults     <- res.Abund
resource.abundance.juveniles  <- res.Abund

resFreqMatrix.skew.clc <- matrix(resource.frequency.skew.clc, nrow=2, ncol=
   length(resource.frequency.skew.clc), byrow = TRUE)
resFreqMatrix.skew.clc[1, ] <- resFreqMatrix.skew.clc[1, ]*resource.
   abundance.adults
resFreqMatrix.skew.clc[2, ] <- resFreqMatrix.skew.clc[2, ]*resource.
   abundance.juveniles

resPropMatrix.skew.clc <- matrix(resource.property.skew.clc, nrow=2, ncol=
   length(resource.property.skew.clc), byrow = TRUE)

rownames(resFreqMatrix.skew.clc) <- c("Adult", "Juvenile")
colnames(resFreqMatrix.skew.clc)  <- paste0("Resource␣", 1:ncol(
   resFreqMatrix.skew.clc))

rownames(resPropMatrix.skew.clc)<-c("Adult", "Juvenile")
colnames(resPropMatrix.skew.clc)  <- paste0("Resource␣", 1:ncol(
   resPropMatrix.skew.clc))



# Bimodal normal resources
# These are created using two normal resource distributions next to eacht
   oher

m1 <- -1.25        # Mean of the two distributions
m2 <-  1.25
s <- 0.5           # Standard deviation
Bi.resource.frequency <- c()
Bi.resource.property<- c(seq(from = -2.5, to = 2.5, length.out = Num.Res))



mid.add <- c()
midpoint <- c()

# Similar as the for loop that creates an approximation of normal resource
   except the
# two different normal curves are used. At first a normal curve with mean
   -1.25 is used
# in the middle of the resource distribution it switches to the the normal
   curve with mean
# 1.25. As the total sum under two normal curves is = 2, each frequency
   value is diveded by
# 2 to create a frequency where the total sum = 1.
# This functions value of s and m1 and m2 would need to be adjusted and
```

```
   checked if another
# set of trait values is explored. For example: sum( Bi.resource.frequency)
   should be approximatly
# equal to 1.

for(i in 1:(length(Bi.resource.property))){
  mid.add <- (Bi.resource.property[i+1]-Bi.resource.property[i])/2
  high.midpoint <- Bi.resource.property[(i)]+mid.add
  low.midpoint <- Bi.resource.property[(i)]-mid.add
  if(i == 1){
    Bi.resource.frequency[i] <- pnorm(high.midpoint, mean = m1, sd = s)/2
  }else if(i == length(Bi.resource.property)){
    low.midpoint <- Bi.resource.property[(i-1)] + (Bi.resource.property[i]-
       Bi.resource.property[i-1])/2
    Bi.resource.frequency[i] <- pnorm(low.midpoint, mean = m2, sd = s, lower
       .tail = FALSE)
  }else if (Bi.resource.property[i]<0) {
    Bi.resource.frequency[i] <- (pnorm(high.midpoint, mean = m1, sd = s) -
       pnorm(low.midpoint, mean = m1, sd = s))/2
  }else{
    Bi.resource.frequency[i] <- (pnorm(high.midpoint, mean = m2, sd = s) -
       pnorm(low.midpoint, mean = m2, sd = s))/2
  }
}


resource.abundance.adults.binorm.clc      <- res.Abund
                                # res. abundance of adults and juveniles
resource.abundance.juveniles.binorm.clc   <- res.Abund

# SLC:

resource.prop.binorm.slc  <- Bi.resource.property             # res.
   property
resource.freq.binorm.slc  <- res.Abund*Bi.resource.frequency


# CLC:

resFreqMatrix.binorm.clc  <- matrix(Bi.resource.frequency, nrow=2, ncol=
   length(Bi.resource.frequency), byrow = TRUE)

resFreqMatrix.binorm.clc [1, ] <- resFreqMatrix.binorm.clc [1, ]*resource.
   abundance.adults.binorm.clc
resFreqMatrix.binorm.clc [2, ] <- resFreqMatrix.binorm.clc [2, ]*resource.
   abundance.juveniles.binorm.clc

rownames(resFreqMatrix.binorm.clc ) <- c("Adult", "Juvenile")
colnames(resFreqMatrix.binorm.clc )  <- paste0("Resource␣", 1:ncol(
   resFreqMatrix.binorm.clc ))
```

```
resPropMatrix.binorm.clc  <- matrix(Bi.resource.property, nrow=2, ncol=
   length(Bi.resource.property), byrow = TRUE)


rownames(resPropMatrix.binorm.clc )<-c("Adult", "Juvenile")
colnames(resPropMatrix.binorm.clc )  <- paste0("Resource␣", 1:ncol(
   resPropMatrix.binorm.clc))


# Two resources

resource.prop <- c(-1,1)
resource.frequency <- c(0.5, 0.5)
resource.frequency.as <- c(0.2, 0.8)

resFreqMatrix.2res <- matrix(resource.frequency, nrow=2, ncol=length(
   resource.frequency), byrow = TRUE)
resFreqMatrixAs.2res <- matrix(resource.frequency.as, nrow=2, ncol=length(
   resource.frequency.as), byrow = TRUE)


resFreqMatrix.2res[1, ] <- resFreqMatrix.2res[1, ]*res.Abund
resFreqMatrix.2res[2, ] <- resFreqMatrix.2res[2, ]*res.Abund

resFreqMatrixAs.2res[1, ] <- resFreqMatrixAs.2res[1, ]*res.Abund
resFreqMatrixAs.2res[2, ] <- resFreqMatrixAs.2res[2, ]*res.Abund

rownames(resFreqMatrix.2res) <- c("Adult", "Juvenile")
colnames(resFreqMatrix.2res)  <- paste0("Resource␣", 1:ncol(resFreqMatrixAs
   .2res))

rownames(resFreqMatrixAs.2res) <- c("Adult", "Juvenile")
colnames(resFreqMatrixAs.2res)  <- paste0("Resource␣", 1:ncol(
   resFreqMatrixAs.2res))

resPropMatrix.2res <- matrix(resource.prop, nrow=2, ncol=length(resource.
   prop), byrow = TRUE)


rownames(resPropMatrix.2res)<-c("Adult", "Juvenile")
colnames(resFreqMatrix.2res)  <- paste0("Resource␣", 1:ncol(resPropMatrix.2
   res))
# Clean up -------------
# To de clutter the environment a bit

