Integral Projection Models Across Scales

Dissertation

zur Erlangung des Doktorgrades der Naturwissenschaften (Dr. rer. nat.)

der Naturwissenschaftlichen Fakultät I – Biowissenschaften –

der Martin-Luther-Universität Halle-Wittenberg,

Vorgelegt von

Herrn Sam C. Levin, M.Sc.

**Gutachter:**

1. Dr. Tiffany Knight
2. Dr. Ingolf Kühn
3. Dr. Brian Enquist

Verteidigt am Juni 24th, 2024

# Contents

# Summary

The study of structured populations dynamics has become a key aspect of many sub-disciplines within ecology as a whole. Structured population models provide an elegant tool to scale from individual observations to population level inference. Life tables are probably the most widely used tool to study human demography. Currently, matrix population models (MPMs) are the most widely used structured population models in ecology. Integral projection models (IPMs) are newer, but are seeing increasingly frequent use in the literature. IPMs allow researchers to study the demography of population structured by one or more continuously distributed traits in discrete time. Concurrently, researchers have developed a variety of tools and techniques to parameterize and implement IPMs. However, pressing questions in ecology, evolution, and conservation, and their associated data and computational requirements, have outpaced the development of tooling to aid their implementation, interpretation, and synthesis. This dissertation contributes to filling these gaps by introducing new computational tools and data collection techniques, and then applying them them to an invasive plant genus that has invaded multiple continents. These tools and analyses enable large scale data collection, ease the implementation of complex IPMs, and provide a more streamlined experience for syntheses.

In **Chapter 2**, I introduce a new *R* package, `ipmr`, to implement and analyse IPMs. This *R* package provides a domain specific language to implement deterministic and stochastic models, with two options for handling the latter. Furthermore, `ipmr` handles density dependence with minimal additional overhead. Density dependence is a critically important population process that, until `ipmr`, was not implemented in any *R* packages for structured populations. `ipmr` is designed to accommodate almost any type of regression model used in the vital rate fitting process, and also includes methods for extracting the basic building blocks used in many downstream analyses. This chapter includes two appendices with detailed case studies demonstrating the package's use, and the package itself contains six vignettes that are intended for audiences from a range of technical backgrounds.

In **Chapter 3**, I introduce two additional tools: the PADRINO IPM Database, and `Rpadrino`, an *R* package for interfacing with PADRINO. PADRINO houses IPMs extracted from published peer-reviewed studies, as well as extensive metadata that enables researchers to filter the database down to models that fit their research questions. `Rpadrino` provides a clean interface to rebuild these models from *R* using the `ipmr` *R* package as a backend. This chapter also includes two detailed case studies on how to use these tools for both standalone analyses and in conjunction with other databases to power much larger scale syntheses.

In **Chapter 4**, I examine the drivers of invasiveness in the *Carpobrotus* genus using data collected by unmanned aerial vehicles (UAVs, drones) on four continents. We examine how climate impacts the population dynamics of this widely invasive genus using a combination of IPMs and Life Table Response Experiments (LTREs). We do not find any link between climate drivers and overall population performance, and unmeasured site-specific effects are more important than climate with respect to fitness. This chapter further demonstrates how to scale inference across levels of organization (individuals to populations) and space (local to regional, regional to continental).

In **Chapter 5**, I provide a synthesis of previous chapters. Specifically, I highlight both knowledge gaps that this dissertation fills and new computational capabilities this dissertation provides, the dissertation's limitations, and suggest directions for future research.



Keywords: Integral Projection Model, climate change, invasive species, demography



# Zusammenfassung

Die Untersuchung der Dynamik strukturierter Populationen ist zu einem Schlüsselaspekt vieler Teildisziplinen innerhalb der Ökologie als Ganzes geworden. Strukturierte Populationsmodelle bieten ein elegantes Instrument, um von Einzelbeobachtungen auf die Populationsebene zu schließen. Sterbetafeln sind wahrscheinlich das am häufigsten verwendete Instrument zur Untersuchung der menschlichen Demografie. Derzeit sind Matrix-Populationsmodelle (MPMs) die am häufigsten verwendeten strukturierten Populationsmodelle in der Ökologie.

Integrale Projektionsmodelle (IPMs) sind neuer, werden aber in der Literatur immer häufiger verwendet. Mit IPMs können Forscher die Demografie von Populationen untersuchen, die durch ein oder mehrere kontinuierlich verteilte Merkmale in diskreter Zeit strukturiert sind. Gleichzeitig haben die Forscher eine Vielzahl von Instrumenten und Techniken zur Parametrisierung und Umsetzung von IPMs entwickelt. Die drängenden Fragen in den Bereichen Ökologie, Evolution und Naturschutz und die damit verbundenen Daten- und Berechnungsanforderungen haben jedoch die Entwicklung von Instrumenten zur Unterstützung ihrer Umsetzung, Interpretation und Synthese überholt. Diese Dissertation trägt dazu bei, diese Lücken zu schließen, indem sie neue Berechnungswerkzeuge und Datenerfassungstechniken vorstellt und sie dann auf eine invasive Pflanzengattung anwendet, die in mehrere Kontinente eingedrungen ist. Diese Werkzeuge und Analysen ermöglichen eine groß angelegte Datenerfassung, erleichtern die Umsetzung komplexer IPMs und bieten eine rationalisierte Erfahrung für Synthesen.

In **Kapitel 2** stelle ich ein neues *R*-Paket, `ipmr`, vor, um IPMs zu implementieren und zu analysieren. Dieses *R* Paket stellt eine domänenspezifische Sprache zur Verfügung, um deterministische und stochastische Modelle zu implementieren, mit zwei Optionen für die Handhabung letzterer. Darüber hinaus behandelt `ipmr` die Dichteabhängigkeit mit minimalem zusätzlichem Overhead. Dichteabhängigkeit ist ein sehr wichtiger Populationsprozess, der bis `ipmr` in keinem *R* Paket für strukturierte Populationen implementiert war. `ipmr` ist so konzipiert, dass es fast alle Arten von Regressionsmodellen aufnehmen kann, die im Prozess der Anpassung der Vitalrate verwendet werden, und enthält auch Methoden zur Extraktion der grundlegenden Bausteine, die in vielen nachgeschalteten Analysen verwendet werden. Dieses Kapitel enthält zwei Anhänge mit detaillierten Fallstudien, die die Verwendung des Pakets demonstrieren, und das Paket selbst enthält sechs Vignetten, die für ein Publikum mit unterschiedlichem technischen Hintergrund gedacht sind.

In **Kapitel 3** stelle ich zwei zusätzliche Werkzeuge vor: die PADRINO IPM-Datenbank und `Rpadrino`, ein *R* Paket zur Anbindung an PADRINO. PADRINO enthält IPMs, die aus veröffentlichten, von Experten begutachteten Studien stammen, sowie umfangreiche Metadaten, die es Forschern ermöglichen, die Datenbank nach Modellen zu filtern, die ihren Forschungsfragen entsprechen. `Rpadrino` bietet eine saubere Schnittstelle, um diese Modelle von *R* aus neu zu erstellen, wobei das _R-Paket `ipmr` als Backend dient. Dieses Kapitel enthält auch zwei detaillierte Fallstudien, wie diese Werkzeuge sowohl für eigenständige Analysen als auch in Verbindung mit anderen Datenbanken verwendet werden können, um Synthesen in größerem Umfang durchzuführen.

In **Kapitel 4** untersuche ich die Triebkräfte der Invasivität der Gattung *Carpobrotus* anhand von Daten, die von unbemannten Luftfahrzeugen (UAVs, Drohnen) auf vier Kontinenten gesammelt wurden. Wir untersuchen, wie sich das Klima auf die Populationsdynamik dieser weit verbreiteten invasiven Gattung auswirkt, indem wir eine Kombination aus IPMs und Life Table Response Experiments (LTREs) verwenden. Wir finden keinen Zusammenhang zwischen Klimatreibern und der Gesamtleistung der Population, und nicht gemessene standortspezifische Effekte sind im Hinblick auf die Fitness wichtiger als das Klima. In diesem Kapitel wird außerdem gezeigt, wie man Schlussfolgerungen auf verschiedenen Organisationsebenen (Individuen bis hin zu Populationen) und räumlichen Ebenen (lokal bis regional, regional bis kontinental) ziehen kann.

In **Kapitel 5** gebe ich eine Zusammenfassung der vorangegangenen Kapitel. Insbesondere hebe ich sowohl Wissenslücken hervor, die diese Dissertation schließt, als auch neue rechnerische Möglichkeiten, die diese Dissertation bietet, sowie die Grenzen der Dissertation und schlage Richtungen für zukünftige Forschung vor.

Keywords: Integral Projection Model, Klimawandel, invasive Arten, Demographie

# Chapter 1: Introduction

## 1.1 Population models in ecology

Understanding population dynamics is central to ecology (Crone et al. 2011), evolution (Metcalf & Pavard 2007), and conservation (Morris & Doak 2002). Rates of survival and reproduction form the two main fitness components that drive evolution, and determine the fate of populations as a whole (Coulson 2012). These vital rates are therefore crucial to understanding numerous other phenomena, such as change in population size, changes in causes of mortality and/or fertility, or the direction and magnitude of natural selection. Indeed, the utility of this idea - that the fates of closed populations are determined by rates of birth and death - has been demonstrated numerous times over the years. Notable applications include (though are certainly not limited to) understanding predator-prey interactions (Lotka 1925, Volterra 1926), the consequences of age/stage/size structure in populations (Leslie 1945, Lefkovitch 1965), invasive and endangered species management (Caswell 1978, de Kroon et al. 1986, Morris & Doak 2002), and the consequences of climate change (Teller et al. 2016, Compagnoni et al. 2021).

Early applications of population biology focused on human populations and modeled total population size (rather than, for example, number of age $a$ individuals). These yielded insight into drivers of exponential population growth (and an unhealthy dose of nativist and racist sentiment, Franklin 1751, Malthus 1798). They gave the classic equation:

$$N(t) = N_0 e^{rt}. \tag{1.1.1}$$

$N(t)$ is the population size at time $t$, $N_0$ is the initial population size, and $r$ is the intrinsic rate of population growth. Subsequent developments of logistic growth to account for carrying capacity were introduced by Verhulst in 1838, and (perhaps more famously) applied by A.G. McKendrick (1911) to study bacterial growth:

$$N(t+1) = (1 + r(1 - \frac{N(t)}{K}))N(t). \tag{1.1.2}$$

Here, $K$ represents the carrying capacity of the population, which is the maximum population size that the environment can support. Shortly afterwards, biologists began to recognize the importance of dividing the population into sub-groups to understand how within-population differences in vital rates impacts dynamics. In other words, not all individuals in the $N(t)$ terms are identical. The practice of sub-dividing the population based on some trait is referred to as "structuring the population". Beginning in the early 1900s, theoretical advancements aimed to understand age-specific rates of birth and death are recognized as the beginning of this field of study (McKendrick 1926):

$$\frac{\delta n}{\delta t} + \frac{\delta n}{\delta a} = -m(a)n \tag{1.1.3}$$

Here, $m(a)$ is the age specific mortality rate. However, this approach did not become popular in ecology for quite some time, likely due to the tedious nature of the calculations (Ebert 1999). P.H. Leslie introduced the discrete time, age structured projection matrix in 1945 (Leslie 1945). This approach provided a more elegant way of summarizing vital rates, though still required dutiful attention to detail when performing the calculations by hand:

$$N(a, t+1) = AN(a, t). \tag{1.1.4}$$

$N(a,t)$ and $N(a,t+1)$ represent the population age distribution at time $t$ and $t+1$, and $A$ is an $m \times m$ matrix with $m$ age classes and their associated age-specific survival and fertility rates (where $m$ is maximum age $a$ of the organism in question). Extensions of Leslie's age-based approach to incorporate stage-structured populations (Lefkovitch 1965) and multiple structuring variables (Goodman 1969) advanced theory, but did

not necessarily drive broader uptake. The advent of personal computers and the recognition that numerical analysis methods fit well with this rapidly growing technology lead to a rapid adoption in the late 1970s and early 1980s. Concurrently, Hal Caswell and colleagues' showed how to calculate population responses to absolute (sensitivity, 1978) and proportional (elasticity, 1986) perturbations to matrix elements, which are still a vital analytical tool for applied practitioners and theoreticians. A subsequent text book provided computer code to implement these analyses, along with many others, which helped make these approaches more accessible (Caswell 1989).

In the intervening years, further developments have enabled researchers to accommodate additional biological realism into their structured population models, such as density dependence (Leslie 1959, Liu & Cohen 1987, Jensen 1995) and environmental stochasticity (Caswell 1989). However, there was also a recognition that many species have their life cycle structured by a continuous trait, like weight, height, diameter, or hatching date. Furthermore, partitioning these continuous traits up into discrete categories introduced increased data requirements for parameterization and/or reduced the biological realism of the model by treating all individuals in a given class as identical (Vandermeer 1978). The integral projection model (IPM) was proposed to alleviate this issue (Easterling et al. 2000). IPMs provide a framework to model the dynamics of populations structured by any number of continuous and discrete traits (Ellner & Rees 2006, Ellner et al. 2016). Further developments to the IPM framework yielded methods for including a variety of additional factors, including density dependence (Rose et al. 2005, Adler et al. 2010), environmental stochasticity (Childs et al. 2004, Rees & Ellner 2009), spatial structure (Jongejans et al. 2011), and demographic stochasticity (Vindenes et al. 2011). These developments have enabled analyses across a variety of fields, including evolution (e.g. Coulson et al. 2011), conservation biology (e.g. Ferrer-Cervantes et al. 2012), and global change biology (e.g. Nicole et al. 2011, Simmonds et al. 2020).

The main strengths of IPMs lie in i) the ability to choose continuous traits to model populations, ii) the ability to flexibly incorporate discrete and continuous traits into the same model, iii) efficient parameterization of the model using regression-based toolkits, and iv) numerical computation of the integrals (see 1.2) mean tools available for matrix models are usually available for IPMs. Thus, there is a rich array of life cycles one can model, with a variety of well understood tools for fitting functional forms to real data, and a broad repertoire of tools and frameworks to analyze the resulting IPM with.

## 1.2 Simple Integral Projection Models

An IPM describes how the abundance and distribution of a *trait* (denoted $z$ and $z'$) changes from time $t$ to time $t+1$. Simple IPMs are IPMs that use one, and only one, trait to structure the population in question. The current trait distribution is given by the function $n(z,t)$. A simple IPM for the trait distribution of $z'$ at $t+1$ is then:

$$n(z',t+1) = \int_L^U K(z',z)n(z,t)dz. \tag{1.2.1}$$

$K(z',z)$ is a *kernel* function (often referred to as a kernel) that describes all possible transitions of existing individuals (survival and maturation) and recruitment of new individuals ((a)sexual reproduction) from $t$ to $t+1$. $L, U$ are the lower and upper bounds that the value of trait $z$ can take, and the interval $[L,U]$ is referred to as the *domain*. To compute total population size, we simply integrate the trait distribution functions $\int_L^U n(z,t)$ ($\int_L^U n(z',t+1)$ to get the population size at $t+1$).

To aid interpretability, the kernel $K(z',z)$ is usually decomposed into *sub-kernels* representing transitions related to existing individuals (i.e. survival, change in trait value) and creation of new individuals:

$$K(z',z) = P(z',z) + F(z',z) + C(z',z). \tag{1.2.2}$$

$P(z',z)$ describes transitions of existing individuals due to survival and change in trait value. $F(z',z)$ describes per-capita sexual reproduction contributions. $C(z',z)$ describes per-capita asexual reproduction contributions.

IPM theory assumes these sub-kernels are somewhat smooth functions, with piecewise continuity satisfying this assumption (Ellner et al. 2016, Ch 2.2).

The $P(z', z)$, $F(z', z)$, and $C(z', z)$ sub-kernels are in turn comprised of vital rate functions. It is at this step that IPMs link the researcher's data to the population dynamics. For most IPMs, these vital rate functions are generated via regression models fit to real data collected in the field or in the lab (or some combination thereof).

For example, the $P(z', z)$ kernel from Bogdan et al. (2021) for *Carpobrotus spp.* contains two vital rate functions:

$$P(z', z) = s(z)G(z', z). \tag{1.2.3}$$

$s(z)$ was parameterized using a logistic regression of survival outcome (either 0 (dead) or 1 (alive)) at $t + 1$ on plant size $z$. $G(z', z)$ was parameterized using a linear regression of size $z'$ on size $z$. This resulted in:

$$Logit(s(z)) = \beta_{0,s} + \beta_{1,s} * z, \tag{1.2.4}$$

$$G(z', z) = f_G(z'|\mu_G(z), \sigma_G), \tag{1.2.5}$$

$$\mu_G(z) = \beta_{0,G} + \beta_{1,G} * z. \tag{1.2.6}$$

$\beta_{0,i}$ and $\beta_{1,i}$ are intercepts and slopes from each regression, respectively, and $f_G$ in 1.2.6 denotes a Gaussian probability density function. $\sigma_G$ is the standard deviation of the residuals from the growth regression.

The integrals are usually impossible to solve analytically (Ellner & Rees 2006), so numerical approximations are computed instead. The midpoint rule is the most commonly used one (others are possible, see Ellner et al. 2016, Chapter 6). This divides the domain $[L, U]$ into $m$ artificial bins centered at $z_i$ with a width of $h = (U - L)/m$, and midpoints $z_i = L + (i - 0.5) * h$ for $i = 1, 2, \ldots, m$. The midpoint rule approximation of 1.2.1 becomes:

$$n(z_j, t + 1) = h \sum_{i=1}^{m} K(z_j, z_i)n(z_i, t). \tag{1.2.7}$$

The numerical integration creates a discretized projection matrix $K$ analogous to $A$ in 1.1.4. Matrix multiplication of the discretized kernel and the discretized trait distribution generates a new trait distribution.

Because of the smiliarity between $K(z_j, z_i)$ in 1.2.7 and $A$ in 1.1.4, many of the tools developed for matrix models can be applied to IPMs with little to no additional effort. These include (but are not limited to!) population growth rates ($\lambda$), stable trait distributions ($w$), reproductive values ($v$), sensitivity and elasticity analysis. Statistical and mathematical methods for incorporating environmental stochasticity (Rees & Ellner 2009) and density dependence (Rose et al. 2005, Adler et al. 2010) are available. Furthermore, there are guides available for assistance with the former (Metcalf et al. 2015) and the latter (Ellner et al. 2016, Ch 5).