rm(high.midpoint)
rm(i)
rm(low.midpoint)
rm(m)
rm(m1)
rm(m2)
rm(mid.add)
```

```
rm(midpoint)
rm(N.resource.frequency)
rm(N.resource.property)
rm(res.Abund)
rm(resource.abundance.adults)
rm(resource.abundance.adults.binorm.clc)
rm(resource.abundance.adults.even.clc)
rm(resource.abundance.adults.norm.clc)
rm(resource.abundance.juveniles)
rm(resource.abundance.juveniles.binorm.clc)
rm(resource.abundance.juveniles.even.clc)
rm(resource.abundance.juveniles.norm.clc)
rm(resource.frequency.even.clc)
rm(resource.frequency.skew.clc)
rm(s)
rm(tot)
rm(x)
rm(Bi.resource.frequency)
rm(Bi.resource.property)
rm(resource.property.even.clc)
rm(resource.property.skew.clc)
rm(resource.prop)
rm(resource.frequency)
rm(resource.frequency.as)

# ------------------------

# Parameter initialization ----------------------------


popSize <- 10                                          # Initial population
   size
sigma <- seq(from = 0.05, to = 0.8, length.out = 6)  # Niche width
im <-  0                                               # Determines how if
   there is immigration or not, can be 0 or 1
fmax <-  2                                             # Maximum fecundity
   value
kA <-  0.5                                             # Half saturation
   constants
kJ <-  0.5
mutProb <- 0.00001                                     # Mutational
   probability
mutVar <- 0.05                                         # Variation in the
   amount of trait mutation
time.steps <- 50000                                    # Number of generations
    simulation is run for
iniP <- 0                                              # Initial phenotype for
    simple life cycle
iniPJ <- 0                                             # Initial phenotype for
    juvenile (complex)
iniPA <- 0                                             # Initial phenotype for
    adult (complex)
```

```
nmorphs <-  1                                          # Number of species at
   start of simulation
threshold <-  0.0005                                   # Threshold which is
   used at end of simulation to remove very low abundance species
maxTr = 3                                              # Maximum possible
   trait value of immigrant
minTr = -3                                             # Minimum possible
   trait value of immigrant
```

Simulation example

I run simulations using jobs to free up the console for other things. If you do not have a computer with the cpu power to handle several simulations at once, do the jobs separately. This simulation usually takes my computer around 1.5 days to run. The time can be shortened significantly by reducing the number of repetitions and the number of σ_s values explored. To check on the progress of the simulations, look under background jobs, (same page as console). Please ensure you have run the above code before this for it to work.

```
 # Running simulations -----------------------------


# Even

# The job function runs in a separate environment so we need to import the
   desired parameters and functions.
# The result will be an environment titled the same thing as the job's name.

job::job(even = {

  rep <- 10                              # Number of times simulation will be
     run

  # Where the output of simulations will be stored:

  Total.species.SLC.single.even <- c()

  Total.species.CLC.even <- matrix(data = NA, nrow = length(sigma), ncol =
     length(sigma))
  rownames(Total.species.CLC.even) <- sigma  #ADULTS
  colnames(Total.species.CLC.even) <- sigma #JUVENILES



  # SLC

  Total.species.SLC.even <- list()
  Total.endpoint.SLC.even <- list()

  # Simulations:

  for(r in 1:rep) {

    id <- 1
                                                                          #
```

```
      Used to differentiate different runs in the output

  print(paste0("loop␣", r, "␣started"))
                                          # To see progress of simulation
      in "background jobs"

  Number.species.SLC.even <- c()
  endpoint.SLC.even <- list()

  for(i in 1:length(sigma)){


    outputSLC <- resourceCompetitionSLC(resProp=resource.prop.even.slc,
       iniP = iniP, resFreq=resource.freq.even.slc, resGen=matrix(c(sigma[
       i],sigma[i])),
                                        popSize = popSize, mutProb=mutProb
                                           , mutVar=mutVar, time.steps =
                                           time.steps, im = im, fmax =
                                           fmax, kA = kA, nmorphs =
                                           nmorphs,
                                        threshold = threshold)


    #Filter out similar "species" and collect number of species data

    final.data.SLC.even <- slc.groups(output = outputSLC)
    Number.species.SLC.even[i] <- nrow(final.data.SLC.even)

    #Collect endpoint data
    final.data.SLC.even$ID <- c(rep(id, times = nrow(final.data.SLC.even))
       )


    endpoint.SLC.even[[i]] <- final.data.SLC.even
    id <- id + 1
  }

  Total.species.SLC.even[[r]] <- Number.species.SLC.even
  Total.endpoint.SLC.even[[r]] <- endpoint.SLC.even
}


# Caluclating mean and SD of 10 runs



Total.mean.SLC.even <- sapply(1:length(sigma), function(i) mean(sapply(
   Total.species.SLC.even, function(x) x[i])))

array.data.SLC <- array(unlist(Total.species.SLC.even), dim = c(dim(Total.
   species.SLC.even[[1]]), length(Total.species.SLC.even)))
```

```
Total.sd.SLC.even <- sapply(1:length(sigma), function(i) sd(sapply(Total.
   species.SLC.even, function(x) x[i])))


# CLC

print("clc␣start")

Total.species.CLC.even <- list()

Total.endpoint.CLC.even <- list()



for(a in 1:rep){
  print(paste0("loop␣", a, "␣started"))


  id <- 1

  species.CLC.even <- matrix(data = NA, nrow = length(sigma), ncol =
     length(sigma))
  rownames(species.CLC.even) <- sigma  #ADULTS
  colnames(species.CLC.even) <- sigma #JUVENILES

  endpoint.CLC.even <- c()

  for(b in 1:length(sigma)){

    for(k in 1:length(sigma)){


      outputCLC <- resourceCompetitionCLC(resProp=resPropMatrix.even.clc,
         resFreq=resFreqMatrix.even.clc, iniPA = iniPA, iniPJ = iniPJ,
         resGen=matrix(c(sigma[b],sigma[k])),
                                          popSize = popSize, mutProb=
                                             mutProb, mutVar=mutVar, time.
                                             steps = time.steps, im = im,
                                             fmax = fmax, kA = kA, nmorphs
                                              = nmorphs,
                                          threshold = threshold)



      #Filter out similar "species"
      final.data.CLC.even <- clc.groups(output = outputCLC)

      # Collect Data

      species.CLC.even[b, k] <- nrow(final.data.CLC.even)
      final.data.CLC.even$Adult.gen <- c(rep(sigma[b], times = nrow(final.
         data.CLC.even)))
```

```
        final.data.CLC.even$Juv.gen <- c(rep(sigma[k], times = nrow(final.
           data.CLC.even)))
        final.data.CLC.even$ID <- c(rep(id, times = nrow(final.data.CLC.even
           )))
        endpoint.CLC.even <- rbind(endpoint.CLC.even, final.data.CLC.even)
        id <- id + 1
      }

    }
    Total.species.CLC.even[[a]] <- species.CLC.even
    Total.endpoint.CLC.even[[a]] <- endpoint.CLC.even
  }

  # Calculating mean of 10 runs

  # Combine matrices in the list into a 3D array
  array.data.CLC <- array(unlist(Total.species.CLC.even), dim = c(dim(Total.
     species.CLC.even[[1]]), length(Total.species.CLC.even)))


  # Calculate mean and standard deviation along the third dimension (across
     the list)
  Total.mean.CLC.even <- apply(array.data.CLC, c(1, 2), mean)
  Total.sd.CLC.even <- apply(array.data.CLC, c(1, 2), sd)



  job::export(list(Total.mean.CLC.even, Total.sd.CLC.even, Total.mean.SLC.
     even, Total.sd.SLC.even, Total.endpoint.SLC.even, Total.endpoint.CLC.
     even))
}, import = c(resPropMatrix.even.clc, resFreqMatrix.even.clc,
   resourceCompetitionCLC, resource.prop.even.slc, resource.freq.even.slc,
   resourceCompetitionSLC,
              clc.groups, slc.groups, sigma, popSize, im, fmax, kA, kJ,
                 mutProb, mutVar, time.steps, iniP, iniPA, iniPJ, nmorphs,
                 threshold, maxTr, minTr))




# Normal

job::job(norm = {

  rep <- 10

  Total.species.SLC.single.norm <- c()

  Total.species.CLC.norm <- matrix(data = NA, nrow = length(sigma), ncol =
     length(sigma))
  rownames(Total.species.CLC.norm) <- sigma  #ADULTS
  colnames(Total.species.CLC.norm) <- sigma #JUVENILES
```

```
# SLC

Total.species.SLC.norm <- list()
Total.endpoint.SLC.norm <- list()


for(r in 1:rep) {


  id <- 1

  print(paste0("loop␣", r, "␣started"))

  Number.species.SLC.norm <- c()
  endpoint.SLC.norm <- list()

  for(i in 1:length(sigma)){


    outputSLC <- resourceCompetitionSLC(resProp=resource.prop.norm.slc,
       iniP = iniP, resFreq=resource.freq.norm.slc, resGen=matrix(c(sigma[
       i],sigma[i])),
                                        popSize = popSize, mutProb=mutProb
                                           , mutVar=mutVar, time.steps =
                                           time.steps, im = im, fmax =
                                           fmax, kA = kA, nmorphs =
                                           nmorphs,
                                        threshold = threshold)


    #Filter out similar "species" and collect number of species data

    final.data.SLC.norm <- slc.groups(output = outputSLC)
    Number.species.SLC.norm[i] <- nrow(final.data.SLC.norm)

    #Collect endpoint data
    final.data.SLC.norm$ID <- c(rep(id, times = nrow(final.data.SLC.norm))
       )


    endpoint.SLC.norm[[i]] <- final.data.SLC.norm
    id <- id + 1
  }

  Total.species.SLC.norm[[r]] <- Number.species.SLC.norm
  Total.endpoint.SLC.norm[[r]] <- endpoint.SLC.norm
}
```

```
# Caluclating mean and SD of 10 runs


Total.mean.SLC.norm <- sapply(1:length(sigma), function(i) mean(sapply(
   Total.species.SLC.norm, function(x) x[i])))

array.data.SLC <- array(unlist(Total.species.SLC.norm), dim = c(dim(Total.
   species.SLC.norm[[1]]), length(Total.species.SLC.norm)))

Total.sd.SLC.norm <- sapply(1:length(sigma), function(i) sd(sapply(Total.
   species.SLC.norm, function(x) x[i])))


# CLC

print("clc␣start")

Total.species.CLC.norm <- list()

Total.endpoint.CLC.norm <- list()



for(a in 1:rep){

  print(paste0("loop␣", a, "␣started"))


  id <- 1

  species.CLC.norm <- matrix(data = NA, nrow = length(sigma), ncol =
     length(sigma))
  rownames(species.CLC.norm) <- sigma  #ADULTS
  colnames(species.CLC.norm) <- sigma #JUVENILES

  endpoint.CLC.norm <- c()

  for(b in 1:length(sigma)){

    for(k in 1:length(sigma)){


      outputCLC <- resourceCompetitionCLC(resProp=resPropMatrix.norm.clc,
         resFreq=resFreqMatrix.norm.clc, iniPA = iniPA, iniPJ = iniPJ,
         resGen=matrix(c(sigma[b],sigma[k])),
                                          popSize = popSize, mutProb=
                                             mutProb, mutVar=mutVar, time.
                                             steps = time.steps, im = im,
                                             fmax = fmax, kA = kA, nmorphs
                                              = nmorphs,
                                          threshold = threshold)
```

```
        #Filter out similar "species"
        final.data.CLC.norm <- clc.groups(output = outputCLC)

        # Collect Data

        species.CLC.norm[b, k] <- nrow(final.data.CLC.norm)
        final.data.CLC.norm$Adult.gen <- c(rep(sigma[b], times = nrow(final.
           data.CLC.norm)))
        final.data.CLC.norm$Juv.gen <- c(rep(sigma[k], times = nrow(final.
           data.CLC.norm)))
        final.data.CLC.norm$ID <- c(rep(id, times = nrow(final.data.CLC.norm
           )))
        endpoint.CLC.norm <- rbind(endpoint.CLC.norm, final.data.CLC.norm)
        id <- id + 1
      }

    }
    Total.species.CLC.norm[[a]] <- species.CLC.norm
    Total.endpoint.CLC.norm[[a]] <- endpoint.CLC.norm
  }

  # Calculating mean of 10 runs

  # Combine matrices in the list into a 3D array
  array.data.CLC <- array(unlist(Total.species.CLC.norm), dim = c(dim(Total.
     species.CLC.norm[[1]]), length(Total.species.CLC.norm)))