Simple IPMs have been developed for a wide array of applications, ranging from the theoretical to the practical. For example, Metcalf and colleagues (2009) combined simple IPMs for six tropical tree species with a discretely varying light environment to model life expectancy, age specific mortality, and passage times to a given size conditional on initial state. Bassar and colleagues (2016) developed a simple, density dependent IPM to show how asymmetric competition affects various life histroy traits of *Poecilia reticulata*). Nicole and colleagues (2011) examined how climate change and local habitat conditions would influence the viability of the endangered *Dracocephalum austriacum*. I developed a simple, density-independent IPM for *Lonicera maackii* to show that the degree of competitive release was a function of phylogenetic and functional novelty in its new local community (Levin et al. 2019, Levin et al. 2020). Zucaratto and colleagues (2020) developed

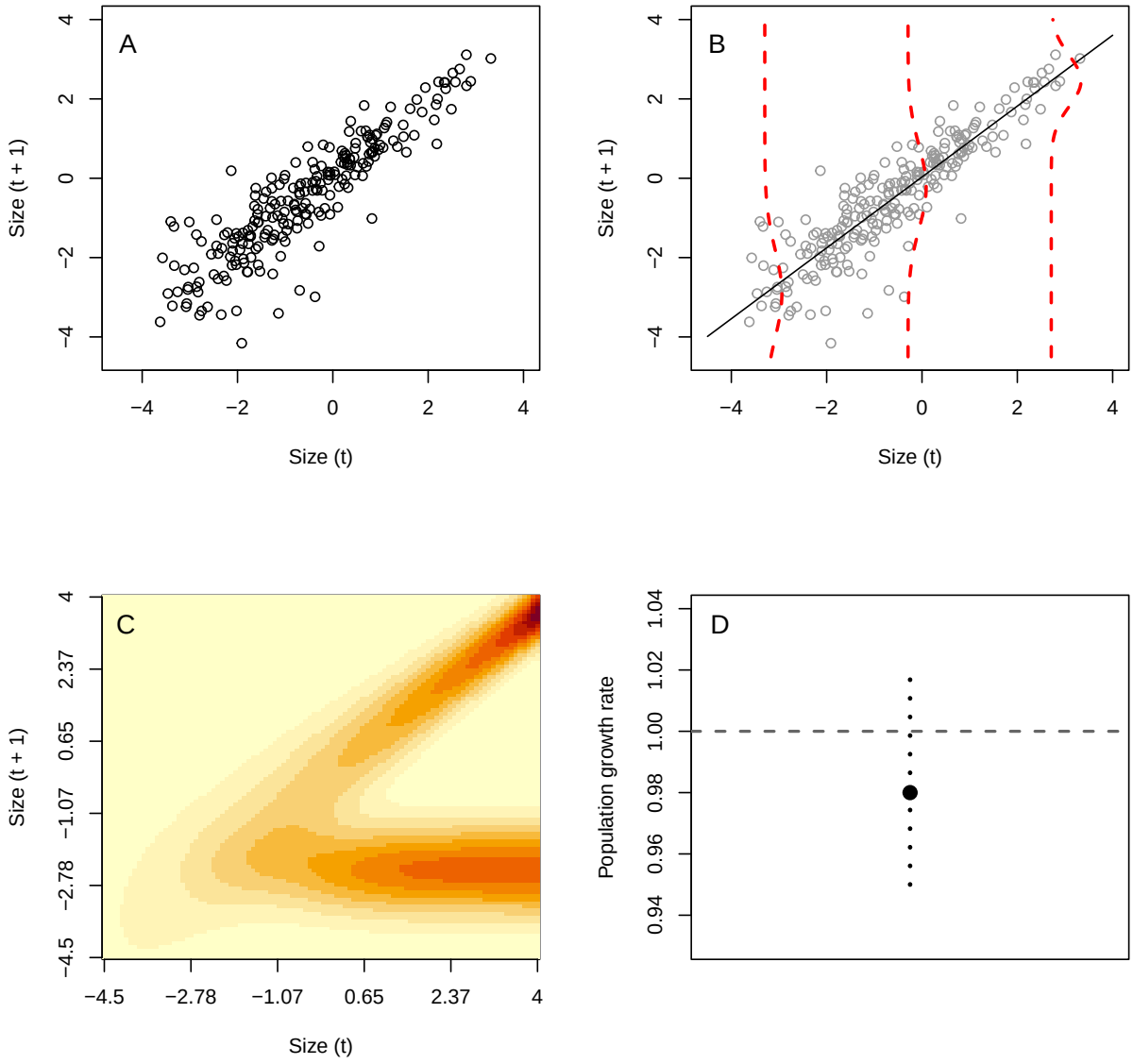



Figure 1: The link between data (A-B) and an IPM (C-D). This is typically a reversal of the way IPMs are described above, in that researchers start with data on sizes and demographic outcomes (A). These data get turned into functions in 1.2.4 - 1.2.6 using a variety of statistical methods, most typically a regression (B). In B, the solid black line depicts the predicted mean size at *t+1* vs observed size at *t*. The red lines depict the variation around the predicted mean. Note that the variance shrinks as the initial size increases - this illustrates the power of IPMs to accommodate a variety of functional relationships. Once individual vital rate functions are computed, these are combined to form P, F, and K (C) found in equations 1.2.2 and 1.2.3. From there, some biological inference is conducted (D). In this case, the per-capita growth rate ($\lambda$) and its confidence interval are depicted.

a simple IPM for the invasive *Roystonea oleracea* to develop a management plan for preserving Ilha Grande, an island off the coast of Brazil whose Atlantic Forest habitat is threatened. Despite the wide application of simple IPMs, there is a need to model life cycles and species that aren't readily summarized by a single, continuous trait (e.g. plants with a discrete seed bank stage). To that end, Ellner & Rees (2006) developed the general IPM, which is covered in the next section.

## 1.3 General Integral Projection Models

General IPMs are an extension of simple IPMs that allow researchers to model a population structured by multiple continuously and/or discretely traits. For example, many plants have long-lived seed banks, and researchers are typically consider this a discrete stage because of low trait variation across individuals, or because they lack sufficient data on said trait variation to create a continuous trait distribution (but see Eager et al. 2013). Therefore, the seedbank is a discrete stage represented by a single number (the number of seeds that are in it). Similarly, age- and trait-structured models may contain many copies of the same trait distribution - one for each age class. For example, an age $\times$ size structured IPM for *Ovis aries* may have 21 copies for log transformed body weight denoted $z_a$ for $a \in [0, 20]$ (Ellner, Childs, & Rees 2016, Ch 6). General IPMs allow researchers to accommodate these more complex life cycles in a single model.

The trait space in a general IPM is no longer one dimensional. We represent the entirety of the space with $Z$. $Z$ can include discrete points $D = \{z_1, z_2, \ldots, z_D\}$, and a set of continuous domains $C = \{Z_{D+1}, Z_{D+2}, \ldots, Z_{D+C}\}$ (Ellner et al. 2016, Chapter 6). Each continuous domain in $C$ is either a closed interval $[L, U]$ (like in a simple IPM), or a closed finite rectangle (e.g. with individuals cross-classified by size $z$ and "individual quality" $q$). Similarly, the population state is now comprised of different components. Discrete states are represented $n_j(t), j = 1, 2, \ldots D$ which give the number of individuals with state $D$. Continuous components are represented $n_j(z_j, t), j = D+1, D+2, \ldots D+C$, where the integral $\int_{Z_j} n_j(z_j, t)dz_j$ gives the number of individuals with state $j$. Transitions within and between traits are defined by a set of sub-kernels $K_{ij}, 1 \leq i, j \leq D+C$. When individuals in with trait $Z_j(t)$ contribute to $Z_i(t+1)$, $K_{ij} \neq 0$. Thus, there are four possible kinds of kernel components:

1. Discrete to discrete transitions: $K_{ij}$ is a single number. This includes, for example, the fraction of seeds in a seedbank that do not germinate and survives to remain in the seedbank the following year, or, when insect eggs remain in stasis across time steps, neither hatching nor dying.

2. Discrete to continuous transitions: $K_{ij} = k_{ij}(z')$. $k_{ij}(z'_i)$ is a state distribution that gives the probability of trait value $z'$ at $t+1$, and is the same for all individuals in $Z_j$. This includes, for example, the size distribution of seedlings emerging from the seedbank, or the size of newly hatched insects.

3. Continuous to discrete transitions: $K_{ij} = k_{ij}(z)$. $k_{ij}(z_j)$ is the state dependent per-capita contribution to state $i$ at $t+1$ from state $j$ at time $t$. This includes, for example, the number of seeds that a size $z_j$ plant produces that do not immediately germinate and enter the seedbank, or the number of insect eggs that a size $z_j$ adult produces that do not hatch immediately.

4. Continuous to continuous transitions: $K_{ij} = K_{ij}(z'_i, z_j)$. These are bivariate kernels functions like the ones introduced above in the simple IPM section. These include, for example, survival/growth and fecundity kernels of existing plants, or survival/growth kernels of adult insects structured by body length. It is worth noting that $i$ and $j$ don't need to be the same trait here. For example, trees may have a continuously structured seedling stage where seedlings are structured by height, and a continuously structured adult stage where adults are structured by diameter at breast height (DBH). In this case, seedlings may start a transition with height $z_j$ at $t$ and become adults with DBH $z'_i$ at $t+1$.

There are two assumptions made by the general model: i) there are a finite number of domains, and ii) all of the continuous domains are bounded. The first assumption is not overly restrictive, because states like ontogeny only have a few possible values. Age is bounded by the maximum lifespan of the species, which need not depend on rate of senescence (which some species do escape) but on the maximum observed age in the data used to parameterize the model (incorporating the biological reality that everything dies eventually). The second assumption is not restrictive unless the state in question is spatial location (in which case, see

Ellner et al. 2016, Ch 8). Most species have a maximum size they can grow to (or, at least, a maximum size they regularly attain before something else kills them), and so imposing a closed domain on the model will not create unrealistic populations (Ellner, Childs & Rees 2016, Ch 6.3).

General IPMs unlock an even broader range of applications, some going beyond traditional ecological questions. For example, IPMs have been embedded into the SIR epidemiological model to estimate onward transmission probability of malaria parasites based on within-host parasite load dynamics (Metcalf et al. 2015). Stepping back into the realm of more traditional ecology, Bruno et al. (2011) used a general IPM to model population dynamics of corals infected with a fungal pathogen. Invasion biology has also benefited from general IPMs, with a multitude of publications addressing mechanisms underlying their rapid population growth (e.g. Crandall & Knight 2017, Levin et al. 2020), rate of spread (e.g. Jongejans et al. 2011), probability of invasion given future climate (e.g. Merow et al. 2017), and potential control pathways (e.g. Erickson et al. 2017, Lommen et al. 2018). General IPMs have proven useful for quantifying how biotic (e.g. Adler et al. 2010, Simmonds et al. 2020) and abiotic interactions (Rees & Ellner 2009, Compagnoni et al. 2021) influence population dynamics. This is far from an exhaustive list, but hopefully illustrates the range of possibilities along the theoretical to applied continuum.

## 1.4 Survey of available tools for IPM implementation

In parallel with developments in IPM theory, researchers have produced tools to assist with their implementation. For example, there are numerous "how-to" guides covering implementation of environmental variation (Metcalf et al. 2015), a practical guide for conservation biologists (Merow et al. 2013), and at least two *R* packages to assist with creating some types of IPMs (Metcalf et al. 2013, Shefferson et al. 2021). Additionally, a book published by Ellner, Childs & Rees (2016) contains in-depth examinations of the theory underlying various IPM applications, and provides free *R* code demonstrating how to implement these with real data.

Additionally, at least two *R* packages exist to help with fitting and implementing IPMs. `IPMPack` assists with both vital rate regression modelling and IPM implementation (Metcalf et al. 2013). It can handle a variety of IPM types, including simple IPMs, a subset of general IPMs, and includes functionality for specifying stochastic variation and conducting a variety of analyses, including life table response experiments, computing life expectancies, passage times, population growth rates, and perturbation based analyses. Furthermore, it contains excellent diagnostic tools to help users understand how their model is performing, and whether it may contain common flaws (e.g. lots of eviction, unrealistic size ranges, etc). `lefko3` is an *R* package offering a similar level of abstraction as `IPMPack` (i.e. handles both statistical modelling and IPM implementation). `lefko3` is designed specifically to help with analyses that incorporate an individual's history into the future state (i.e. $n(z, t+1) = f(n(z,t), n(z,t-1))$, Shefferson et al. 2021). It is more general than `IPMPack` in that it can handle a greater range of data non-independence (i.e. with mixed effects models) and a greater range of distributions for $z$. Both packages require a fixed set of vital rate functional forms, and rely on other *R* packages, such as `nlme`, `lme4`, `VGAM`, and `glmmTMB`. This introduces a tradeoff between perceived ease of use on the one hand, and flexibility in modeling single vital rates (and by extension, the life cycle as a whole) on the other hand. This tradeoff represents a substantial gap in the available tooling at our disposal, because pressing research questions demand more flexibility in IPM parameterization and underlying functional forms.

Much of this required complexity arises from the need to faithfully represent vital rates and their dependence not just on the trait (or traits) in the IPM, but external environmental factors as well. These external factors can include (but are not limited to) other species or changing climate conditions. Furthermore, the responses need not be linear (in fact, they often may not be). These issues are actually a subset of general regression modelling problems, and the field of computational statistics is constantly developing new tools to address them. For example, `mgcv` implements generalized additive models to handle estimation of semi-parametric smooth functions using maximum likelihood (Wood 2017). `lme4` similarly uses maximum likelihood to estimate linear mixed models to handle non-independence in the underlying data (e.g. from spatial clustering, phylogenetic structure, or nested experimental design, Bates et al. 2015). `brms` has become a popular interface to the `Stan` library for estimating Bayesian models with Markov Chain Monte Carlo algorithms (Buerkner 2018, Stan Development Team 2020). `brms` is particularly popular because of it provides a familiar *R* formula interface and allows users to specify nearly any type of model, linear or non-linear, that they desire. It also

leverages `mgcv`'s spline basis-generating functions (and the duality of certain spline bases as random effects) to estimate GAMs in a fully Bayesian framework.

There is an astounding array of additional tools researchers may use to estimate trait-demography relationships, and this is only a brief overview. However, population biologists do not have a framework that allows us to harness the full breadth of these tools in an IPM context without writing all analysis code from scratch. This is problematic because 1) untested IPM code may contain bugs, 2) it potentially creates barriers to access for early career researchers who may not have a strong support network to teach them how to create IPMs, and 3) it creates problems in performing syntheses because we have no common data structure to represent IPMs and their underlying functional forms.

## 1.5 Demographic data for broader syntheses

Given the age and maturity of modelling structured populations, researchers have become interested in compiling databases of published literature for synthesis applications. COMPADRE (Salguero-Gomez et al. 2014), COMADRE (Salguero-Gomez et al. 2015), and DatLIFE (DatLIFE 2018) are the most directly relevant efforts for this dissertation. Others, including popler (Compagnoni et al. 2019), BIEN (Maitner et al. 2017), and $D^3$ (Hintze et al. 2013) are all important sources of information, but do not necessarily incorporate population structure or the entire life cycle of the species into their data. Still others, such as the HMD (Human Mortality Database 2022), HLD (Human Lifetable Database 2022), and IPUMS Terra (Ruggles et al. 2018) are important contributions, but limit their phylogenetic scope to *Homo sapiens*.

COMPADRE and COMADRE are the most closely linked databases to this dissertation. Their history stretches back to the late 1980s, when Silvertown and Franco began archiving published MPMs as time- and space-averaged matrix models. The synthetic rewards included, for example, the concept of the fast-slow continuum of plant strategies (Franco & Silvertown 1997), a broad examination of the relative contributions of vital rates to population growth rates (Silvertown et al. 1993), and a synthesis of the evolution of senesence in plants (Silvertown et al. 2001). Researchers spent much of the 2000s working in parallel to compile similar data sets until unifying under the umbrella of COMPADRE (and expanding to mammals via COMADRE) in 2011 (Salguero-Gomez et al. 2014). In subsequent years, these databases have been enhanced with additional data and metadata to assist researchers in linking these data to other data sources to address increasingly complicated questions. Since becoming open access in 2014 (COMPADRE) and 2015 (COMADRE), they have been cited 157 and 77 times, respectively. Clearly, these data sources have provided a substantial return on investment. Some of these returns include quantifiying the diversity of rates of ageing (Jones et al. 2014), plant and mammal species responses to climate change (Compagnoni et al. 2021, Paniw et al. 2021), and the expansion of the fast-slow continuum theory to include a reproductive strategy axis (Salguero-Gomez et al. 2015).

IPM usage is growing rapidly (Levin et al. 2021). IPMs are now the tool of choice to analyze populations that are structured by continuous traits, because they alleviate the need to construct arbitrary bins that discretize these continuous traits (e.g. create size classes for a tree species, Ellner & Rees 2006). Given the success of MPM repositories, a database of IPMs is necessary for synthetic studies in ecology. However, there are unique challenges posed in implementing such a database. Existing approaches to digitizing MPMs entail entering the iteration matrix/sub-matrix elements into databases (i.e. entering that actual value of the matrix cell). This is feasible for MPMs, since most do not have many rows and columns (the majority of COMPADRE and COMADRE matrices are less than $20 \times 20$). Furthermore most transition elements are on the order of $[0.01, 1]$, making them easier to enter by hand. IPM iteration matrices are much higher dimensional (on the order of $50 \times 50 - 1,000 \times 1,000$), and the transition elements can be much smaller (i.e. $[1^{-10}, 1^{-2}]$). Entering the latter by hand would inevitably result in errors, and might not even be possible since researchers rarely publish raw projection kernels (SC Levin, personal observation). Finally, capturing the IPM's underlying functional form, rather than just the iteration matrices/sub-matrices, enables more applications. Thus, in addition to the usual metadata and parameter value digitization that comes with creating a database, there are two extra tasks that must be completed to have a working IPM database:

1. Define a syntax to represent IPMs symbolically in a database. This representation needs to be flexible

enough to accommodate models across the range of complexities in the existing literature, and able to handle as much forthcoming, additional complexity as possible.