  # Calculate mean and standard deviation along the third dimension (across
     the list)
  Total.mean.CLC.norm <- apply(array.data.CLC, c(1, 2), mean)
  Total.sd.CLC.norm <- apply(array.data.CLC, c(1, 2), sd)



  job::export(list(Total.mean.CLC.norm, Total.sd.CLC.norm, Total.mean.SLC.
     norm, Total.sd.SLC.norm, Total.endpoint.SLC.norm, Total.endpoint.CLC.
     norm))
}, import = c(resPropMatrix.norm.clc, resFreqMatrix.norm.clc,
   resourceCompetitionCLC, resource.prop.norm.slc, resource.freq.norm.slc,
   resourceCompetitionSLC,
              clc.groups, slc.groups, sigma, popSize, im, fmax, kA, kJ,
                 mutProb, mutVar, time.steps, iniP, iniPA, iniPJ, nmorphs,
                 threshold, maxTr, minTr))
```

```
# Skewed


job::job(skew = {

  rep <- 10

  Total.species.SLC.single.skew <- c()

  Total.species.CLC.skew <- matrix(data = NA, nrow = length(sigma), ncol =
     length(sigma))
  rownames(Total.species.CLC.skew) <- sigma  #ADULTS
  colnames(Total.species.CLC.skew) <- sigma #JUVENILES



  # SLC

  Total.species.SLC.skew <- list()
  Total.endpoint.SLC.skew <- list()


  for(r in 1:rep) {

    id <- 1

    print(paste0("loop␣", r, "␣started"))

    Number.species.SLC.skew <- c()
    endpoint.SLC.skew <- list()

    for(i in 1:length(sigma)){


      outputSLC <- resourceCompetitionSLC(resProp=resource.prop.skew.slc,
         iniP = iniP, resFreq=resource.freq.skew.slc, resGen=matrix(c(sigma[
         i],sigma[i])),
                                          popSize = popSize, mutProb=mutProb
                                             , mutVar=mutVar, time.steps =
                                             time.steps, im = im, fmax =
                                             fmax, kA = kA, nmorphs =
                                             nmorphs,
                                          threshold = threshold)


      #Filter out similar "species" and collect number of species data

      final.data.SLC.skew <- slc.groups(output = outputSLC)
      Number.species.SLC.skew[i] <- nrow(final.data.SLC.skew)

      #Collect endpoint data
```

```
    final.data.SLC.skew$ID <- c(rep(id, times = nrow(final.data.SLC.skew))
       )


    endpoint.SLC.skew[[i]] <- final.data.SLC.skew
    id <- id + 1
  }

  Total.species.SLC.skew[[r]] <- Number.species.SLC.skew
  Total.endpoint.SLC.skew[[r]] <- endpoint.SLC.skew
}


# Caluclating mean and SD of 10 runs



Total.mean.SLC.skew <- sapply(1:length(sigma), function(i) mean(sapply(
   Total.species.SLC.skew, function(x) x[i])))

array.data.SLC <- array(unlist(Total.species.SLC.skew), dim = c(dim(Total.
   species.SLC.skew[[1]]), length(Total.species.SLC.skew)))

Total.sd.SLC.skew <- sapply(1:length(sigma), function(i) sd(sapply(Total.
   species.SLC.skew, function(x) x[i])))


# CLC

print("clc␣start")

Total.species.CLC.skew <- list()

Total.endpoint.CLC.skew <- list()



for(a in 1:rep){


  print(paste0("loop␣", a, "␣started"))

  id <- 1

  species.CLC.skew <- matrix(data = NA, nrow = length(sigma), ncol =
     length(sigma))
  rownames(species.CLC.skew) <- sigma  #ADULTS
  colnames(species.CLC.skew) <- sigma #JUVENILES

  endpoint.CLC.skew <- c()

  for(b in 1:length(sigma)){
```

```
    for(k in 1:length(sigma)){


      outputCLC <- resourceCompetitionCLC(resProp=resPropMatrix.skew.clc,
         resFreq=resFreqMatrix.skew.clc, iniPA = iniPA, iniPJ = iniPJ,
         resGen=matrix(c(sigma[b],sigma[k])),
                                          popSize = popSize, mutProb=
                                             mutProb, mutVar=mutVar, time.
                                             steps = time.steps, im = im,
                                             fmax = fmax, kA = kA, nmorphs
                                              = nmorphs,
                                          threshold = threshold)



      #Filter out similar "species"
      final.data.CLC.skew <- clc.groups(output = outputCLC)

      # Collect Data

      species.CLC.skew[b, k] <- nrow(final.data.CLC.skew)
      final.data.CLC.skew$Adult.gen <- c(rep(sigma[b], times = nrow(final.
         data.CLC.skew)))
      final.data.CLC.skew$Juv.gen <- c(rep(sigma[k], times = nrow(final.
         data.CLC.skew)))
      final.data.CLC.skew$ID <- c(rep(id, times = nrow(final.data.CLC.skew
         )))
      endpoint.CLC.skew <- rbind(endpoint.CLC.skew, final.data.CLC.skew)
      id <- id + 1
    }

  }
  Total.species.CLC.skew[[a]] <- species.CLC.skew
  Total.endpoint.CLC.skew[[a]] <- endpoint.CLC.skew
}

# Calculating mean of 10 runs

# Combine matrices in the list into a 3D array
array.data.CLC <- array(unlist(Total.species.CLC.skew), dim = c(dim(Total.
   species.CLC.skew[[1]]), length(Total.species.CLC.skew)))


# Calculate mean and standard deviation along the third dimension (across
   the list)
Total.mean.CLC.skew <- apply(array.data.CLC, c(1, 2), mean)
Total.sd.CLC.skew <- apply(array.data.CLC, c(1, 2), sd)
```

```
  job::export(list(Total.mean.CLC.skew, Total.sd.CLC.skew, Total.mean.SLC.
     skew, Total.sd.SLC.skew, Total.endpoint.SLC.skew, Total.endpoint.CLC.
     skew))
}, import = c(resPropMatrix.skew.clc, resFreqMatrix.skew.clc,
   resourceCompetitionCLC, resource.prop.skew.slc, resource.freq.skew.slc,
   resourceCompetitionSLC,
              clc.groups, slc.groups, sigma, popSize, im, fmax, kA, kJ,
                 mutProb, mutVar, time.steps, iniP, iniPA, iniPJ, nmorphs,
                 threshold, maxTr, minTr))