2. Implement an engine that can transform this database representation into something that's actually useful for a researcher who is not interested in the underlying technical details, and simply wants to focus on their research questions (i.e. most researchers). Given the dominance of *R* (R Core Development Team 2022) in the field of ecology, this would be the logical choice for such an engine. However, to facilitate including a broader array of backgrounds, bindings to other languages would be ideal. Thus, the syntax in Objective 1 should not be overly specific to *R*, and follow more general mathematical notation.

## 1.6 IPMs and consideration of scale

Given the range of applications for IPMs, it is worth stepping back and considering the various scales they might function across and how they have been applied to these scales so far. When using the term scale, I am referring broadly to time, space, and phylogeny. Within each of these, there are differing ways in which they apply:

1. Temporal scale, with respect to IPMs, usually comes in two flavors - the time step of the model iteration (i.e. the amount of actual time between $t$ and $t+1$), and the time frame of the total projection (i.e. stochastic models versus transient dynamic models versus asymptotic dynamics in deterministic models). In the former, the value is usually determined either by the speed in which the organism completes its life cycle, or, by the time step that has a meaningful biological interpretation. The latter is largely determined by the research question. Stochastic and asymptotic dynamics have dominated the literature over the years, but there is growing appreciation of the importance of transient dynamics.

2. Spatial scale, with respect to IPMs, typically refers to either the extent of a population, or the number of populations studied and how far apart they are, or both. The former can very greatly across study species. For example, the sampling grain for a population of *Canis lupus* (Yellowstone National Park, Coulson et al. 2011) is very different from the sampling grain for *Carduus nutans* ($1m \times 1m$ plots, Levin et al. 2019). For the latter, there are numerous considerations, largely driven by the research question. For example, consideration of competitive interactions across many sites would require separate models, as competitive pressure decays rapidly with distance between individuals. Consideration of weather drivers of population dynamics requires sampling a broad spatial extent at small spatial grains to capture variation in both weather and population performance. This is far from an exhaustive list, and is only meant to illustrate that space must be considered.

3. Phylogenetic scale, with respect to IPMs, is not one that is often considered. However, it is a useful concept, as many questions in ecology and evolution seek to generalize from species to genus to family, etc. The vast, vast majority of IPMs quantify dynamics of one species. Some may include feedbacks between species (e.g. Bruno et al. 2011, Adler et al. 2010). However, there is some work that examines "family-level" demography (Traill et al. 2021).

Invasive species, in addition to presenting a pressing ecological (Duenas et al. 2021) and economic threat (Pimentel et al. 2005), present an ideal system to examine all three of these meanings of scale. Species that become invasive usually undergo rapid spread, presenting a series of transient dynamics at the leading edge, while converging to asymptotic dynamics in the older center of invasion (Powell et al. 2011). The spatial scale of invasions can be viewed as local, in which the individuals in the population come to dominate their immediate area, and large extents (e.g. regional, continental) into which they spread and establish larger populations (Powell et al. 2011). Phylogenetic scale is especially prominent in the context of invasion biology, with work spanning nearly every taxonomic level of organization from single species (Zucaratto et al. 2021), collections of species (e.g. Suehs et al. 2004), and even the entire angiosperm clade (Darwin 1859). Despite all of this, there is still much debate over how these various meanings of scale should apply to invasion biology.

## 1.7 Objectives of the dissertation

The first objective of this dissertation is to provide a consistent interface for implementing IPMs. `ipmr` (**Chapter 2**) provides an engine for implementing IPMs from symbolic representations. It imposes no restrictions on vital rate models - researchers may use the full breadth of *R*'s modelling capabilities. Furthermore, it provides tools for basic analyses, and is extensively documented with examples of more complicated ones. The latter point substantially reduces difficulty in beginning to use the package, and helps researchers focus on their research questions rather than details of programming. Chapter two is published as: Levin, S.C., Childs, D.Z., Compagnoni, A., Evers, S., Knight, T.M. & Salguero-Gomez, R (2021) ipmr: flexible implementation of Integral Projection Models in R. Methods in Ecology and Evolution 12(10): 1826-1834.

The second objective of this dissertation is to provide a database of peer-reviewed published IPMs that researchers may use for large scale synthesis. This breaks down into a two sub-objectives:

1. Provide a central repository with consistent representations of peer-reviewed published IPMs. This is provided via PADRINO (**Chapter 3**).
2. Provide a tool that abstracts away the details of the database and enables researchers to easily subset and reproduce these IPMs. This is provided via `Rpadrino` (**Chapter 3**) and `ipmr` (**Chapter 2**).

PADRINO provides a repository for IPMs with extensive metadata. It also contains symbolic representations of the IPMs and the parameters used to implement them. It can store determinstic or stochastic, density/frequency-dependent or -independent, and simple or general IPMs. The syntax used to store the IPMs symbolically mirrors the mathematical notation for them. `Rpadrino` provides an interface to download PADRINO, subset PADRINO, and translate PADRINO's syntax into valid `ipmr` code. `ipmr` is then used as backend for reconstructing IPM objects. Chapter 3 is published as: Levin, S.C., Evers, S., Potter, T., Guerrero, M.P., Childs, D.Z., Compagnoni, A., Knight, T.M. & Salguero-Gomez, R. (2022) Rpadrino: an R package to access and use PADRINO, an open access database of Integral Projection Models. Methods in Ecology and Evolution. DOI: https://doi.org/10.1111/2041-210X.13910.

**Chapter 4** of this dissertation examines how environmental drivers affect demography of invasive species at both local and broad spatial scales using the *Carpobrotus* genus. I used drones to collect demographic data from 13 sites on four continents across the native and invaded range. I develop an IPM using `ipmr` for each site that includes environmental covariates to show how *Carpobrotus* responds to environmental drivers. Finally, I use a Life Table Response Experiment to quantify the contributions of climate drivers to overall fitness.

In **Chapter 5**, I provide a synthesis of findings and recommendations for future research.

### Citations


1. Adler, P.B., Ellner, S.P. & Levine, J.M. (2010). Coexistance of perennial plants: an embarassment of niches. Ecology Letters 13: 1019-1029. https://doi.org/10.1111/j.1461-0248.2010.01496.x
2. Bassar, R.D., Childs, D.Z., Rees, M., Tuljapurkar, S., Reznick, D.N. & Coulson, T. (2016) The effects of asymmetric competition on the life history of Trinidadian guppies. Ecology Letters 19(3): 268-278.
3. Bogdan, A., Levin, S.C., Salguero-Gómez, R., Knight, T.M. (2021). Demographic analysis of Israeli Carpobrotus populations: management strategies and future directions. PLoS ONE 16(4): e0250879. https://doi.org/10.1101/2020.12.08.415174
4. Caswell, H. (1978). A general formula for the sensitivity of population growth rate to changes in life history parameters. Theoretical Population Biology 14(2): 215-230. https://www.jstor.org/stable/2528566
5. Caswell, H. (1989). Matrix Models 1st edition.

6. Childs, D.Z., Rees, M., Rose, K.E., Grubb, P.J., & Ellner, S.P. (2004). Evolution of size-dependent flowering in a variable environment: construction and analysis of a stochastic integral projection model. Proceedings of the Royal Society B 271(1547): 425-434. https://doi.org/10.1098/rpsb.2003.2597

7. Compagnoni, A., Levin, S.C., Childs, D.Z., Harpole, S., Paniw, M., Roemer, G., Burns, J.H., Che-Castaldo, J., Rueger, N., Kunstler, G., Bennett, J.M., Archer, C.R., Jones, O.R., Salguero-Gómez, R., & Knight, T.M. (2021). Herbaceous perennial plants with short generation time have stronger responses to climate anomalies than those with longer generation time. Nature Communications 12: 1824. https://doi.org/10.1038/s41467-021-21977-9

8. Compagnoni, A., Bibian, A.J., Ochocki, B.M., Rogers, H.S., Schultz, E.L., Sneck, M.E., Elderd, B.D., Inouye, D.W., Jacquemyn, H. & Miller, T.E.X. (2016) The effect of demographic coorelations on the stochastic population dynamics of perennial plants. Ecological Monographs 86(4): 480-494.

9. Compagnoni, A., Bibian, A.J., Ochocki, B.M., Levin, S.C., Zhu, K. & Miller, T.E.X. (2019) popler: an R package for extraction and synthesis of population time series from long-term ecological research (LTER) network. Methods in Ecology and Evolution 11(2): 258-264.

10. Coulson, T., MacNulty, D.R., Stahler, D.R., von Holdt, B., Wayne, R.K., & Smith, D.W. (2011). Modeling effects of environmental change on wolf population dynamics, trait evolution, and life history. Science 334(6060): 1275-1278. https://doi.org/10.1126/science.1209441

11. Crandall, R.M. & Knight, T.M. (2017). Role of multiple invasion mechanisms and their interaction in regulating the population dynamics of an exotic tree. Journal of Applied Ecology 55(2):885-894. https://doi.org/10.1111/1365-2664.13020

12. Crone, E.E., Menges, E.S., Ellis, M.M., Bell, T., Bierzychudek, P, Ehrlen, J. et al. (2011) How do ecologists use matrix population models? Ecology Letters 14(1): 1-8. DOI: https://doi.org/10.1111/j.1461-0248.2010.01540.x

13. Darwin, C. (1859) On the Origin of Species By Means of Natural of Selection. John Murray: UK.

14. Doak, D. & Morris W.F. (2002). Quantitative Conservation Biology: Theory and Practice of Population Viability Analysis. Oxford, UK: Oxford University Press

15. Easterling, M.R., Ellner, S.P., & Dixon, P.M. (2000). Size-specific sensitivity: Applying a new structured population model. Ecology 81(3): 694-708. https://doi.org/10.1890/0012-9658(2000)081%5B0694:SSSAAN%5D2.0.CO;2

16. Ebert, T.A. (1999). Plant and Animal Populations: Methods in Demography. Academic Press, San Diego, California.

17. Ellner, S.P. & Rees, M. (2006). Integral Projection Models for species with complex demography. The American Naturalist 167(3): 410-428.

18. Ellner, S.P., Childs, D.Z., & Rees, M. (2016) Data driven modelling of structured populations: a practical guide to the Integral Projection Model. Springer, Switzerland.

19. Erickson, R.A., Eager, E.A., Brey, M.B., Hansen, M.J., & Kocovsky, P.M. (2017). An integral projection model with YY-males and application to evaluating grass carp control. Ecological Modelling 361: 14-25. https://doi.org/10.1016/j.ecolmodel.2017.07.030

20. Franco, M. & Silvertown, J. (1997) Life History Variation in Plants: An Exploration of the Fast-Slow Continuum Hypothesis. Cambridge University Press, Cambridge, UK, New York, NY, USA.

21. Franklin, B. (1751) Observations concering the increase of mankind, peopling of countries, etc.

22. Ferrer-Cervantes, M.E., Mendez-Gonzalez, M.E., Quintana-Ascencio, P-F., Dorantes, A., Dzib, G., & Duran, R. (2012). Population dynamics of the cactus *Mammillaria gaumeri*: an integral projection model approach. Population Ecology 54: 321-334. DOI: https://doi.org/10.1007/s10144-012-0308-7

23. Goodman, L.A. (1969) The analysis of population growth when the birth and death rates depend upon several factors. Biometrics 25(4): 659-681.

24. Jensen, A.L. (1995) Simple density-dependent matrix for population projection. Ecological modelling 77: 43-48.

25. Jones, O.R., Scheuerlein, A., Salguero-Gomez, R., Camarda, C.G., Schaible, R., Casper, B.P., *et al.* (2014). Diversity of ageing across the tree of life. Nature 505: 169-173.

26. Jongejans, E., Shea, K., Skarpaas, O., Kelly, D., & Ellner, S.P. (2011). Importance of individual and environmental variation for invasive species spread: a spatial integral projection model. Ecology 92(1): 86-97. https://doi.org/10.1890/09-2226.1

27. de Kroon, H., Plaisier A., van Groenendael, J., & Caswell, H. (1986). Elasticity: the relative contribution of demographic parameters to population growth rate. Ecology 67(5): 1427-1431. https://doi.org/10.2307/1938700.

28. Human LifetableDatabase. University of California, Berkeley (USA), and Max Planck Institute for Demographic Research (Germany). Available at www.lifetable.de.

29. Human Mortality Database. University of California, Berkeley (USA), and Max Planck Institute for Demographic Research (Germany). Available at www.mortality.org or www.humanmortality.de.

30. Lefkovitch, L.P. (1965). The study of population growth in organisms grouped by stages. Biometrics 21(1): 1-18. https://www.jstor.org/stable/2528348

31. Leslie, P.H. (1945) The use of matrices in certain population mathematics. Biometrika, 33(3): 183–212.

32. Levin, S.C., Crandall, R.M. & Knight, T.M. (2019) Population projection models for 14 alien plant species in the presence and absence of aboveground competition. Ecology 100(6): e02681.

33. Levin, S.C., Crandall, R.M., Pokoski, T., Stein, C. & Knight, T.M. (2020) Phylogenetic and functional distinctiveness explain alien plant population responses to competition. Proceedings of the Royal Society B 287: 20201070.

34. Levin, S.C., Childs, D.Z., Compagnoni, A., Evers, S., Knight, T.M. & Salguero-Gomez, R (2021) ipmr: flexible implementation of Integral Projection Models in R. Methods in Ecology and Evolution 12(10): 1826-1834.

35. Liu, L. & Cohen, J.E. (1987). Equilibrium and local stability in a logistic matrix model for age-structured populations. Journal of Mathematical Biology 25: 73-88.

36. Lommen, S.T.E., Jongejans, E., Leitsch-Vitalos, M., Tokarska-Guzik, B., Zalai, M., Mueller-Schaerer, H. & Karrer, G. (2018) Time to cut: population models reveal how to mow invasive common ragweed cost-effectively. NeoBiota 39: 53-78. DOI: 10.3897/neobiota.39.23398

37. Lotka, A.J. (1925) Elements of Physical Biology, Williams and Wilkins.

38. Maitner B.S., Boyle, B., Casler, N., Condit, R., Donoghue, J., Duran, S.M. et al. (2017) The BIEN R package: a tool to access the Botanical Information and Ecology Network database. Methods in Ecology and Evolution 9(2): 373-379.

39. Hintze C., Heydel, F., Hoppe, C., Cunze, S., Koenig, A. & Tackenberg, O. (2013). D^3: The Dispersal and Diaspore Database - baseline data and statistics on seed dispersal. Perspectives in Plant Ecology, Evolution, and Systematics 15(3): 180-192.

40. Malthus, T. (1798) An essay on the principle of population. J. Johnson, London, England.

41. McKendrick, A.G. (1926) Applications of mathematics to medical problems. Proceedings of the Edinburgh Mathematical Society 44: 98-130

42. McKendrick, A.G. & Kesava Pai, M. (1911). The Rate of Multiplication of Micro-organisms: A Mathematical Study. Proceedings of the Royal Society of Edinburgh. 31: 649–653. doi: 10.1017/S0370164600025426.

43. Merow, C., Dahlgren, J.P., Metcalf, C.J.E., Childs, D.Z., Evans, M.E.K., Jongejans, E., Record, S., Rees, M., Salguero-Gomez R., & McMahon, S.M. (2014). Advancing population ecology with integral projection models: a practical guide. Methods in Ecology and Evolution 5: 99-110. https://doi.org/10.1111/2041-210X.12146S

44. Merow, C., Bois, S.T., Allen, J.M., Xie, Y. & Silander Jr., J.A. (2017) Climate change both facilitates and inhibits invasive plant ranges in New England. Proceedings of the National Academy of Sciences 114(16): E3276-E3284.

45. Metcalf et al. (2009) A time to grow and a time to die:

46. Metcalf, C.J.E., Ellner, S.P., Childs, D.Z., Salguero-Gómez, R., Merow, C., McMahon, S.M., Jongejans, E., & Rees, M. (2015). Statistical modelling of annual variation for inference on stochastic population dynamics using Integral Projection Models. Methods in Ecology and Evolution 6(9): 1007-1017. https://doi.org/10.1111/2041-210X.12405

47. Metcalf, C.J.E., Graham, A.L., Martinez-Bakker, M. & Childs, D.Z. (2015) Opportunities and challenges of Integral Projection Models for modelling host-parasite dynamics. Journal of Animal Ecology. DOI: 10.1111/1365-2656.12456

48. Metcalf, C.J.E., McMahon, S. M., Salguero-Gómez, R. & Jongejans, E. (2013). IPMpack: an R package for integral projection models. Methods in Ecology and Evolution. 4(2): 195-200. https://doi.org/10.1111/2041-210x.12001

49. Metcalf, C.J.E. & Pavard, S. (2007) Why evolutionary biologists should be demographers. Trends in Ecology and Evolution 22(4): 205-212. DOI: 10.1016/j.tree.2006.12.001.

50. Nicole, F., Dahlgren, J.P., Vivat, A., Till-Bottraud, I. & Ehrlen, J. (2011). Interdependent effects of habitat quality and climate on population growth of an endangered plant. Journal of Ecology 99: 1211-1218.

51. Rees, M. & Ellner, S.P. (2009) Integral projection models for populations in temporally varying environments. Ecological Monographs 79(4): 575-594.

52. Rose, K.E., Louda, S.M. & Rees, M. (2005) Demographic and evolutionary impacts of native and invasive insect herbivores on *Cirsium canescens*. Ecology 86(2): 453-465.

53. R Core Team (2021). R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. URL https://www.R-project.org/.

54. Ruggles, S., Manson, S.M, Kugler, T.A., Haynes, D.A., Van Riper, D.C., Bakhtsiyarava, M. (2018). IPUMS Terra: Integrated data on population and environment: Version 2. Minneapolis, MN: IPUMS.

55. Salguero-Gómez, R., Jones, O.R., Archer, C.R., Bein, C., de Buhr, H., Farack, C., Gottschalk, F., Hartmann, A., Henning, A., Hoppe, G., Roemer, G., Ruoff, T., Sommer, V., Wille, J. Voigt, J., Zeh, S., Vieregg, D., Buckley, Y.M., Che-Castaldo, J., Hodgson, D., et al. (2016) COMADRE: a global database of animal demography. Journal of Animal Ecology 85: 371-384. https://doi.org/10.1111/1365-2656.12482

56. Salguero-Gómez, R, Jones, O.R., Archer, C.A., Buckley, Y.M., Che-Castaldo, J., Caswell, C., Hodgson, D., Scheuerlein, A., Conde, D.A., Brinks, E., de Buhr, H., Farack, C., Gottschalk, F., Hartmann, A., Henning, A., Hoppe, G., Roemer, G., Runge, J., Ruoff, T., et al. (2014) The COMPADRE Plant Matrix Database: an online repository for plant population dynamics. Journal of Ecology 103: 202-218. https://doi.org/10.1111/1365-2745.12334

57. Salguero-Gómez, R., Jones, O.R., Jongejans, E., Blomberg, S.P., Hodgson, D.J., Mbeau-Ache, C., Zuidema, P., de Kroon, H., & Buckley, Y.M. (2015). Fast-slow continuum and reproductive strategies

structure plant life history variation worldwide. Proceedings of the National Academy of Sciences 113(1): 230-235.

58. Silvertown, J., Franco, M. & Perez-Ishiwara, R. (2001) Evolution of senescence in iteroparous perennial plants. Evolutionary Ecology Research, 3, 393–412.

59. Silvertown, J., Franco, M., Pisanty, I. & Mendoza, A. (1993) Comparative plant demography: relative importance of life-cycle components to the finite rate of increase in woody and herbaceous perennials. Journal of Ecology, 81, 465–476.

60. Simmonds E.G., Cole, E.F., Sheldon, B.C., & Coulson, T. (2020). Phenological asynchrony: a ticking time-bomb for seemingly stable populations? Ecology Letters 23(12): 1766-1775. https://doi.org/10.1111/ele.13603

61. Shefferson, R.P., Kurokawa, S., & Ehrlen, J. (2020). LEFKO3: analysing individual history through size-classified matrix population models. Methods in Ecology and Evolution. https://doi.org/10.1111/2041-210X.13526

62. Teller, B.J., Adler, P.B., Edwards, C.B., Hooker, G. & Ellner, S.P. (2016) Linking demography with drivers: climate and competition. Methods in Ecology and Evolution 7: 171-183. DOI: 10.1111/2041-210X.12486

63. Traill, L., Plard, F., Galliard, J-M. & Coulson, T. (2021). Can we use a functional trait to construct a generalized model for ungulate populations? Ecology 102(4): e03289.

64. Vandermeer, J. (1978). Choosing category size in a stage projection matrix. Oecologia 32: 79-84.

65. Verhulst, P.H. (1838) Notice sur la loi que la population poursuit dans son accroissement. Corresp. mathématique et physique 10, 113–121

66. Vindenes, Y. Engen, S. & Saether, B-E. (2011) Integral projection models for finite populations in a stochastic environment. Ecology 92(5): 1146-1156.

67. Volterra, V. (1926) "Variazioni e fluttuazioni del numero d'individui in specie animali conviventi", Mem. Acad. Lincei Roma, 2, 31–113.

68. Zucarratto, R., Pires, A.S., Bergallo, H.G. & Portela, R.de C.Q. (2020) Felling the giants: integral projection models indicate adult management to control an exotic invasive palm. Plant Ecology. DOI: 10.1007/s11258-020-01090-5

69. Suehs, C.M., Affre, L., & Medail, F. (2004). Invasion dynamics of two alien *Carpobrotus* (Aizoaceae) taxa on a Mediterranean island: I. Genetic diversity and introgression. Heredity 92, 31–40. doi: 10.1038/sj.hdy.6800374

70. Powell, K.I., Chase, J.M., & Knight, T.M. (2011). A synthesis of plant invasion effects on biodiversity across spatial scales. American Journal of Botany 98: 539-548. https://doi.org/10.3732/ajb.1000402

71. Duenas, M-A., Hemming, D.J., Roberts, A., & Diaz-Soltero, H. (2021). The threat of invasive species to IUCN-listed critically endangered species: A systematic review. Global Ecology and Conservation 26: e01476 https://doi.org/10.1016/j.gecco.2021.e01476

72. Pimentel, D., Zuniga, R., & Morrison, D. (2005). Update on the environmental and economic costs associated with alien-invasive species in the United States Ecological Economics 82(3): 273-288. https://doi.org/10.1016/j.ecolecon.2004.10.002

# Chapter 2: `ipmr`: Flexible implementation of Integral Projection Models in R

Authors:

Sam C. Levin

Sanne Evers

Tomos Potter

Mayra Pena-Guerrero

Dylan Z. Childs

Aldo Compagnoni

Tiffany M. Knight

Roberto Salguero-Gomez

*Published in* Methods in Ecology and Evolution 12(10): 1826-1834.

https://doi.org/10.1111/2041-210X.13683

Received: 3 February 2021 | Accepted: 21 June 2021

DOI: 10.1111/2041-210X.13683

APPLICATION

Methods in Ecology and Evolution BRITISH ECOLOGICAL SOCIETY

# ipmr: Flexible implementation of Integral Projection Models in R

Sam C. Levin[1,2,3] | Dylan Z. Childs[4] | Aldo Compagnoni[1,2,3] | Sanne Evers[1,2,5] | Tiffany M. Knight[1,2,5] | Roberto Salguero-Gómez[3]

[1]Institute of Biology, Martin Luther University Halle-Wittenberg, Halle (Saale), Germany

[2]German Centre for Integrative Biodiversity Research (iDiv) Halle-Jena-Leipzig, Leipzig, Germany

[3]Department of Zoology, University of Oxford, Oxford, UK

[4]Department of Animal and Plant Sciences, University of Sheffield, Sheffield, UK

[5]Department of Community Ecology, Helmholtz Centre for Environmental Research-UFZ, Halle (Saale), Germany

**Correspondence**
Sam C. Levin
Email: levisc8@gmail.com

**Funding information**
R.S.-G. was supported by a NERC Independent Research Fellowship (NE/M018458/1). S.C.L., A.C., S.E. and T.M.K. were funded by the Alexander von Humboldt Foundation in the framework of the Alexander von Humboldt Professorship of TM Knight endowed by the German Federal Ministry of Education and Research.

**Handling Editor:** Giovanni Strona

## Abstract

1. Integral projection models (IPMs) are an important tool for studying the dynamics of populations structured by one or more continuous traits (e.g. size, height, body mass). Researchers use IPMs to investigate questions ranging from linking drivers to population dynamics, planning conservation and management strategies, and quantifying selective pressures in natural populations. The popularity of stage-structured population models has been supported by *R* scripts and packages (e.g. IPMpack, popbio, popdemo, lefko3) aimed at ecologists, which have introduced a broad repertoire of functionality and outputs. However, pressing ecological, evolutionary and conservation biology topics require developing more complex IPMs, and considerably more expertise to implement them. Here, we introduce ipmr, a flexible *R* package for building, analysing and interpreting IPMs.
2. The ipmr framework relies on the mathematical notation of the models to express them in code format. Additionally, this package decouples the model parameterization step from the model implementation step. The latter point substantially increases ipmr's flexibility to model complex life cycles and demographic processes.
3. ipmr can handle a wide variety of models, including those that incorporate density dependence, discretely and continuously varying stochastic environments, and multiple continuous and/or discrete traits. ipmr can accommodate models with individuals cross-classified by age and size. Furthermore, the package provides methods for demographic analyses (e.g. asymptotic and stochastic growth rates) and visualization (e.g. kernel plotting).
4. ipmr is a flexible *R* package for integral projection models. The package substantially reduces the amount of time required to implement general IPMs. We also provide extensive documentation with six vignettes and help files, accessible from an R session and online.

**KEYWORDS**
elasticity, integral projection model, life history, population dynamics, population growth rate, sensitivity, structured populations

'Tiffany M. Knight and Roberto Salguero-Gómez contributed equally to this work

## 1 | INTRODUCTION

Integral projection models (IPMs) are an important and widely used tool for ecologists studying structured population dynamics in discrete time. Since the paper introducing IPMs was published over two decades ago (Easterling et al., 2000), at least 255 peer-reviewed publications on at least 250 plant species and 60 animal species have used IPMs (ESM, Table S1; Figure S1). These models have addressed questions ranging from invasive species population dynamics (e.g. Crandall & Knight, 2017), effect of climate drivers on population persistence (e.g. Compagnoni et al., 2021), evolutionary stable strategies (e.g. Childs et al., 2004) and rare/endangered species conservation (e.g. Ferrer-Cervantes et al., 2012).

The IPM was introduced as alternative to matrix population models, which model populations structured by discrete traits (Caswell, 2001). Some of the advantages of using an IPM include (a) the ability to model populations structured by continuously distributed traits, (b) the ability to flexibly incorporate discrete and continuous traits in the same model (e.g. seeds in a seedbank and a height-structured plant population, Crandall & Knight, 2017, or number of females, males and age-1 recruits for fish species, Erickson et al., 2017), (c) efficient parameterization of demographic processes with familiar regression methods (Coulson, 2012), and (d) the numerical discretization of continuous kernels (see below) means that the tools available for matrix population models are usually also applicable for IPMs. Furthermore, researchers have developed methods to incorporate spatial dynamics (Jongejans et al., 2011), environmental stochasticity (Rees & Ellner, 2009) and density/frequency dependence into IPMs (Adler et al., 2010; Ellner et al., 2016). These developments were accompanied by the creation of software tools and guides to assist with IPM parameterization, implementation and analysis. These tools range from *R* scripts with detailed annotations (Coulson, 2012; Ellner et al., 2016; Merow et al., 2014) to *R* packages (Metcalf et al., 2013; Shefferson et al., 2020).

Despite the array of resources available to researchers, implementing an IPM is still not a straightforward exercise. For example, an IPM that simulates a population for 100 time steps requires the user to either write or adapt from published guides multiple functions (e.g. to summarize demographic functions into the proper format), implement the numerical approximations of the model's integrals, ensure that individuals are not accidentally sent beyond the integration bounds ('unintentional eviction', sensu Williams et al., 2012) and track how the population state changes over the course of a simulation. Stochastic IPMs present further implementation challenges. In addition to the aforementioned elements, users must generate the sequence of environments that the population experiences. There are multiple ways of simulating environmental stochasticity, each with their own strengths and weaknesses (Metcalf et al., 2015).

ipmr manages these key details while providing the user flexibility in their models. ipmr uses the rlang package for metaprogramming (Henry & Wickham, 2020), which enables ipmr to provide a miniature domain-specific language for implementing IPMs. ipmr aims to mimic the mathematical syntax that describes IPMs as closely as possible (Figure 1; Box 1; Tables 1 and 2). This *R* package can handle models with individuals classified by a mixture of any number of continuously and discretely distributed traits. Furthermore, ipmr introduces specific classes and methods to deal with both discretely and continuously varying stochastic environments, density-independent and -dependent models, as well as age-structured populations (Case Study 2). ipmr decouples the parameterization (i.e. regression model fitting) and implementation steps (i.e. converting the regression parameters into a full IPM), and does not attempt to help users with the parameterization task. This provides greater flexibility in modelling trait–demography relationships, and enables users to specify IPMs of any functional form that they desire.

## 2 | TERMINOLOGY AND IPM CONSTRUCTION

An IPM describes how the abundance and distribution of trait values (also called *state variables/states*, denoted $z$ and $z'$) for a population changes in discrete time. The distribution of trait values in a population at time $t$ is given by the function $n(z, t)$. A simple IPM for the trait distribution $z'$ at time $t + 1$ is then

$$n(z', t+1) = \int_L^U K(z', z)\, n(z, t)\, dz. \tag{1}$$

$K(z', z)$, known as the *projection kernel*, describes all possible transitions of existing individuals and recruitment of new individuals from $t$ to $t + 1$, generating a new trait distribution $n(z', t + 1)$. $L, U$ are the lower and upper bounds for values that the trait $z$ can have, which defines the *domain* over which the integration is performed. The integral $\int_L^U n(z, t)\, dz$ gives the total population size at time $t$.

To make the model more biologically interpretable, the projection kernel $K(z', z)$ is usually split into *sub-kernels* (Equation 2). For example, a projection kernel to describe a life cycle where individuals can survive, transition to different state values, and reproduce via sexual and asexual pathways, can be split as follows.

$$K(z', z) = P(z', z) + F(z', z) + C(z', z), \tag{2}$$

where $P(z', z)$ is a sub-kernel describing transitions due to survival and trait changes of existing individuals, $F(z', z)$ is a sub-kernel describing per-capita sexual contributions of existing individuals to recruitment and $C(z', z)$ is a sub-kernel describing per-capita asexual contributions of existing individuals to recruitment. The sub-kernels are typically comprised of functions derived from regression models that relate an individual's trait value $z$ at time $t$ to a new trait value $z'$ at $t + 1$. For example, the $P$ kernel for Soay sheep *Ovis aries* on St. Kilda (Equation 3) may contain two regression models: (a) a logistic regression of survival on log body mass (Equation 4) and (b)

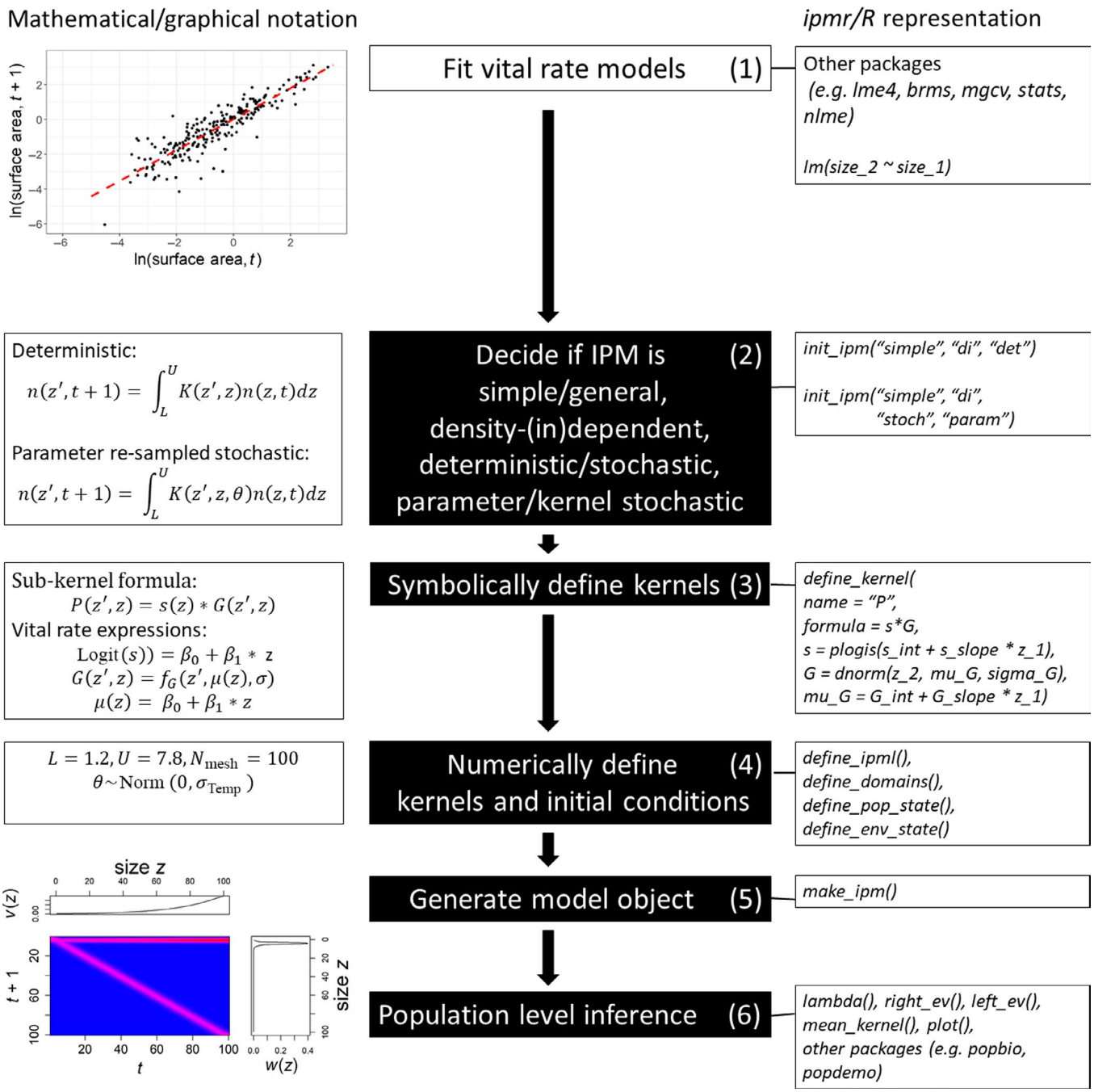



**FIGURE 1** There are generally six steps in defining an IPM with ipmr. (1) Vital rate models are fit to demographic data collected from field sites. This step requires the use of other packages, as ipmr does not contain facilities for regression modelling. The figure on the left shows the fitted relationship between size at $t$ and $t + 1$ for *Carpobrotus* spp. in Case Study 1. (2) The next step is deciding what type of IPM is needed. This is determined by both the research question and the data used to parameterize the regression models. This process is initiated with init_ipm(). In step (3), kernels are defined using ipmr's syntax to represent kernels and vital rate functions. (4) Having defined symbolic representations of the model, the numerical definition is given. Here, the integration rule, domain bounds and initial population conditions are defined. For some models, initial environmental conditions can also be defined. (5) make_ipm() numerically implements the proto_ipm object, (6) which can then be analysed further. The figure at the bottom left shows a $K(z', z)$ kernel created by make_ipm() and make_iter_kernel(). The line plots above and to the right display the left and right eigenvectors, extracted with left_ev() and right_ev(), respectively

a linear regression of log body mass at $t + 1$ on log body mass at $t$ (Equations 5–6). In this example, $f_G$ is a normal probability density function with $\mu_G$ given by the linear predictor of the mean, and with $\sigma_G$ computed from the standard deviation of the residuals from the linear regression model.