# Bimodal Normal

job::job(binorm = {

  rep <- 3

  Total.species.SLC.single.binorm <- c()

  Total.species.CLC.binorm <- matrix(data = NA, nrow = length(sigma), ncol =
      length(sigma))
  rownames(Total.species.CLC.binorm) <- sigma  #ADULTS
  colnames(Total.species.CLC.binorm) <- sigma #JUVENILES



  # SLC

  Total.species.SLC.binorm <- list()
  Total.endpoint.SLC.binorm <- list()

  for(r in 1:rep) {


    id <- 1

    print(paste0("loop␣", r, "␣started"))

    Number.species.SLC.binorm <- c()
    endpoint.SLC.binorm <- list()

    for(i in 1:length(sigma)){


      outputSLC <- resourceCompetitionSLC(resProp=resource.prop.binorm.slc,
         iniP = iniP, resFreq=resource.freq.binorm.slc, resGen=matrix(c(
         sigma[i],sigma[i])),
                                          popSize = popSize, mutProb=mutProb
                                             , mutVar=mutVar, time.steps =
                                             time.steps, im = im, fmax =
                                             fmax, kA = kA, nmorphs =
                                             nmorphs,
```

```
                                        threshold = threshold)


    #Filter out similar "species" and collect number of species data

    final.data.SLC.binorm <- slc.groups(output = outputSLC)
    Number.species.SLC.binorm[i] <- nrow(final.data.SLC.binorm)

    #Collect endpoint data
    final.data.SLC.binorm$ID <- c(rep(id, times = nrow(final.data.SLC.
       binorm)))


    endpoint.SLC.binorm[[i]] <- final.data.SLC.binorm
    id <- id + 1
  }

  Total.species.SLC.binorm[[r]] <- Number.species.SLC.binorm
  Total.endpoint.SLC.binorm[[r]] <- endpoint.SLC.binorm
}


# Caluclating mean and SD of 10 runs



Total.mean.SLC.binorm <- sapply(1:length(sigma), function(i) mean(sapply(
   Total.species.SLC.binorm, function(x) x[i])))

array.data.SLC <- array(unlist(Total.species.SLC.binorm), dim = c(dim(
   Total.species.SLC.binorm[[1]]), length(Total.species.SLC.binorm)))

Total.sd.SLC.binorm <- sapply(1:length(sigma), function(i) sd(sapply(Total
   .species.SLC.binorm, function(x) x[i])))


# CLC

print("clc␣start")

Total.species.CLC.binorm <- list()

Total.endpoint.CLC.binorm <- list()


for(a in 1:rep){
  print(paste0("loop␣", a, "␣started"))

  id <- 1

  species.CLC.binorm <- matrix(data = NA, nrow = length(sigma), ncol =
     length(sigma))
```

```
  rownames(species.CLC.binorm) <- sigma  #ADULTS
  colnames(species.CLC.binorm) <- sigma #JUVENILES

  endpoint.CLC.binorm <- c()

  for(b in 1:length(sigma)){

    for(k in 1:length(sigma)){


      outputCLC <- resourceCompetitionCLC(resProp=resPropMatrix.binorm.clc
         , resFreq=resFreqMatrix.binorm.clc, iniPA = iniPA, iniPJ = iniPJ,
          resGen=matrix(c(sigma[b],sigma[k])),
                                          popSize = popSize, mutProb=
                                             mutProb, mutVar=mutVar, time.
                                             steps = time.steps, im = im,
                                             fmax = fmax, kA = kA, nmorphs
                                              = nmorphs,
                                          threshold = threshold)



      #Filter out similar "species"
      final.data.CLC.binorm <- clc.groups(output = outputCLC)

      # Collect Data

      species.CLC.binorm[b, k] <- nrow(final.data.CLC.binorm)
      final.data.CLC.binorm$Adult.gen <- c(rep(sigma[b], times = nrow(
         final.data.CLC.binorm)))
      final.data.CLC.binorm$Juv.gen <- c(rep(sigma[k], times = nrow(final.
         data.CLC.binorm)))
      final.data.CLC.binorm$ID <- c(rep(id, times = nrow(final.data.CLC.
         binorm)))
      endpoint.CLC.binorm <- rbind(endpoint.CLC.binorm, final.data.CLC.
         binorm)
      id <- id + 1
    }

  }
  Total.species.CLC.binorm[[a]] <- species.CLC.binorm
  Total.endpoint.CLC.binorm[[a]] <- endpoint.CLC.binorm
}

# Calculating mean of 10 runs

# Combine matrices in the list into a 3D array
array.data.CLC <- array(unlist(Total.species.CLC.binorm), dim = c(dim(
   Total.species.CLC.binorm[[1]]), length(Total.species.CLC.binorm)))
```

```
  # Calculate mean and standard deviation along the third dimension (across
     the list)
  Total.mean.CLC.binorm <- apply(array.data.CLC, c(1, 2), mean)
  Total.sd.CLC.binorm <- apply(array.data.CLC, c(1, 2), sd)



  job::export(list(Total.mean.CLC.binorm, Total.sd.CLC.binorm, Total.mean.
     SLC.binorm, Total.sd.SLC.binorm, Total.endpoint.SLC.binorm, Total.
     endpoint.CLC.binorm))
}, import = c(resPropMatrix.binorm.clc, resFreqMatrix.binorm.clc,
   resourceCompetitionCLC, resource.prop.binorm.slc, resource.freq.binorm.
   slc,
               resourceCompetitionSLC, clc.groups, slc.groups, sigma, popSize
                  , im, fmax, kA, kJ, mutProb, mutVar, time.steps, iniP,
                  iniPA, iniPJ, nmorphs, threshold, maxTr, minTr))