$$P(z', z) = s(z) * G(z', z), \quad (3)$$

$$\text{Logit}(s(z)) = \alpha_s + \beta_s * z, \quad (4)$$

$$G(z', z) = f_G(z', \mu_G, \sigma_G), \quad (5)$$

**BOX 1 Code to implement a simple IPM from parameter estimates in ipmr. Because ipmr does not include functions to assist with regression modelling, this example skips the step of working with actual data and instead uses hypothetical parameter values. We see that given this set of conditions, if nothing were to change, the population would increase by ~2% each year. The case studies provide details on further use cases and analyses that are possible with ipmr.**

```
library(ipmr)

# This section produces the result of Step 1 in Figure 1.

data_list <- list(
    s_i = -0.65,      # Intercept of the survival model (Logistic regression)
    s_z = 0.75,       # Slope of the survival model
    G_i = 0.96,       # Intercept of the growth model (Gaussian regression)
    G_z = 0.66,       # Slope of the growth model
    sd_G = 0.67,      # Standard deviation of residuals of growth model
    mu_r = -0.08,     # Mean of the recruit size distribution
    sd_r = 0.76,      # Standard deviation of the recruit size distribution
    r_n_i = -1,       # Intercept of recruit production model (Poisson regression)
    r_n_z = 0.3       # Slope of recruit production model.
)

#  Step 2 in Figure 1. This is how ipmr initializes a model object.
#  All functions prefixed with define_* generate proto_ipm objects. These
#  are converted into IPMs using the make_ipm() function in step 5.

example_proto_ipm <- init_ipm(sim_gen    = "simple",
                              di_dd      = "di",
                              det_stoch = "det")

# Step 3 in Figure 1. Note the link between how the model was defined
# mathematically and how it is defined here.

example_proto_ipm <- define_kernel(
    example_proto_ipm,
    name             = "P",
    formula          = surv * Grow,
    surv             = plogis(s_i + s_z * z_1),
    Grow             = dnorm(z_2, mu_G, sd_G),
    mu_G             = G_i + G_z * z_1,
    data_list        = data_list,
    states           = list(c("z"))
)

example_proto_ipm <- define_kernel(
    example_proto_ipm,
    name             = "F",
    formula          = recr_number * recr_size,
    recr_number      = exp(r_n_i + r_n_z * z_1),
    recr_size        = dnorm(z_2, mu_r, sd_r),
    data_list        = data_list,
    states           = list(c("z"))
)
```

**BOX 1 (Continued)**

```
# Step 4 in Figure 1. These next 3 functions define:
# 1. The numerical integration rules and how to iterate the
#    model (define_impl).
# 2. The range of values the the trait "z" can take on, and the number of
#    meshpoints to use when dividing the interval (define_domains).
# 3. The initial population state (define_pop_state).

example_proto_ipm <- define_impl(
  example_proto_ipm,
  list(
    P = list(int_rule = "midpoint", state_start = "z", state_end = "z"),
    F = list(int_rule = "midpoint", state_start = "z", state_end = "z")
  )
)

example_proto_ipm <- define_domains(
  example_proto_ipm,
  z = c(-2.65, 4.5, 250)  # format: c(L, U, m), m is number of meshpoints
)

example_proto_ipm <- define_pop_state(
  example_proto_ipm,
  n_z = rep(1/250, 250)
)

# Step 5 in Figure 1.

example_ipm <- make_ipm(example_proto_ipm)

# Step 6 in Figure 1.

lambda(example_ipm)
```

**TABLE 1** Translations between mathematical notation, R's formula notation and ipmr's notation for the simplified version of Bogdan et al.'s *Carpobrotus* IPM. The ipmr column contains the expressions used in each kernel's definition. R expressions are not provided for sub-kernels and model iteration procedures because they typically require defining functions separately, and there are many ways to do this step (examples are in the R code for each case study in the appendix). The plogis() function computes the inverse logit transformation of an expression. $s$ corresponds to survival, $G$ corresponds to change in size conditional on survival, $r_p$ is the probability of reproducing, $r_n$ is the number of propagules produced by reproductive individuals and $p_r$ is the probability that a propagule becomes a new recruit at $t + 1$

| Math formula | R formula | ipmr |
|---|---|---|
| $\mu_G = \alpha_G + \beta_G * z$ | size_2 ~ size_1, family =gaussian() | mu_G = G_int + G_slope * z |
| $G(z', z) = f_G(z', \mu_G, \sigma_G)$ | G = dnorm(z_2, mu_G, sd_G) | G = dnorm(z_2, mu_G, sd_G) |
| $\text{logit}(s(z)) = \alpha_s + \beta_s * z$ | surv ~size_1, family = binomial() | s = plogis(s_int + s_slope * z) |
| $\log(r_n(z)) = \alpha_{r_n} + \beta_{r_n} * z$ | fec ~size_1, family = poisson() | r_n = exp(r_n_int + r_n_slope * z) |
| $\text{logit}(r_p(z)) = \alpha_{r_p} + \beta_{r_p} * z$ | repr ~ size_1, family = binomial() | r_p = plogis(r_p_int + r_p_slope * z) |
| $r_d(z') = f_{r_d}(z', \mu_{r_d}, \sigma_{r_d})$ | dnorm(z_2, mu_f_d, sigma_f_d) | r_d = dnorm(z_2, f_d_mu, f_d_sigma) |
| $p_r = \frac{\#\ \text{Recruits}(t+1)}{\#\ \text{flowers}(t)}$ | p_r = n_new_recruits / n_flowers | p_r = n_new / n_flowers |
| $P = s(z) * G(z', z)$ | | P = s * G |
| $F(z', z) = r_p(z) * r_n(z) * r_d(z') * p_r$ | | F = r_p * r_n * r_d * p_r |
| $n(z', t+1) = \int_L^U [P(z', z) + F(z', z)]n(z, t)\,dz$ | | |

**TABLE 2** Translations between mathematical notation, R's formula notation and ipmr's notation for Ellner et al. (2016) *Ovis aries* IPM. The ipmr column contains the expressions used in each kernel's definition. R expressions are not provided for sub-kernels and model iteration procedures because they typically require defining functions separately, and there are many ways to do this step (examples are in the R code for each case study in the appendix). ipmr supports a suffix based syntax to avoid repetitively typing out the levels of discrete grouping variables. These are represented as 'a' in the Math column, 'age' in the R formula column, and are highlighted in bold in the ipmr column. $s$ corresponds to survival, $G$ corresponds to change in size conditional on survival, $m_p$ is the probability of mating, $r_p$ is the probability that a mating produces a new recruit at $t + 1$ and $B$ is the size distribution of new recruits at $t + 1$ whose mean depends on parent size at time $t$. $F_a$ is divided by 2 because this IPM only tracks females

| Math formula | R formula | ipmr |
|---|---|---|
| $\text{Logit}\left(s\left(z,a\right)\right) = \alpha_s + \beta_{s,z} * z + \beta_{s,a} * a$ | surv ~ size_1 + age, family = binomial() | s_age = plogis(s_int + s_z * z_1 + s_a * **age**) |
| $G\left(z', z, a\right) = f_G\left(z', \mu_G\left(z,a\right), \sigma_G\right)$ | G = dnorm(size_2, mu_G_age, sigma_G) | G_age = dnorm(z_2, mu_G_age, sigma_G) |
| $\mu_G\left(z,a\right) = \alpha_G + \beta_{G,z} * z + \beta_{G,a} * a$ | size_2 ~ size_1 + age, family = gaussian() | mu_G_age = G_int + G_z * z + G_a * **age** |
| $\text{Logit}\left(m_p\left(z,a\right)\right) = \alpha_{m_p} + \beta_{m_p,z} * z + \beta_{m_p,a} * a$ | repr ~size_1 + age, family = binomial() | m_p_age = plogis(m_p_int + m_p_z * z + m_p_a * **age**) |
| $\text{Logit}\left(r_p\left(a\right)\right) = \alpha_{r_p} + \beta_{r_p,a} * a$ | recr ~age, family = binomial() | r_p_age = plogis(r_p_int + r_p_a * **age**) |
| $B\left(z', z\right) = f_B\left(z', \mu_B\left(z\right), \sigma_B\right)$ | b = dnorm(size_2, mu_rc_size, sigma_rc_size) | rc_size = dnorm(z_2, mu_rc_size, sigma_rc_size) |
| $\mu_B\left(z\right) = \alpha_B + \beta_{B,z} * z$ | rc_size_2 ~ size_1, family = gaussian() | mu_rc_size = rc_size_int + rc_size_z * z |
| $P_a\left(z', z\right) = s\left(z,a\right) * G\left(z', z, a\right)$ | | P_age = s_age * g_age * d_z |
| $F_a\left(z', z\right) = s\left(z,a\right) * m_p\left(z,a\right) * r_p\left(a\right) * B\left(z', z\right)/2$ | | F_age = s_age * f_p_age * r_p_age * rc_size / 2 |
| $n_0\left(z', t+1\right) = \sum_{a=0}^{M+1} \int_L^U F_a\left(z', z\right) n_a\left(z,t\right) dz$ | | |
| $n_a\left(z', t+1\right) = \int_L^U P_{a-1}\left(z', z\right) n_{a-1}\left(z,t\right) dz$ | | |
| $n_{M+1}\left(z', t+1\right) = \int_L^U [P_{M+1}\left(z', z\right) n_{M+1}\left(z,t\right) + P_M\left(z', z\right) n_M\left(z,t\right)] dz$ | | |

$$\mu_G = \alpha_G + \beta_G * z. \tag{6}$$

Analytical solutions to the integral in Equation 1 are usually not possible (Ellner & Rees, 2006). However, numerical approximations of these integrals can be constructed using a numerical integration rule. A commonly used rule is the midpoint rule (more complicated and precise methods are possible and will be implemented, though are not yet, see Ellner et al., 2016, Chapter 6). The midpoint rule divides the domain $\left[L, U\right]$ into $m$ artifical size bins centered at $z_i$ with width $h = (U - L)/m$. The midpoints $z_i = L + (i - 0.5) * h$ for $i = 1, 2, \ldots, m$. The midpoint rule approximation for Equation 1 then becomes:

$$n\left(z_j, t+1\right) = h \sum_{i=1}^{m} K\left(z_j, z_i\right) n\left(z_i, t\right). \tag{7}$$

In practice, the numerical approximation of the integral converts the continuous projection kernel into a (large) discretized matrix. A matrix multiplication of the discretized projection kernel and the discretized trait distribution then generates a new trait distribution, a process referred to as *model iteration* (sensu Easterling et al., 2000).

Equations 1 and 2 are an example of a *simple IPM*. A critical aspect of ipmr's functionality is the distinction between *simple IPMs* and *general IPMs*. A simple IPM incorporates a single continuous state variable. Equations 1 and 2 represent a simple IPM because there is only one continuous state, $z$, and no additional discrete states. A general IPM models one or more continuous state variables, and/or discrete states. General IPMs are useful for modelling species with more complex life cycles. Many species' life cycles contain multiple life stages that are not readily described by a single state variable. Similarly, individuals with similar trait values may behave differently depending on environmental context. For example, Bruno et al. (2011) modelled aspergillosis impacts on sea fan coral *Gorgonia ventalina* population dynamics by creating a model where colonies were cross classified by tissue area (continuously distributed) and infection status (a discrete state with two levels—infected and uninfected). Coulson et al. (2010) constructed a model for Soay sheep where the population was structured by body weight (continuously distributed) and age (discrete state). Mixtures of multiple continuous and discrete states are also possible. Indeed, the vital rates of many species with complex life cycles are often best described with multivariate state distributions (Caswell & Salguero-Gómez, 2013). A complete definition of the simple/general distinction is given in Ellner et al. (2016, Chapter 6).

## 2.1 | A brief worked example of a simple IPM

Box 1 shows a brief example of how ipmr converts parameter estimates into an IPM. Perhaps the most frequently used metric derived from IPMs is the asymptotic per-capita population growth rate ($\lambda$, Caswell, 2001). When $\lambda > 1$, the population is growing,

while $\lambda < 1$ indicates population decline. ipmr makes deriving estimates of $\lambda$ straightforward. Box 1 demonstrates how to parameterize a simple, deterministic IPM and estimate $\lambda$. The example uses a hypothetical species that can survive and grow, and reproduce sexually (but not asexually, so $C(z', z) = 0$ in Equation 2). The population is structured by size, denoted $z$ and $z'$, and there is no seedbank.

The $P(z', z)$ kernel is given by Equation 3, and the vital rates therein by Equations 4–6. The $F(z', z)$ kernel is given Equation 8:

$$F(z', z) = r_d(z') * r_n(z), \tag{8}$$

$$r_d(z') = f_{r_d}(z', \mu_{r_d}, \sigma_{r_d}), \tag{9}$$

$$\text{Log}(r_n(z)) = \alpha_{r_n} + \beta_{r_n} * z. \tag{10}$$

Equation 9 is a recruit size distribution (where $f_{r_d}$ denotes a normal probability density function), and Equation 10 describes the number of new recruits produced by plants as a function of size $z$.

The code in Box 1 substitutes the actual probability density function (dnorm()) for $f_G$ and $f_{r_d}$, and uses inverse link functions instead of link functions. Otherwise, the math and the code should look quite similar.

## 2.2 | Case study 1: A simple IPM

One use for IPMs is to evaluate potential performance and management of invasive species in their non-native range (e.g. Erickson et al., 2017). Calculating sensitivities and elasticities of $\lambda$ to kernel perturbations can help identify conservation management strategies (Baxter et al., 2006; Caswell, 2001; de Kroon et al., 1986; Ellner et al., 2016). Bogdan et al. (2021) constructed a simple IPM for a *Carpobrotus* species growing north of Tel Aviv, Israel. The model includes four regressions, and an estimated recruit size distribution. Table 1 provides the mathematical formulae, the corresponding R model formulae and the ipmr notation for each one. The case study materials also offer an alternative implementation that uses the generic predict() function to generate the same output. The final part of the case study provides examples of functions that compute kernel sensitivity and elasticity, the per-generation growth rate, and generation time for the model, as well as how to visualize these results.

## 2.3 | Case study 2: A general age × size IPM

We use an age- and size-structured IPM from Ellner et al. (2016) to illustrate how to create general IPMs with ipmr. This case study demonstrates the suffix syntax for vital rate and kernel expressions, which is a key feature of ipmr (highlighted in bold in the 'ipmr' column in Table 2). The suffixes appended to each variable name in the ipmr formulation correspond to the subscript- and/or superscript used in the mathematical formulation. ipmr internally expands the model expressions and substitutes the range of ages and/or grouping variables in for the suffixes. This allows users to specify their model in a way that closely mirrors its mathematical notation, and saves users from the potentially error-prone process of re-typing model definitions many times or using for loops over the range of discrete states. The case study then demonstrates how to compute age-specific survival and fertility from the model outputs.

# 3 | DISCUSSION OF ADDITIONAL APPLICATIONS

We have shown above how ipmr handles a variety of model implementations that go beyond the capabilities of existing scripts and packages. The underlying implementation based on metaprogramming should be able to readily incorporate future developments in parameterization methods. Regression modelling is a field that is constantly introducing new methods. As long as these new methods have functional forms for their expected value (or a function to compute them, such as predict()), ipmr should be able to implement IPMs using them.

Finally, one particularly useful aspect of the package is the proto_ipm data structure. The proto_ipm is the common data structure used to represent every model class in ipmr and provides a concise, standardized format for representing IPMs. Furthermore, the proto_ipm object is created without any raw data, only functional forms and parameters. We are in the process of creating the PADRINO IPM database using ipmr and proto_ipms as an 'engine' to re-build published IPMs using only functional forms and parameter estimates. This database could act as an IPM equivalent of the popular COMPADRE and COMADRE matrix population model databases (Salguero-Gómez et al., 2016; Salguero-Gómez et al., 2014). Recent work has highlighted the power of syntheses that harness many structured population models (Adler et al., 2014; Compagnoni et al., 2021; Salguero-Gómez et al., 2016). Despite the wide variety of models that are currently published in the IPM literature, ipmr's functional approach is able to reproduce nearly all of them without requiring any raw data at all.

### ACKNOWLEDGEMENTS

We thank the Associate Editor and two anonymous reviewers for comments that greatly improved this manuscript.

### CONFLICTS OF INTEREST

The authors declare no conflicts of interest.

### AUTHORS' CONTRIBUTIONS

All authors contributed to package design. S.C.L. implemented the package. All authors wrote the first draft of the manuscript and contributed to revisions.

## PEER REVIEW

The peer review history for this article is available at https://publons.com/publon/10.1111/2041-210X.13683.

## DATA AVAILABILITY STATEMENT

The *Carpobrotus* dataset is included in the ipmr *R* package. The package is available on GitHub at https://github.com/levisc8/ipmr, CRAN at https://cran.r-project.org/web/packages/ipmr/index.html (Levin et al., 2021), and Zenodo at https://doi.org/10.5281/zenodo.5095062 (Levin, 2021). The paper and case studies do not use any other data.

## ORCID

*Sam C. Levin* https://orcid.org/0000-0002-3289-9925
*Dylan Z. Childs* https://orcid.org/0000-0002-0675-4933
*Aldo Compagnoni* https://orcid.org/0000-0001-8302-7492
*Sanne Evers* https://orcid.org/0000-0002-8002-1658
*Tiffany M. Knight* https://orcid.org/0000-0003-0318-1567
*Roberto Salguero-Gómez* https://orcid.org/0000-0002-6085-4433

## REFERENCES

Adler, P. B., Ellner, S. P., & Levine, J. M. (2010). Coexistance of perennial plants: An embarassment of niches. *Ecology Letters*, *13*, 1019–1029. https://doi.org/10.1111/j.1461-0248.2010.01496.x

Adler, P. B., Salguero-Gómez, R., Compagnoni, A., Hsu, J. S., Ray-Mukherjee, J., Mbeau-Ache, C., & Franco, M. (2014). Functional traits explain variation in plant life history strategies. *Proceedings of the National Academy of Sciences of the United States of America*, *111*(2), 740–745. https://doi.org/10.1073/pnas.1315179111

Baxter, P. W. J., McCarthy, M. A., Possingham, H. P., Menkhorst, P. W., & McLean, N. (2006). Accounting for management costs in sensitivity analyses of matrix population models. *Conservation Biology*, *20*(3), 893–905. https://doi.org/10.1111/j.1523-1739.2006.00378.x

Bogdan, A., Levin, S. C., Salguero-Gómez, R., & Knight, T. M. (2021). Demographic analysis of Israeli *Carpobrotus* populations: Management strategies and future directions. *PLoS ONE*, *16*(4), e0250879. https://doi.org/10.1101/2020.12.08.415174

Bruno, J. F., Ellner, S. P., Vu, I., Kim, K., & Harvell, C. D. (2011). Impacts of aspergillosis on sea fan coral demography: Modeling a moving target. *Ecological Monographs*, *81*(1), 123–139. https://doi.org/10.1890/09-1178.1

Caswell, H. (2001). *Matrix population models: Construction, analysis, and interpretation* (2nd ed.). Sinauer Associates Inc.