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

Plotting example

There are of course many ways to plot the data so feel free to use your own method. But this is how I produced the graphs in the thesis. The graphs produced by this is the mean number of species for all the different combinations of σ_s (figure 4 and 8) and graphs showing phenotypic endpoints for the complex communities for all $\sigma_a = \sigma_j$

```
# Plotting Mean number of Species
   ---------------------------------------------

# Even


Total.mean.CLC.even <- even$Total.mean.CLC.even
Total.mean.SLC.even <- even$Total.mean.SLC.even

Total.sd.CLC.even <-  even$Total.sd.CLC.even
Total.sd.SLC.even <-  even$Total.sd.SLC.even

x <- as.factor(sigma)



df.CLC <- data.frame(
  Juvenile.trait = rep(x, each = length(x)),
  Adult.trait = rep(x, times = length(x)),
  Richness = as.vector(Total.mean.CLC.even),
  sd = as.vector(Total.sd.CLC.even),
  Cycle = rep("Complex", times = length(x)*length(x))
)

df.SLC <- data.frame(
```

```
  Juvenile.trait = x,
  Adult.trait = x,
  Richness = as.vector(Total.mean.SLC.even),
  sd = as.vector(Total.sd.SLC.even),
  Cycle = rep("Simple", times = length(x))
)



df.combined <- rbind(df.CLC, df.SLC)

color_palette <- magma(length(sigma))

even.plot <- ggplot(df.combined, aes(x = Adult.trait, y = Richness, shape =
   Cycle, color = Juvenile.trait, stroke = 1.7)) +
  geom_point(data = ~filter(.x, Cycle == "Simple"),size = 7, position =
     position_dodge(0.2), color = "black") +
  geom_point(data = ~filter(.x, Cycle == "Complex"),size = 7, position =
     position_dodge(0.2)) +
  #geom_errorbar(aes(ymin=Richness-sd, ymax=Richness+sd), width=.05) +   #
     position=position_dodge(.9)  # If you wish to add error bars include
     this.
  scale_y_continuous(limits = c(0, 30)) +
  xlab("Adult␣Generalism") +
  ylab("Number␣of␣species") +
  labs(color = "Juvenile␣\nGeneralism", shape = "Life␣cycle") +
  ggtitle("Even␣Resource␣distribution") +
  theme_minimal(base_family = "LM␣Roman␣10", base_size = 15) +
  theme(plot.title = element_text(size = 18)) +
  scale_shape_manual(values = c(1,4)) +
  scale_color_manual(values = c(color_palette, "black"))




even.plot

# Normal

Total.mean.CLC.norm <- norm$Total.mean.CLC.norm
Total.mean.SLC.norm <- norm$Total.mean.SLC.norm

Total.sd.CLC.norm <-  norm$Total.sd.CLC.norm
Total.sd.SLC.norm <-  norm$Total.sd.SLC.norm

x <- as.factor(sigma)



df.CLC <- data.frame(
  Juvenile.trait = rep(x, each = length(x)),
  Adult.trait = rep(x, times = length(x)),
```

```
  Richness = as.vector(Total.mean.CLC.norm),
  sd = as.vector(Total.sd.CLC.norm),
  Cycle = rep("Complex", times = length(x)*length(x))
)

df.SLC <- data.frame(
  Juvenile.trait = x,
  Adult.trait = x,
  Richness = as.vector(Total.mean.SLC.norm),
  sd = as.vector(Total.sd.SLC.norm),
  Cycle = rep("Simple", times = length(x))
)



df.combined <- rbind(df.CLC, df.SLC)


norm.plot <- ggplot(df.combined, aes(x = Adult.trait, y = Richness, shape =
   Cycle, color = Juvenile.trait, stroke = 1.7)) +
  geom_point(data = ~filter(.x, Cycle == "Simple"),size = 7, position =
     position_dodge(0.2), color = "black") +
  geom_point(data = ~filter(.x, Cycle == "Complex"),size = 7, position =
     position_dodge(0.2)) +
  #geom_errorbar(aes(ymin=Richness-sd, ymax=Richness+sd), width=.1) +   #
     position=position_dodge(.9)
  scale_y_continuous(limits = c(0, 30)) +
  xlab("Adult␣Generalism") +
  ylab("Number␣of␣species") +
  labs(color = "Juvenile␣\nGeneralism", shape = "Life␣cycle") +
  ggtitle("Normal␣Resource␣distribution") +
  theme_minimal(base_family = "LM␣Roman␣10", base_size = 15) +
  theme(plot.title = element_text(size = 18)) +
  scale_shape_manual(values = c(1,4)) +
  scale_color_manual(values = c(color_palette, "black"))

norm.plot

# Skewed


Total.mean.CLC.skew <- skew$Total.mean.CLC.skew
Total.mean.SLC.skew <- skew$Total.mean.SLC.skew

Total.sd.CLC.skew <-  skew$Total.sd.CLC.skew
Total.sd.SLC.skew <-  skew$Total.sd.SLC.skew

x <- as.factor(sigma)



df.CLC <- data.frame(
```

```
  Juvenile.trait = rep(x, each = length(x)),
  Adult.trait = rep(x, times = length(x)),
  Richness = as.vector(Total.mean.CLC.skew),
  sd = as.vector(Total.sd.CLC.skew),
  Cycle = rep("Complex", times = length(x)*length(x))
)

df.SLC <- data.frame(
  Juvenile.trait = x,
  Adult.trait = x,
  Richness = as.vector(Total.mean.SLC.skew),
  sd = as.vector(Total.sd.SLC.skew),
  Cycle = rep("Simple", times = length(x))
)



df.combined <- rbind(df.CLC, df.SLC)


skew.plot <- ggplot(df.combined, aes(x = Adult.trait, y = Richness, shape =
   Cycle, color = Juvenile.trait, stroke = 1.7)) +
  geom_point(data = ~filter(.x, Cycle == "Simple"),size = 7, position =
     position_dodge(0.2), color = "black") +
  geom_point(data = ~filter(.x, Cycle == "Complex"),size = 7, position =
     position_dodge(0.2)) +
  #geom_errorbar(aes(ymin=Richness-sd, ymax=Richness+sd), width=.05) +   #
     position=position_dodge(.9)
  scale_y_continuous(limits = c(0, 30)) +
  xlab("Adult␣Generalism") +
  ylab("Number␣of␣species") +
  labs(color = "Juvenile␣\nGeneralism", shape = "Life␣cycle") +
  ggtitle("Skewed␣Resource␣distribution") +
  theme_minimal(base_family = "LM␣Roman␣10", base_size = 15) +
  theme(plot.title = element_text(size = 18)) +
  scale_shape_manual(values = c(1,4)) +
  scale_color_manual(values = c(color_palette, "black"))

skew.plot

# Binormal

Total.mean.CLC.binorm <- binorm$Total.mean.CLC.binorm
Total.mean.SLC.binorm <- binorm$Total.mean.SLC.binorm

Total.sd.CLC.binorm <-  binorm$Total.sd.CLC.binorm
Total.sd.SLC.binorm <-  binorm$Total.sd.SLC.binorm

x <- as.factor(sigma)
```

```
df.CLC <- data.frame(
  Juvenile.trait = rep(x, each = length(x)),
  Adult.trait = rep(x, times = length(x)),
  Richness = as.vector(Total.mean.CLC.binorm),
  sd = as.vector(Total.sd.CLC.binorm),
  Cycle = rep("Complex", times = length(x)*length(x))
)

df.SLC <- data.frame(
  Juvenile.trait = x,
  Adult.trait = x,
  Richness = as.vector(Total.mean.SLC.binorm),
  sd = as.vector(Total.sd.SLC.binorm),
  Cycle = rep("Simple", times = length(x))
)



df.combined <- rbind(df.CLC, df.SLC)


binorm.plot <- ggplot(df.combined, aes(x = Adult.trait, y = Richness, shape
   = Cycle, color = Juvenile.trait, stroke = 1.7)) +
  geom_point(data = ~filter(.x, Cycle == "Simple"),size = 7, position =
     position_dodge(0.2), color = "black") +
  geom_point(data = ~filter(.x, Cycle == "Complex"),size = 7, position =
     position_dodge(0.2)) +
  #geom_errorbar(aes(ymin=Richness-sd, ymax=Richness+sd), width=.05) +   #
     position=position_dodge(.9)
  scale_y_continuous(limits = c(0, 30)) +
  xlab("Adult␣Generalism") +
  ylab("Number␣of␣species") +
  labs(color = "Juvenile␣\nGeneralism", shape = "Life␣cycle") +
  ggtitle("Bimodal␣Normal␣Resource␣distribution") +
  theme_minimal(base_family = "LM␣Roman␣10", base_size = 15) +
  theme(plot.title = element_text(size = 18)) +
  scale_shape_manual(values = c(1,4)) +
  scale_color_manual(values = c(color_palette, "black"))

binorm.plot

# Together:

all.plots <- (even.plot + norm.plot) / (skew.plot + binorm.plot)
all.plots + plot_layout(guides = "collect") + plot_annotation(tag_levels = "
   A",
  tag_prefix = "(",
  tag_suffix = ")")

# Plotting Phenotype Endpoint --------------------

# Even ----------------
```

```
# Plotting several runs ----------------------------------------

# Adult = Juvenile sigma

Res <- list()