Caswell, H., & Salguero-Gómez, R. (2013). Age, stage and senescence in plants. *Journal of Ecology*, *101*(3), 585–595. https://doi.org/10.1111/1365-2745.12088

Childs, D. Z., Rees, M., Rose, K. E., Grubb, P. J., & Ellner, S. P. (2004). Evolution of size-dependent flowering in a variable environment: Construction and analysis of a stochastic integral projection model. *Proceedings of the Royal Society B: Biological Sciences*, *271*(1547), 425–434. https://doi.org/10.1098/rpsb.2003.2597

Compagnoni, A., Levin, S. C., Childs, D. Z., Harpole, S., Paniw, M., Roemer, G., Burns, J. H., Che-Castaldo, J., Rueger, N., Kunstler, G., Bennett, J. M., Archer, C. R., Jones, O. R., Salguero-Gomez, R., & Knight, T. M. (2021). Herbaceous perennial plants with short generation time have stronger responses to climate anomalies than those with longer generation time. *Nature Communications*, *12*, 1824. https://doi.org/10.1038/s41467-021-21977-9

Coulson, T. N. (2012). Integral projection models, their construction and use in posing hypotheses in ecology. *Oikos*, *121*, 1337–1350. https://doi.org/10.1111/j.1600-0706.2012.00035.x

Coulson, T., Tuljapurkar, S., & Childs, D. Z. (2010). Using evolutionary demography to link life history theory, quantitative genetics and population ecology. *Journal of Animal Ecology*, *79*, 1226–1240. https://doi.org/10.1111/j.1365-2656.2010.01734.x

Crandall, R. M., & Knight, T. M. (2017). Role of multiple invasion mechanisms and their interaction in regulating the population dynamics of an exotic tree. *Journal of Applied Ecology*, *55*(2), 885–894. https://doi.org/10.1111/1365-2664.13020

de Kroon, H., Plaisier, A., van Goenendael, J., & Caswell, H. (1986). Elasticity: The relative contribution of demographic parameters to population growth rate. *Ecology*, *67*(5), 1427–1431.

Easterling, M. R., Ellner, S. P., & Dixon, P. M. (2000). Size specific sensitivity: Applying a new structured population model. *Ecology*, *81*(3), 694–708.

Ellner, S. P., Childs, D. Z., & Rees, M. (2016). *Data-driven modelling of structured populations: A practical guide to the integral projection model*. Springer International Publishing AG.

Ellner, S. P., & Rees, M. (2006). Integral projection models for species with complex demography. *The American Naturalist*, *167*(3), 410–428.

Erickson, R. A., Eager, E. A., Brey, M. B., Hansen, M. J., & Kocovsky, P. M. (2017). An integral projection model with YY-males and application to evaluating grass carp control. *Ecological Modelling*, *361*, 14–25. https://doi.org/10.1016/j.ecolmodel.2017.07.030

Ferrer-Cervantes, M. E., Mendez-Gonzalez, M. E., Quintana-Ascencio, P.-F., Dorantes, A., Dzib, G., & Duran, R. (2012). Population dynamics of the cactus *Mammillaria gaumeri*: An integral projection model approach. *Population Ecology*, *54*, 321–334. https://doi.org/10.1007/s10144-012-0308-7

Henry, L., & Wickham, H. (2020). *rlang: Functions for base types and core R and 'Tidyverse' features*. R package version 0.4.7. https://CRAN.R-project.org/package=rlang

Jongejans, E., Shea, K., Skarpaas, O., Kelly, D., & Ellner, S. P. (2011). Importance of individual and environmental variation for invasive species spread: A spatial integral projection model. *Ecology*, *92*(1), 86–97. https://doi.org/10.1890/09-2226.1

Levin, S. C. (2021). Data from: Levisc8/ipmr: (Version v0.0.3). *Zenodo*, https://doi.org/10.5281/zenodo.5095062

Levin, S. C., Compagnoni, A. C., Childs, D. Z., Evers, S., Salguero-Gomez, R., & Knight, T. M. (2021). *ipmr: Fits Integral projection models using an expression based framework*. R package version 0.0.2. https://CRAN.R-project.org/package=ipmr

Merow, C., Dahlgren, J. P., Metcalf, C. J. E., Childs, D. Z., Evans, M. E. K., Jongejans, E., Record, S., Rees, M., Salguero-Gomez, R., & McMahon, S. M. (2014). Advancing population ecology with integral projection models: A practical guide. *Methods in Ecology and Evolution*, *5*, 99–110. https://doi.org/10.1111/2041-210X.12146S

Metcalf, C. J. E., Ellner, S. P., Childs, D. Z., Salguero-Gómez, R., Merow, C., McMahon, S. M., Jongejans, E., & Rees, M. (2015). Statistical modelling of annual variation for inference on stochastic population dynamics using Integral Projection Models. *Methods in Ecology and Evolution*, *6*(9), 1007–1017. https://doi.org/10.1111/2041-210X.12405

Metcalf, C. J. E., McMahon, S. M., Salguero-Gómez, R., & Jongejans, E. (2013). IPMpack: An R package for integral projection models. *Methods in Ecology and Evolution.*, *4*(2), 195–200. https://doi.org/10.1111/2041-210x.12001

Salguero-Gómez, R., Jones, O. R., Archer, C. R., Bein, C., de Buhr, H., Farack, C., Gottschalk, F., Hartmann, A., Henning, A., Hoppe, G., Roemer, G., Ruoff, T., Sommer, V., Wille, J., Voigt, J., Zeh, S., Vieregg, D., Buckley,

Y. M., Che-Castaldo, J., … Vaupel, J. W. (2016). COMADRE: A global database of animal demography. *Journal of Animal Ecology*, *85*, 371–384. https://doi.org/10.1111/1365-2656.12482

Rees, M., & Ellner, S. P. (2009). Integral projection models for populations in temporally varying environments. *Ecological Monographs*, *79*(4), 575–594. https://doi.org/10.1890/08-1474.1.

Salguero-Gómez, R., Jones, O. R., Archer, C. A., Buckley, Y. M., Che-Castaldo, J., Caswell, C., Hodgson, D., Scheuerlein, A., Conde, D. A., Brinks, E., de Buhr, H., Farack, C., Gottschalk, F., Hartmann, A., Henning, A., Hoppe, G., Roemer, G., Runge, J., Ruoff, T., … Vaupel, J. W. (2014). The COMPADRE Plant Matrix Database: An online repository for plant population dynamics. *Journal of Ecology*, *103*, 202–218. https://doi.org/10.1111/1365-2745.12334

Shefferson, R. P., Kurokawa, S., & Ehrlen, J. (2020). LEFKO3: Analysing individual history through size-classified matrix population models. *Methods in Ecology and Evolution*. https://doi.org/10.1111/2041-210X.13526

Williams, J. L., Miller, T. E. X., & Ellner, S. P. (2012). Avoiding unintentional eviction from integral projection models. *Ecology*, *93*(9), 2008–2014. https://doi.org/10.1890/11-2147.1

## SUPPORTING INFORMATION

Additional supporting information may be found online in the Supporting Information section.

**How to cite this article:** Levin, S. C., Childs, D. Z., Compagnoni, A., Evers, S., Knight, T. M., & Salguero-Gómez, R. (2021). ipmr: Flexible implementation of Integral Projection Models in R. *Methods in Ecology and Evolution*, 00, 1–9. https://doi.org/10.1111/2041-210X.13683

## Chapter 3: Rpadrino: an R package to access and use PADRINO, an open access database of Integral Projection Models

Authors:

Sam C. Levin

Sanne Evers

Tomos Potter

Mayra Pena-Guerrero

Dylan Z. Childs

Aldo Compagnoni

Tiffany M. Knight

Roberto Salguero-Gomez

*Published in* Methods in Ecology and Evolution 13(9): 1923-1929.

https://doi.org/10.1111/2041-210X.13910

Received: 8 March 2022 | Accepted: 20 May 2022

DOI: 10.1111/2041-210X.13910

Methods in Ecology and Evolution BRITISH ECOLOGICAL SOCIETY

APPLICATION

# Rpadrino: An R package to access and use PADRINO, an open access database of Integral Projection Models

Sam C. Levin[1,2,3] | Sanne Evers[1,2] | Tomos Potter[3,4] | Mayra Peña Guerrero[1,2] | Dylan Z. Childs[5] | Aldo Compagnoni[1,2,3] | Tiffany M. Knight[1,2,6] | Roberto Salguero-Gómez[3]

[1]Institute of Biology, Martin Luther University Halle-Wittenberg, Halle (Saale), Germany; [2]German Centre for Integrative Biodiversity Research (iDiv) Halle-Jena-Leipzig, Leipzig, Germany; [3]Department of Zoology, University of Oxford, Oxford, UK; [4]Department of Biological Sciences, Florida State University, Tallahassee, FL, USA; [5]Department of Animal and Plant Sciences, University of Sheffield, Sheffield, UK and [6]Department of Community Ecology, Helmholtz Centre for Environmental Research-UFZ, Halle (Saale), Germany

**Correspondence**
Sam C. Levin
Email: levisc8@gmail.com

**Funding information**
Alexander von Humboldt-Stiftung; Natural Environment Research Council, Grant/Award Number: NE/M018458/1

**Handling Editor:** Giovanni Strona

## Abstract

1. Discrete time structured population projection models are an important tool for studying population dynamics. Within this field, integral projection models (IPMs) have become a popular method for studying populations structured by continuously distributed traits (e.g. height, weight). Databases of discrete time, discrete state structured population models, for example DATLife (life tables) and COMPADRE & COMADRE (matrix population models), have made quantitative syntheses straightforward to implement. These efforts allow researchers to address questions in both basic and applied ecology and evolutionary biology. Since their introduction in 2000, over 300 works containing IPMs have been published, offering opportunities for ecological synthesis too. We describe a novel framework to quickly reconstruct these models for subsequent analyses using Rpadrino R package, which serves as an interface to PADRINO, a new database of IPMs.
2. We introduce an R package, Rpadrino, which enables users to download, subset, reconstruct, and extend published IPMs. Rpadrino makes use of recently created software, ipmr, to provide an engine to reconstruct a wide array of IPMs from their symbolic representations and conduct subsequent analyses. Rpadrino and ipmr are extensively documented to help users learn their usage.
3. Rpadrino currently enables users to reconstruct 280 IPMs from 40 publications that describe the demography of 14 animal and 26 plant species. All of these IPMs are tested to ensure they reproduce published estimates. Rpadrino provides an interface to augment PADRINO with external data and modify

Tiffany M. Knight and Roberto Salguero-Gómez—Joint senior authors.

parameter values, creating a platform to extend models beyond their original purpose while retaining full reproducibility.
4. PADRINO and Rpadrino provide a toolbox for asking new questions and conducting syntheses with peer-reviewed published IPMs. Rpadrino provides a user-friendly interface so researchers do not need to worry about the database structure or syntax, and can focus on their research questions and analyses. Additionally, Rpadrino is thoroughly documented and provides numerous examples of how to perform analyses which are not included in the package's functionality.

**KEYWORDS**

database, demography, elasticity, life history, open access, population dynamics, sensitivity

## 1 | INTRODUCTION

Demography provides an excellent approach to examine the ecology (Crone et al., 2011), evolutionary biology (Metcalf & Pavard, 2007), and conservation biology of any species (Doak & Morris, 2002). Environmental conditions and biotic interactions influence vital rates (e.g. survival, development, and reproduction) across the entire life cycle, which then govern its short-term and long-term performance (Caswell, 2001). A variety of methods exist for combining vital rates into demographic models; discrete-time, structured population models are among the most popular (Caswell, 2001; Crone et al., 2011). Indeed, there is a rich history of using such structured population models across a variety of sub-disciplines in ecology (e.g. Adler et al., 2010; Caswell, 2001; Easterling et al., 2000; Ellner et al., 2016).

In ecology, matrix projection models (MPMs) are the most widely used structured population model. MPMs divide the population into discrete classes corresponding to some trait value (e.g. developmental state, age, or size), and then model the population using vital rates computed for each class. Researchers have also recognized that, for some species, vital rates are best predicted as a function of one or more continuous traits (e.g. size, height, mass), rather than as a function of discrete classes (Easterling et al., 2000). Integral projection models (IPMs), which are continuously structured population models, have become an increasingly important tool for ecologists interested in addressing broad biological questions through a demographic lens (Gonzalez et al., 2021). IPMs combine vital rate functions of continuous traits into projection kernels, which describe how the abundance and distribution of trait values in a population change in discrete time (Easterling et al., 2000). IPMs have been used to investigate a variety of topics, such as invasive species spread (e.g. Erickson et al., 2017; Jongejans et al., 2011), evolutionary stable strategies (e.g. Childs et al., 2004), the effect of climate drivers on population persistence (Compagnoni, Pardini, & Knight, 2021; Salguero-Gómez et al., 2012), and linking evolutionary feedbacks to population dynamics (Coulson et al., 2011).

In order to reconstruct and use an IPM, researchers need, at a minimum, the symbolic representation of the model and the associated parameter values. Existing demographic databases enter transition values directly, rather than a symbolic version of the model and the values associated with the symbols separately. For example, COMPADRE and COMADRE store transition matrices as numeric matrices (sub-matrices corresponding to survival and development (*U*), sexual reproduction (*F*), asexual reproduction (*C*), and their sum (*A*), rather than symbolic matrices with parameter values separately. In general, this data format limits the variety of potential analyses, because individual matrix elements may be composed of multiple vital rates and this information is lost by storing only the resulting values (i.e. the elements of *F* may be comprised of both probability of reproducing and the per-capita number of propagules produced). To avoid this issue for IPMs, one needs to reconstruct the IPM using the functional form of the kernels and vital rates, as well as the associated parameter estimates. One can use tools that associate the symbols with their values to accomplish this task (e.g. metaprogramming and *rlang*, Henry & Wickham, 2021). *ipmr* is an *R* package for users to interactively develop their own IPMs from symbolic model representations and parameter estimates, and perform downstream analyses (Levin et al., 2021). *Rpadrino* extends this framework to include *reconstructing* previously published IPMs that are stored in the PADRINO database.

Here, we introduce *Rpadrino*. *Rpadrino* provides access to PADRINO, an open access database of IPMs. Specifically, PADRINO houses symbolic representations of IPMs, their parameter values, and associated metadata to aid users in selecting appropriate models. *Rpadrino* is an R package that enable users to download PADRINO, manage the dataset locally, modify, reconstruct, and analyse IPMs from PADRINO. In the following, we describe how to interact with PADRINO using *Rpadrino* and discuss future directions for *Rpadrino* and PADRINO. We also provide two case studies that demonstrate (a) how to use PADRINO and *Rpadrino* to reconstruct published IPMs, conduct perturbation analyses, compute some life cycle events, and troubleshoot problems, and (b) how to use *Rpadrino* and *ipmr* to combine PADRINO IPMs with user-specified

IPMs, and then how to use PADRINO data with other databases, using BIEN (Maitner et al., 2017) and COMPADRE (Salguero-Gómez et al., 2014) as examples. The latter is intended to demonstrate the potential for *Rpadrino* in broad, interoperable, macro-ecological applications. Finally, our supplementary materials also contain a detailed overview of the PADRINO database, along with the associated assumptions and challenges.

## 2 | AN INTRODUCTION TO IPMs AND PADRINO

First, we provide a brief review of how IPMs are structured. The simplest form of the IPM can be expressed as

$$n(z',t+1)=\int_L^U \left[P(z',z)+F(z',z)+C(z',z)\right]n(z,t)\mathrm{d}z, \quad (1)$$

where $n(z',t+1)$ and $n(z,t)$ are the distributions of trait values ($z$) of individuals in the population at time $t+1$ and $t$, $P(z',z)$ is a kernel describing the survival and development of existing individuals, $F(z',z)$ is a kernel describing per-capita sexual reproduction and $C(z',z)$ is a kernel describing per-capita asexual reproduction (i.e. clonal reproduction). Each kernel may be comprised of any number of vital rate functions (Ellner et al., 2016). Analytical solutions to the integrals in Equation (1) are not available (Ellner & Rees, 2006). Therefore, the integrals are numerically approximated, resulting in a large iteration matrix (typically ranging from $45 \times 45$ to $1{,}000 \times 1{,}000$ in dimension, based on data from PADRINO), and then some quantities of interest are computed (Ellner et al., 2016).

Before introducing *Rpadrino*, we provide a brief overview of PADRINO. PADRINO is an open-access database of integral projection models. PADRINO defines a syntax to symbolically represent IPMs as text strings, and stores the values of those symbols in separate tables. The syntax used is very similar to the mathematical notation of IPMs and is largely 'language-agnostic' (i.e. aims to avoid idiosyncrasies of specific programming languages). For example, a survival/growth kernel with the form $P(z',z)=s(z)*G(z',z)$ would be `P = s * G` in PADRINO's syntax. $G(z',z)=f_G(z'|\mu_g(z),\sigma_G)$ (where $f_G$ denotes a normal probability density function) becomes `G = Norm(mu_g, sd_g)`. This notation should be translatable to many computing languages beyond just *R* (e.g. Python or Julia). Additionally, PADRINO stores extensive metadata to help researchers find IPMs that work for their questions. A more complete description of the database, how IPMs are digitized, and the associated challenges is available in the ESM and the project webpage (https://padrinodb.github.io/Padrino/, Table 1, Appendix, Tables S1 and S2).