pdf("plots.even.combined.side.pdf")   # This will create a pdf in your
   working directory that is filled with the plots
                                          # created after the command dev.
   off() is used.

for(s in 1:length(sigma)){
  adu.sigma <- sigma[s]


  juv.sigma <- adu.sigma


  last.year.list.even <- data.frame()

  for(i in 1:length(even$Total.endpoint.CLC.even)){
    this.run <- even$Total.endpoint.CLC.even[[i]]
    this.run$run <- rep(i, time = nrow(this.run))
    this.run <- this.run[this.run$Adult.gen == adu.sigma, ]
    this.run <- this.run[this.run$Juv.gen == juv.sigma, ]
    last.year.list.even <- rbind(last.year.list.even, this.run)
  }

  plot.list.even <- list()

  for (i in 1:9){

    data <- last.year.list.even[last.year.list.even$run == i, ]

    color.palette <- plasma(length(data$Adult_Trait))

    plot.list.even[[i]] <- ggplot(data, aes(x = Juvenile_Trait, y = Adult_
       Trait)) +
      geom_point(aes(size=Num_Individuals), color = color.palette, show.
         legend = FALSE) +
      labs(title = substitute(sigma == value, list(value = adu.sigma)), x =
         "Juvenile␣Trait", y = "Adult␣Trait", size = "Number␣of␣individuals"
         ) +                  # Labels for the axes
      scale_x_continuous(limits = c(-3, 3))+
      scale_y_continuous(limits = c(-3, 3))+
      scale_size_continuous(limits=c(1,40000),breaks=c(seq(from = 0, to =
         40000, by = 5000))) +
      theme(aspect.ratio=1) +
      theme_minimal(base_family = "LM␣Roman␣10", base_size = 10)


  }
```

```
  plots <- wrap_plots(plot.list.even)



  Res[[s]] <- plots + plot_annotation(
    title = 'Even␣Distribution',
    theme = theme(plot.title = element_text(hjust = 0.5, size = 15, family =
        "LM␣Roman␣10"), plot.subtitle = element_text(hjust = 0.5, size = 15,
        family = "LM␣Roman␣10"))
  )+ coord_fixed()

  print(Res[[s]])


}

dev.off()

# Normal ------------------
# Plotting several runs ----------------------------------------

# Adult = Juvenile sigma

Res <- list()

pdf("plots.norm.combined.side.pdf")

for(s in 1:length(sigma)){
  adu.sigma <- sigma[s]

  juv.sigma <- adu.sigma


  last.year.list.norm <- data.frame()

  for(i in 1:length(norm$Total.endpoint.CLC.norm)){
    this.run <- norm$Total.endpoint.CLC.norm[[i]]
    this.run$run <- rep(i, time = nrow(this.run))
    this.run <- this.run[this.run$Adult.gen == adu.sigma, ]
    this.run <- this.run[this.run$Juv.gen == juv.sigma, ]
    last.year.list.norm <- rbind(last.year.list.norm, this.run)
  }

  plot.list.norm <- list()

  for (i in 1:9){

    data <- last.year.list.norm[last.year.list.norm$run == i, ]

    color.palette <- plasma(length(data$Adult_Trait))
```

```
    plot.list.norm[[i]] <- ggplot(data, aes(x = Juvenile_Trait, y = Adult_
       Trait)) +
      geom_point(aes(size=Num_Individuals), color = color.palette, show.
         legend = FALSE) +
      labs(title = substitute(sigma == value, list(value = adu.sigma)), x =
         "Juvenile␣Trait", y = "Adult␣Trait", size = "Number␣of␣individuals"
         ) +                  # Labels for the axes
      scale_x_continuous(limits = c(-3, 3))+
      scale_y_continuous(limits = c(-3, 3))+
      scale_size_continuous(limits=c(1,40000),breaks=c(seq(from = 0, to =
         40000, by = 5000))) +
      theme_minimal(base_family = "LM␣Roman␣10", base_size = 10)


  }


  plots <- wrap_plots(plot.list.norm)

  Res[[s]] <- plots + plot_annotation(
    title = 'Normal␣Distribution',
    theme = theme(plot.title = element_text(hjust = 0.5, size = 15, family =
        "LM␣Roman␣10"), plot.subtitle = element_text(hjust = 0.5, size = 15,
        family = "LM␣Roman␣10"))
  )+ coord_fixed()
  print(Res[[s]])


}

dev.off()

# Skewed --------------------
# Plotting several runs ----------------------------------------

# Adult = Juvenile sigma
Res <- list()

pdf("plots.skew.combined.side.pdf")

for(s in 1:length(sigma)){
  adu.sigma <- sigma[s]

  juv.sigma <- adu.sigma


  last.year.list.skew <- data.frame()

  for(i in 1:length(skew$Total.endpoint.CLC.skew)){
    this.run <- skew$Total.endpoint.CLC.skew[[i]]
    this.run$run <- rep(i, time = nrow(this.run))
```

```
    this.run <- this.run[this.run$Adult.gen == adu.sigma, ]
    this.run <- this.run[this.run$Juv.gen == juv.sigma, ]
    last.year.list.skew <- rbind(last.year.list.skew, this.run)
  }

  plot.list.skew <- list()

  for (i in 1:9){

    data <- last.year.list.skew[last.year.list.skew$run == i, ]

    color.palette <- plasma(length(data$Adult_Trait))

    plot.list.skew[[i]] <- ggplot(data, aes(x = Juvenile_Trait, y = Adult_
       Trait)) +
      geom_point(aes(size=Num_Individuals), color = color.palette, show.
         legend = FALSE) +
      labs(title = substitute(sigma == value, list(value = adu.sigma)), x =
         "Juvenile␣Trait", y = "Adult␣Trait", size = "Number␣of␣individuals"
         ) +                  # Labels for the axes
      scale_x_continuous(limits = c(-3, 3))+
      scale_y_continuous(limits = c(-3, 3))+
      scale_size_continuous(limits=c(1,40000),breaks=c(seq(from = 0, to =
         40000, by = 5000))) +
      theme_minimal(base_family = "LM␣Roman␣10", base_size = 10)


  }


  plots <- wrap_plots(plot.list.skew)

  Res[[s]] <- plots + plot_annotation(
    title = 'Skewed␣Distribution',
    theme = theme(plot.title = element_text(hjust = 0.5, size = 15, family =
        "LM␣Roman␣10"), plot.subtitle = element_text(hjust = 0.5, size = 15,
        family = "LM␣Roman␣10"))
  )+ coord_fixed()
  print(Res[[s]])


}

dev.off()


# Bi modal Normal -------------------
# Plotting several runs ----------------------------------------

# Adult = Juvenile sigma
Res <- list()
```

```
pdf("plots.binorm.combined.pdf")

for(s in 1:length(sigma)){
  adu.sigma <- sigma[s]

  juv.sigma <- adu.sigma


  last.year.list.binorm <- data.frame()

  for(i in 1:length(binorm$Total.endpoint.CLC.binorm)){
    this.run <- binorm$Total.endpoint.CLC.binorm[[i]]
    this.run$run <- rep(i, time = nrow(this.run))
    this.run <- this.run[this.run$Adult.gen == adu.sigma, ]
    this.run <- this.run[this.run$Juv.gen == juv.sigma, ]
    last.year.list.binorm <- rbind(last.year.list.binorm, this.run)
  }

  plot.list.binorm <- list()

  for (i in 1:9){

    data <- last.year.list.binorm[last.year.list.binorm$run == i, ]

    color.palette <- plasma(length(data$Adult_Trait))

    plot.list.binorm[[i]] <- ggplot(data, aes(x = Juvenile_Trait, y = Adult_
       Trait)) +
      geom_point(aes(size=Num_Individuals), color = color.palette, show.
         legend = FALSE) +
      labs(title = substitute(sigma == value, list(value = adu.sigma)), x =
         "Juvenile␣Trait", y = "Adult␣Trait", size = "Number␣of␣individuals"
         ) +                  # Labels for the axes
      scale_x_continuous(limits = c(-3, 3))+
      scale_y_continuous(limits = c(-3, 3))+
      scale_size_continuous(limits=c(1,40000),breaks=c(seq(from = 0, to =
         40000, by = 5000))) +
      theme_minimal(base_family = "LM␣Roman␣10", base_size = 10)


  }


  plots <- wrap_plots(plot.list.binorm)

  Res[[s]] <- plots + plot_annotation(
    title = 'Bimodal␣Normal␣Distribution',
    theme = theme(plot.title = element_text(hjust = 0.5, size = 15, family =
        "LM␣Roman␣10"), plot.subtitle = element_text(hjust = 0.5, size = 15,
        family = "LM␣Roman␣10"))
  )+ coord_fixed()
  print(Res[[s]])
```

```
}

dev.off()
```