**TABLE 1** Taxonomic representation of IPMs accessible via Rpadrino. These numbers represent the number of models that are error checked and accurately reproduce the published IPM (see 'Data Validation' in the Appendix for more details). Models that are partially entered or still contain errors are not considered here. We are in the process of correcting them and/or retrieving additional information from the authors. See Appendix for details.

| Kingdom | # of unique ipm_ids | # of unique species | # of publications |
|---|---|---|---|
| Totals | 280 | 56 | 40 |
| Animalia | 22 | 16 | 14 |
| Plantae | 258 | 40 | 26 |

## 3 | *RPADRINO* AND *IPMR*

*Rpadrino* is an *R* package that contains functions for downloading the PADRINO database, data querying and management, modifying existing functional forms and parameter values, and reconstructing models. Model reconstruction is powered by the *ipmr R* package (Levin et al., 2021). While users do not need to know how to use *ipmr* to use *Rpadrino*, the two packages are designed to work with and enhance each other. This means that users can combine IPMs reconstructed with *Rpadrino* with IPMs of their own constructed with *ipmr* in a single, coherent analysis (case study 2). Furthermore, users can go from downloading the database to reconstructing IPM objects in as little as 3 function calls. A more in depth workflow is provided below.

The flexibility of IPMs and their broad application across ecology, evolution, and conservation biology mean that there is no fixed set of steps in a workflow using *Rpadrino*. However, there are generally four steps that a researcher must take when using *Rpadrino*. The first step is to identify studies of interest (Figure 1, Step 1a), and, optionally, augment PADRINO's metadata with additional information from other sources (e.g. environmental data, GBIF, Figure 1, Step 1). *Rpadrino* represents PADRINO objects as a list of data.frames (referred to as tables in subsequent text). *Rpadrino* uses the shared ipm_id column across all tables to track information related to each IPM. Therefore, subsetting relies on identifying the correct ipm_ids, and then using those to select the IPMs of interest (Box 1, case study 1 and 2). data.frames should be familiar to most *R* users, and the ability to modify them should readily accommodate the range of further analyses that researchers may be interested in. Users may augment any table with additional information corresponding to, for example, spatial or temporal covariates from other open access databases. Furthermore, *Rpadrino* provides numerous access functions for metadata that streamline subsetting (Box 1).

The second step in the *Rpadrino* workflow is to construct a list of `proto_ipm` objects using `pdb_make_proto_ipm()` (Figure 1, Box 1). This function translates PADRINO's syntax into *ipmr* code, and then builds a `proto_ipm` object for each unique `ipm_id`. For some models, users may choose to create deterministic or stochastic IPMs at this step. *Rpadrino*'s default behaviour is to generate deterministic models whenever possible. This behaviour encompasses instances where authors generated models with no time or space

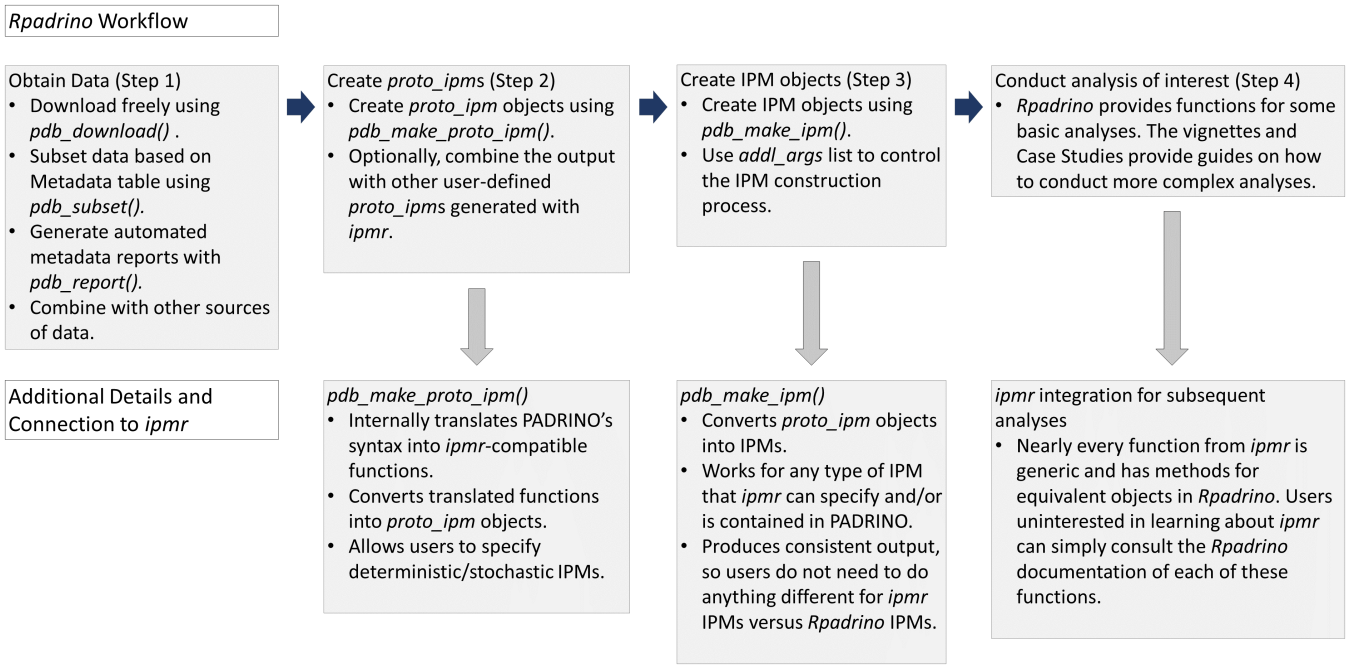



**FIGURE 1** An overview of general workflows with Rpadrino. The upper row of boxes displays the workflow, while the lower row provides additional details on each step and how ipmr fits in to them. The first step is to download and subset PADRINO based on the research question at hand. At this point, if the research question calls for it, users may augment the PADRINO object with data from other sources. The extensive metadata table in PADRINO is designed to make this as straightforward as possible. Once the dataset is prepared, users can also generate automated reports on their data subset (1). After preparing the dataset, users create proto_ipm objects using a single function: pdb_make_proto_ipm(). This function translates PADRINO's syntax into ipmr code and then uses ipmr to generate the proto_ipm objects. All of these steps take place internally, and so users do not need to understand PADRINO's syntax or how ipmr works to move forward (2). Once proto_ipms are created, users generate IPM objects with a single function, pdb_make_ipm(). Examples of how to modify default building settings are in each case study in the supplementary materials and in the function's documentation (3). Finally, Rpadrino provides some basic analytical machinery, including deterministic and stochastic growth rates and eigenvectors, the ability to make iteration kernels from sub-kernels, and mean kernels (4). Examples of how to conduct more complex analyses are included in the package vignettes and the case studies in the supplementary materials

varying parameters, and where authors included discretely varying environments. The latter can be implemented as deterministic models because all parameter values are known before the IPM is built. IPMs with continuous environmental variation require sampling the environment at each model iteration, usually by sampling from distributions randomly. These are always considered stochastic models. This is also the step where, if needed, users should combine their own `proto_ipm`'s produced by *ipmr* with the `proto_ipm`'s produced by *Rpadrino*.

The third step in the *Rpadrino* workflow is creating IPM objects with `pdb_make_ipm()` (Figure 1, Box 1). `pdb_make_ipm()` uses *ipmr*'s `make_ipm()` function to build IPM objects. Users may specify additional options to pass to `make_ipm()` (e.g. normalize the population size to always equal 1, return the vital rate function values as well as the sub-kernels and population state). The various arguments users can modify are described in the *ipmr* documentation for `make_ipm()`.

The fourth and final step in an *Rpadrino* workflow is to conduct the analyses of interest (Figure 1, Box 1). *Rpadrino* provides functions to extract per-capita growth rates, eigenvectors (Caswell, 2001; Ellner et al., 2016, Ch. 2, demonstrated in Box 1), assess convergence to asymptotic dynamics (Caswell, 2001), compute mean kernels for stochastic IPMs (Ellner et al., 2016, Ch 7), and modify existing IPMs with new parameter values and functional forms. Additionally, the documentation on the *Rpadrino* website (https://padrinODB.github.io/Rpadrino/index.html) and the Supplementary Materials for this paper contain details on how to conduct more complicated analyses with IPM objects (e.g. perturbation analyses (Ellner et al., 2016, Ch 4), size at death calculations (Metcalf et al., 2009)). The package documentation and the recent publication describing *ipmr* also contain code demonstrating analyses on single IPM objects (Levin et al., 2021). These can be extended via the `apply` family of functions.

```
# Install and load the CRAN version:
# install.packages("Rpadrino")
library(Rpadrino)
# Step 1 from main text -----
# pdb_download() downloads a copy PADRINO. We can specify a path to save the
# downloaded database using `save = TRUE` and
# `destination = 'path/to/file/'`. We'll call the object we create 'pdb',
# which is short for Padrino DataBase.
pdb <- pdb_download(save = FALSE)
```

**BOX 1** ***An example of a simple analysis workflow using*** **Rpadrino**

*The first step in using* Rpadrino *is to install and load the package. After that, we can use* Rpadrino *to download* PADRINO *and, optionally, save it locally on our computer. Once the data are downloaded, we can make use of* Rpadrino*'s metadata accessor functions to quickly select models that meet our criteria (step 1). The concept of the* `ipm_id` *is explained in greater detail in the Appendix of this manuscript. The next step is to use these* `ipm_id`*s to create a list of* `proto_ipm`*'s using* `pdb_make_proto_ipm()` *(step 2). After this step, we can create actual IPM objects using* `pdb_make_ipm()` *(step 3). Once IPM objects are created, the following steps are according to the demands of the research question. In this case, asymptotic population growth rates, stable size distributions, and reproductive values are extracted (step 4). Note that since the* Geum radiatum *model includes a number of year-specific estimates, multiple values are generated for each quantity we want to extract. The concise representation and reconstruction of models such as this is powered by* ipmr*'s parameter set index notation, which is described in greater detail on the package website (https://levisc8.github.io/ipmr/articles/index-notation.html). However, users do not need to be familiar with this notation unless they wish to modify the IPM in question (see case study 1 for an example of modifying* PADRINO *IPMs with* Rpadrino*).*

```
# We can use Rpadrino's metadata accessors to get a selection of ipm_ids
# that we want to use. For this example, we'll select models for Carpobrotus
# species and Geum radiatum. First, extract the 'species_accepted' column.
# The output of this will be named, and the names are the ipm_id associated
# with each piece of metadata. Thus, we can subset the names of the 'spps'
# object to get the ipm_ids we need.
spps <- pdb_species_accepted(pdb)
ids <- names(spps)[spps %in% c("Carpobrotus_spp", "Geum_radiatum")]
# Step 2 from main text -----
# Next, we create a list of proto_ipm's using pdb_make_proto_ipm().
my_proto_ipms <- pdb_make_proto_ipm(pdb, ipm_id = ids)
# Step 3 from main text -----
# After creating the proto_ipm list, we can call pdb_make_ipm() to construct
# actual IPM objects.
my_ipms <- pdb_make_ipm(my_proto_ipms)
# Step 4 from main text -----
# After re-building our published IPMs, the next step is to analyse them.
# In this case, we'll just extract the asymptotic population growth rates,
# stable size distribution, and the reproductive values. Note that for the
# Geum IPMs, there are multiple year-specific values that are returned.
# All values related to population-level traits are computed via iteration,
# as this approach handles more complicated IPM systems more efficiently
# than eigenvector/eigenvalue based approaches for larger IPMs, and
# introduces little to no additional computation time for simpler and/or
# smaller IPMs.
lambdas <- lambda(my_ipms)
ssds <- right_ev(my_ipms, iterations = 150, tolerance = 1e-7)
repro_vs <- left_ev(my_ipms, iterations = 150, tolerance = 1e-7).
```

## 4 | CHALLENGES

There are numerous challenges associated with reproducing published IPMs. Challenges related to digitizing and storing IPMs are discussed in the ESM. Important challenges remain in the reconstruction of IPMs. Semi- or non-parametric models may be used to generate IPMs whose functional form is not known a priori. We have not yet developed a general syntax for representing these models in PADRINO, though work is ongoing. Additionally, *ipmr* is not yet able to handle two-sex models (e.g. Stubbered et al., 2019), time-lagged models (e.g. Kuss et al., 2008), or periodic models (e.g. Letcher et al., 2014). These types of IPMs do not yet represent a substantial portion of the literature. Nonetheless, it is our intention to continue developing functionality to accommodate them in future releases of *Rpadrino*, *ipmr* and PADRINO.

## 5 | OPPORTUNITIES AND FUTURE DIRECTIONS

*Rpadrino* presents unique opportunities for synthesis in both theoretical and applied contexts. The expanded range of phylogenetic and geographical coverage can be used in conjunction with other demographic databases (e.g. COM(P)ADRE (Salguero-Gómez et al., 2014; Salguero-Gómez et al., 2016), popler (Compagnoni et al.,

2019), DatLife (DatLife, 2021)) to power larger scale syntheses than were possible before (e.g. Compagnoni, Levin, et al., 2021). For example, one could use IPMs from PADRINO and matrix population models from COMPADRE and COMADRE to create life tables (Jones et al., 2021), which could then be combined with life tables from DATLife for further analysis (e.g. Jones et al., 2014). The intermediate life table conversion steps may not be necessary, as many of the same life history traits and population level parameters may be calculated from all of these models (Caswell, 2001; Ellner et al., 2016). Furthermore, recent publications combine biotic and abiotic interactions into demographic models providing a robust theoretical toolbox for exploring species responses to environmental drivers such as climate change (e.g. Abrego et al., 2021; Simmonds et al., 2020). *Rpadrino* also provides functionality to modify parameter values and functional forms of the IPMs it stores, giving theoreticians a wide array of realistic life histories to experiment with. These features will enable researchers to carry out more detailed and comprehensive analyses at various spatial, temporal, and phylogenetic scales. The examples given here are far from an exhaustive list, but hopefully demonstrates the potential for this new tool in demography, ecology and evolutionary biology (Table 1).

## AUTHORS' CONTRIBUTIONS

S.C.L. designed *ipmr* and *Rpadrino* with contributions from all authors, and S.C.L. implemented the packages; S.E., T.P., M.P.G., and S.C.L. entered the data into PADRINO; S.C.L. wrote the first draft of the manuscript and all authors provided comments.

## ACKNOWLEDGEMENTS

We thank L Sfedu for help with designing the figures, and the associate editor and two anonymous reviewers for comments that greatly improved this manuscript.

## FUNDING INFORMATION

R.S.-G. was supported by a NERC Independent Research Fellowship (NE/M018458/1). S.C.L., A.C., S.E. and T.M.K. were funded by the Alexander von Humboldt Foundation in the framework of the Alexander von Humboldt Professorship of T.M.K.

## CONFLICT OF INTEREST

The authors declare no conflict of interest.

## PEER REVIEW

The peer review history for this article is available at https://publons.com/publon/10.1111/2041-210X.13910.

## DATA AVAILABILITY STATEMENT

PADRINO (Levin et al., 2022a, 2022b) is available via the *Rpadrino* R package, as well as on Github (https://github.com/padrinoDB/Padrino) and Zenodo (https://zenodo.org/badge/latestdoi/109448718). *Rpadrino* (Levin et al., 2022a, 2022b) is available on CRAN (https://cran.r-project.org/package=Rpadrino), Github (https://github.com/padrinoDB/Rpadrino), and Zenodo (https://zenodo.org/badge/latestdoi/124245125). *pdbDigitUtils* (Levin et al., 2022a, 2022b) is available on Github (https://github.com/padrinoDB/pdbDigitUtils) and Zenodo (https://zenodo.org/badge/latestdoi/348737812). There is no other data associated with this paper.

## ORCID

*Sam C. Levin* https://orcid.org/0000-0002-3289-9925
*Sanne Evers* https://orcid.org/0000-0002-8002-1658
*Roberto Salguero-Gómez* https://orcid.org/0000-0002-6085-4433

## REFERENCES

Abrego, N., Roslin, T., Huotari, T., Ji, Y., Schmidt, N. M., Wang, J., Yu, D. W., & Ovaskainen, O. (2021). Accounting for species interactions is necessary for predicting how arctic arthropod communities respond to climate change. *Ecography*, *44*, 885–896. https://doi.org/10.1111/ecog.05547

Adler, P. B., Ellner, S. P., & Levine, J. M. (2010). Coexistance of perennial plants: An embarassment of niches. *Ecology Letters*, *13*, 1019–1029. https://doi.org/10.1111/j.1461-0248.2010.01496.x

Caswell, H. (2001). *Matrix population models: Construction, analysis, and interpretation* (2nd ed.). Sinauer Associates Inc.

Childs, D. Z., Rees, M., Rose, K. E., Grubb, P. J., & Ellner, S. P. (2004). Evolution of size-dependent flowering in a variable environment: construction and analysis of a stochastic integral projection model. *Proceedings of the Royal Society B: Biological Sciences*, *271*(1547), 425–434. https://doi.org/10.1098/rpsb.2003.2597

Clark, J. S. (2003). Uncertainty in population growth rates calculated from demography: The hierarchical approach. *Ecology*, *84*, 1370–1381.

Compagnoni, A., Bibian, A. J., Ochocki, B. M., Levin, S. C., Zhu, K., & Miller, T. E. X. (2019). popler: An R package for extraction and synthesis of population time series from the long-term ecological research (LTER) network. *Methods in Ecology and Evolution*, *11*(2), 258–264.