References

Chang, W. 2023. extrafont: Tools for Using Fonts.

Garnier, Simon, Ross, Noam, Rudis, Robert, Camargo, A. Pedro, Sciaini, Marco, Scherer, and Cédric. 2023. viridis(Lite) - Colorblind-Friendly Color Maps for R.

Lindeløv, J. K. 2021. job: Run Code as an RStudio Job - Free Your Console.

Pedersen, T. L. 2024. patchwork: The Composer of Plots.

R Core Team. 2023. R: A Language and Environment for Statistical Computing. R Foundation for Statistical Computing, Vienna, Austria.

Saul, M. 2021. FamilyRank: Algorithm for Ranking Predictors Using Graphical Domain Knowledge.

Wickham, H. 2016. ggplot2: Elegant Graphics for Data Analysis. Springer-Verlag New York.

Wickham, H., M. Averick, J. Bryan, W. Chang, L. D. McGowan, R. François, G. Grolemund, A. Hayes, L. Henry, J. Hester, M. Kuhn, T. L. Pedersen, E. Miller, S. M. Bache, K. Müller, J. Ooms, D. Robinson, D. P. Seidel, V. Spinu, K. Takahashi, D. Vaughan, C. Wilke, K. Woo, and H. Yutani. 2019. Welcome to the tidyverse. Journal of Open Source Software 4:1686.

Wickham, H., R. François, L. Henry, K. Müller, and D. Vaughan. 2023. dplyr: A Grammar of Data Manipulation.