Compagnoni, A., Levin, S. C., Childs, D. Z., Harpole, S., Paniw, M., Roemer, G., Burns, J. H., Che-Castaldo, J., Rueger, N., Kunstler, G., Bennett, J. M., Archer, C. R., Jones, O. R., Salguero-Gómez, R., & Knight, T. M. (2021). Herbaceous perennial plants with short generation time have stronger responses to climate anomalies than those with longer generation time. *Nature Communications*, *12*, 1824. https://doi.org/10.1038/s41467-021-21977-9

Compagnoni, A., Pardini, E., & Knight, T. M. (2021). Increasing temperature threatens an already endangered coastal plant species. *Ecosphere*, *12*(3), e03454. https://doi.org/10.1002/ecs2.3454

Coulson, T., MacNulty, D. R., Stahler, D. R., von Holdt, B., Wayne, R. K., & Smith, D. W. (2011). Modeling effects of environmental change on wolf population dynamics, trait evolution, and life history. *Science*, *334*(6060), 1275–1278. https://doi.org/10.1126/science.1209441

Crone, E. E., Menges, E. S., Ellis, M. M., Bell, T., Bierzychudek, P., Ehrlen, J., Kaye, T. N., Knight, T. M., Lesica, P., Morris, W. F., Oostermeijer, G., Quintana-Ascencio, P. F., Stanley, A., Ticktin, T., Valverde, T., & Williams, J. L. (2011). How do ecologists use matrix population models? *Ecology Letters*, *14*(1), 1–8. https://doi.org/10.1111/j.1461-0248.2010.01540.x

DATLife – The Demography Across the Tree of Life – database. (2021). Max-Planck Institute for Demographic Research (Germany). Retrieved from www.datlife.org

Doak, D., & Morris, W. F. (2002). *Quantitative conservation biology: Theory and practice of population viability analysis*. Oxford University Press.

Easterling, M. R., Ellner, S. P., & Dixon, P. M. (2000). Size specific sensitivity: Applying a new structured population model. *Ecology*, *81*(3), 694–708.

Ellner, S. P., Childs, D. Z., & Rees, M. (2016). *Data-driven modelling of structured populations: A practical guide to the integral projection model*. Springer International Publishing AG.

Ellner, S. P., & Rees, M. (2006). Integral Projection Models for species with complex demography. *The American Naturalist*, *167*(3), 410–428.

Erickson, R. A., Eager, E. A., Brey, M. B., Hansen, M. J., & Kocovsky, P. M. (2017). An integral projection model with YY-males and application to evaluating grass carp control. *Ecological Modelling*, *361*, 14–25. https://doi.org/10.1016/j.ecolmodel.2017.07.030

Gonzalez, E., Childs, D. Z., Quintana-Ascenscio, P. F., & Salguero-Gomez, R. (2021). Integral projection models. In R. Salguero-Gomez, & M. Gamelon (Eds.), *Demographic methods across the tree of life*. Oxford University Press.

Henry, L., & Wickham, H. (2021). rlang: Functions for base types and core R and 'Tidyverse' features. R package version 0.4.11. Retrieved from https://CRAN.R-project.org/package=rlang

Jones, O. R., Barks, P., Stott, I., James, T. D., Levin, S. C., Petry, W. K., Capdevila, P., Che-Castaldo, J., Jackson, J., Römer, G., Schuette, C., Thomas, C. C., & Salguero-Gómez, R. (2021). Rcompadre and Rage—Two R packages to facilitate the use of the COMPADRE and COMADRE databases and calculation of life history traits from matrix population models. *Methods in Ecology and Evolution*, *13*(4), 770–781. https://doi.org/10.1111/2041-210X.13792

Jones, O. R., Scheuerlein, A., Salguero-Gómez, R., Camarda, C. G., Schaible, R., Casper, B. B., Dahlgren, J. P., Ehrlén, J., García, M. B., Menges, E. S., Quintana-Ascencio, P. F., Caswell, H., Baudisch, A., & Vaupel, J. W. (2014). The diversity of ageing across the tree of life. *Nature*, *505*, 169–173. https://doi.org/10.1038/nature12789

Jongejans, E., Shea, K., Skarpaas, O., Kelly, D., & Ellner, S. P. (2011). Importance of individual and environmental variation for invasive species spread: A spatial integral projection model. *Ecology*, *92*(1), 86–97. https://doi.org/10.1890/09-2226.1

Kuss, P., Rees, M., Aegisdottir, H. H., Ellner, S. P., & Stoecklin, J. (2008). Evolutionary demography of long-lived monocarpic perennials: A time-lagged integral projection model. *Journal of Ecology*, *96*, 821–832. https://doi.org/10.1111/j.1365-2745.2008.01374.x

Letcher, B. H., Schueller, P., Bassar, R. D., Nislow, K. H., Coombs, J. A., Sakrejda, K., Morrissey, M., Sigourney, D. B., Whiteley, A. R., O'Donnell, M. J., & Dubreuil, T. L. (2014). Robust estimates of environmental effects on population vital rates: An integrated capture-recapture model of seasonal brook trout growth, survival and movement in a stream network. *Journal of Animal Ecology*, *84*(2), 337–352. https://doi.org/10.1111/1365-2656.12308

Levin, S. C., Childs, D. Z., Compagnoni, A., Evers, S., Knight, T. M., & Salguero-Gómez, R. (2021). ipmr: Flexible implementation of Integral Projections Models in R. *Methods in Ecology and Evolution*, *12*, 1826–1834. https://doi.org/10.1111/2041-210X.13683

Levin, S. C., Evers, S., Potter, T., Guerrero, M. P., Childs, D. Z., Compagnoni, A., Knight, T. M., & Salguero-Gómez, R. (2022a). PADRINO v0.0.1. https://doi.org/10.5281/zenodo.6573870

Levin, S. C., Evers, S., Potter, T., Guerrero, M. P., Childs, D. Z., Compagnoni, A., Knight, T. M., & Salguero-Gómez, R. (2022b). Rpadrino v0.0.4.1: Methods in ecology and evolution release. https://doi.org/10.5281/zenodo.6573880

Maitner, B., Boyle, B., Casler, N., Condit, R., Donoghue, J., Duran, S. M., Guaderrama, D., Hinchliff, C. E., Jørgensen, P. M., Kraft, N. J. B., McGill, B., Merow, C., Morueta-Holme, N., Peet, R. K., Sandel, B., Schildhauer, M., Smith, S. A., Svenning, J.-C., Thiers, B., … Enquist, B. J. (2017). The bien r package: A tool to access the Botanical Information and Ecology Network (BIEN) database. *Methods in Ecology and Evolution*, *9*(2), 373–379. https://doi.org/10.1111/2041-210X.12861

Metcalf, C. J. E., & Pavard, S. (2007). Why evolutionary biologists should be demographers. *Trends in Ecology and Evolution*, *22*(4), 205–212.

Metcalf, C. J. E., Horvitz, C. C., Tuljapurkar, S., & Clark, D. A. (2009). A time to grow and a time to die: A new way to analyze the dynamics of size, light, age and death of tropical trees. *Ecology*, *90*, 2766–2778.

Salguero-Gómez, R., & Gamelon, M. (2021). *Demographic methods across the tree of life*. Oxford University Press.

Salguero-Gómez, R., Jones, O. R., Archer, C. A., Buckley, Y. M., Che-Castaldo, J., Caswell, C., Hodgson, D., Scheuerlein, A., Conde, D. A., Brinks, E., de Buhr, H., Farack, C., Gottschalk, F., Hartmann, A., Henning, A., Hoppe, G., Roemer, G., Runge, J., Ruoff, T., … Vaupel, J. W. (2014). The COMPADRE Plant Matrix Database: An online repository for plant population dynamics. *Journal of Ecology*, *103*, 202–218. https://doi.org/10.1111/1365-2745.12334

Salguero-Gómez, R., Jones, O. R., Archer, C. R., Bein, C., de Buhr, H., Farack, C., Gottschalk, F., Hartmann, A., Henning, A., Hoppe, G., Roemer, G., Ruoff, T., Sommer, V., Wille, J., Voigt, J., Zeh, S., Vieregg, D., Buckley, Y. M., Che-Castaldo, J., … Vaupel, J. W. (2016). COMADRE: A global database of animal demography. *Journal of Animal Ecology*, *85*, 371–384. https://doi.org/10.1111/1365-2656.12482

Salguero-Gómez, R., Siewert, W., Casper, B. B., & Tielboerger, K. (2012). A demographic approach to study effects of climate change in desert plants. *Philosophical Transactions of the Royal Society B*, *367*, 3100–3114.

Simmonds, E. G., Cole, E. F., Sheldon, B. C., & Coulson, T. (2020). Phenological asynchrony: A ticking time-bomb for seemingly stable populations? *Ecology Letters*, *23*(12), 1766–1775. https://doi.org/10.1111/ele.13603

Stubbered, M. W., Vindenes, Y., Vollestad, L. A., Winfield, I. J., Stenseth, N. C., & Langangen, O. (2019). Effects of size- and sex-selective harvesting: an integral projection model approach. *Ecology and Evolution*, *9*, 12556–12570. https://doi.org/10.1002/ece3.5719

## SUPPORTING INFORMATION

Additional supporting information may be found in the online version of the article at the publisher's website.

**How to cite this article:** Levin, S. C., Evers, S., Potter, T., Guerrero, M. P., Childs, D. Z., Compagnoni, A., Knight, T. M., & Salguero-Gómez, R. (2022). Rpadrino: An R package to access and use PADRINO, an open access database of Integral Projection Models. *Methods in Ecology and Evolution*, *13*, 1923–1929. https://doi.org/10.1111/2041-210X.13910

# Chapter 4: Relationship between climate and fitness of a highly invasive succulent

Authors:

Sam C. Levin

Roberto Salguero-Gómez

Tiffany M. Knight

**Abstract**

Invasive species pose a substantial threat to biodiversity. As such, understanding the drivers that explain the success of their populations is key for prevention and management. Mediterranean dune ecosystems are especially threatened by biological invasions and climate change due to their widespread distribution. To understand the relationship between climate change in biological invasions, we need detailed information on how climate influences the fitness and performance of invasive species. Carpobrotus species are a widely distributed invasive species in dune ecosystems, making them ideal for studying how climate affects fitness. We developed a novel, drone-based method of collecting demographic data for 13 sites distributed across the native and invaded ranges, and used these data to parameterize Integral Projection Models to estimate individual fitness and population performance as a function of climate variables. Contrary to our initial expectations, a life table response experiment on these models indicates that climate is not a strong driver of fitness. We discuss the multiple local factors that may contribute to Carpobrotus' success. Our results suggest that climate change is not likely to increase the invasive potential of this genus around the globe, despite of the changing climatic conditions projected for the coming decades.

**Introduction**

Invasive species are ecological pests at local, regional, and global scales (Pimentel et al. 2005, Vila et al. 2011, Levin et al. 2020). There is an extensive body of work devoted to understanding how species become invasive (reviewed in Lowry et al. 2013 and Jeschke & Heger 2018). However, generalization across families, genera, and even species remains challenging as environmental contexts can vary greatly across the invaded range, and population level processes (e.g. founder effects, rapid evolution, or hybridization) can induce rapid change in the invading population (reviewed in Prentis *et al.* 2008). The environmental context of invasive species is receiving increasing attention, as there is growing recognition that climate change may favor invasive populations and produce an even greater threat to ecosystems (Beaury et al. 2020).

There is a growing body of literature using species distribution models to understand how climate influences species ranges and to predict how climate change will alter them. These studies find contrasting expectations for species. For example, Bradley and colleagues (2009) predicted that *Centauria solstitialis* and various *Tamarix spp* should undergo range expansion, while *Bromus tectorum* is expected to undergo range contraction under future climate conditions. Similarly, Merow & colleagues (2017) found that *Allaria petiolata* will likely experience range contraction, but *Berberis thumbergii* will likely undergo range expansion. Furthermore, Allen & Bradley (2016) found that, while the hundreds of invasive plants in the continental U.S. would persist under future climate, the invasive plant species richness would decline in large portions of the country. Despite the effort dedicated to studying this, there is no clear consensus on when climate change should cause range expansion or contraction. Furthermore, there are substantial gaps left in our knowledge of how the many facets of climate change (*e.g.* increased temperature, variance in precipitation, $CO_2$ enrichment) will affect various species and biomes (reviewed in Ziska 2022).

Species distribution models (SDMs) are often employed to understand species' current and future ranges under climate change (Guisan & Thullier 2005, Bradley et al. 2009, Allen & Bradley 2020). However, SDMs have also been criticized for their applications (Briscoe et al. 2019, Lee-Yaw et al. 2021), as they are often

unable to predict occurrences in environments that are not covered by the observed range of the species in question (Merow et al. 2017, reviewed in Liu et al. 2020 and in Lake et al. 2020). Classical SDMs make use of presence data (or presence-absence data in the best cases) for a given species, and do not necessarily include information on spatial variation in population performance. Furthermore, making predictions on invaded range occupancy and performance based solely on data from the native range is risky because of novel biotic interactions that may arise, and because this practice often involves extrapolating estimates beyond the observed range of environmental data (Broennimann & Guisan 2008). Finally, the assumption that species are at equilibrium with their environment and thus occupy all suitable areas is unlikely to be true when using data from the native range (Elith et al. 2010).

One approach to handling these issues is to incorporate demographic data from both the native and invaded range, and sample broadly across environmental gradients (Merow et al. 2017, Briscoe et al. 2019). This approach explicitly links vital rates (e.g. survival, reproduction, and dispersal) to environmental conditions, allowing for a mechanistic approach to making performance projections (Briscoe et al. 2019). A drawback of this method is that it entails sampling populations on multiple continents and combining data from multiple sources, which is laborious and expensive. However, new methods of rapidly collecting demographic data, such as unmanned aerial vehicles (UAVs, drones hereafter) can reduce those burdens. Indeed, a number of authors have already used this technology to map populations of invasive species and estimate their fitness (Innangi et al. 2023, Bogdan et al. 2020, reviewed in Cavendar-Bares et al. 2022).

Mediterranean biomes are threatened by both climate change and invasions (Klausmeyer & Shaw 2009). These biomes are home to vast array of endemic plant and animal species. The Mediterranean ecosystems are threatened by the *Carpobrotus* genus (Aizoaceae), particularly *C. edulis*, *C. acinaciformis*, *C. chilensis*, and their hybrids, which are native to South Africa but have been introduced on six continents. These *Carpobrotus* species are spreading, mat forming succulents that are exceptionally competitive for water and light (D'Antonio & Mahall 1991, Novoa et al. 2012, Campoy et al. 2018). There is a growing body of evidence indicating these species are allelopathic, with their dessicated litter acidifying the soil and preventing other species from germinating (Novoa et al. 2012, Novoa et al. 2014).

Here, we apply novel demographic sampling methods to the *Carpobrotus* genus across 13 sites spread throughout their native and invaded range. We use drones as a fast and cheap way of generating detailed maps of each population (Jackson et al. 2022). We repeat sampling of populations across two years to estimate vital rates as a function of climate variation. Mediterranean ecosystems can experience strong water limitation, and *Carpobrotus* is exceptionally competitive for water resources (D'Antonio & Mahall 1991, Novoa et al. 2012, Campoy et al. 2018). We expect that this combination of factors will lead to reduced competition at water limited sites, because *Carpobrotus* will exclude inferior competitors for this essential nutrient. Therefore, we expect that *Carpobrotus* populations will have higher per-capita growth rates in environments that receive less precipitation. We use a combination of Integral Projection Models (IPMs, Easterling et al. 2000, Ellner & Rees 2006), generalized additive models (GAMs), and a life table response experiment (LTRE, Caswell 1989) to test these hypotheses.

## Methods

The primary goal of this study was to understand how environmental variation affects *Carpobrotus* population dynamics. Therefore, we chose geographic areas to sample based on maximizing the amount of climate space that we would sample with our study. Within each geographic area, we chose sites based on the extent of the population (*i.e.* had enough ramets to perform statistical analysis on), and the ability to sample without disturbing local flora and fauna (i.e. was not home to threatened/endangered bird species, Table 4.1). Consequently, this study was conducted across 13 sites spread across South Africa (native range), New Zealand (invaded), Portugal (invaded), and Israel (invaded, Figure 4.1). Sites in the Northern Hemisphere were sampled in March/April of 2018-2020, and sites in the Southern Hemisphere were sampled in September/October of 2018-2019, so that plants were observed during their peak flowering period (*i.e.* early spring).

We used drones to collect aerial imagery each population. Briefly, we developed flight plans to map each

population using DJI Ground Station Pro v2 (henceforth called "GSP", SZ DJI Technology Co.) for iPad (Apple Inc). Transects were flown with a DJI Phantom 4 Pro v1 (SZ DJI Technology Co.) along the lines of the flight plan generated by DJI GSP and images were recorded at 1.8 - 2m intervals. When batteries reached a critical level of power (i.e. ~20%), we landed the drone, switched the batteries out, and the flight was resumed from the last stopping point. We generated flight plans by using the GSP app to draw polygons over an area of interest, and then let the app compute the optimal flight path given our desired output image resolution. We selected a flight path that generated a resolution of 0.2 - 0.45 cm/pixel. Flight altitude and degree of image overlap are key factors in determining the resolution of imagery and the ability to combine images to form a single map. We needed to generate resolutions of less than 2 cm/pixel in the resulting orthomosaic maps of each population (see below). Therefore, we used the app generate flight plants that had 70-85% image overlap on the front and side, and flew at altitudes between 10 and 15m above ground level. The average width of open flowers is 7.46 cm (+/- 0.008 S.E.) and unripe fruits is 2.32 cm (+/- 0.002 S.E.) across all populations (SC Levin, unpublished data), so this resolution is sufficient to allow us to mark the vast majority of unripe fruits and flowers in the resulting maps (orthomosaics, see below for more details). We deviated from these protocols when vegetation structure would not allow us to sample conventionally (e.g. there was a large tree in the middle of the site). In those cases, we flew the portions of the missions with obstacles manually, and deviated from flight plans to avoid collisions.

We mapped populations and extracted demographic data from these digitized maps. Once all images were captured after the first aerial survey, individual photos were processed into a single composite, georeferenced orthomosaic map using Pix4Dmapper (Pix4D SA, 2019). We drew individual polygons around contiguous Carpobrotus ramets that were visible on the orthomosaic, assigned each polygon a unique ID number, and counted flowers using a point layer in QGIS (QGIS Development Team 2021). We repeated the process in 2019 by overlaying the 2019 orthomosaic on the 2018 orthomosaic and marking polygons for all surviving individuals as well as new recruits also in QGIS. The surveys sometimes did not perfectly overlap with each other, so ramets from the first survey that were not flown over again were included in probability of flowering and flower production analyses, but not in survival and growth analyses (see below). Individual ramets that were within the range of space sampled in both time periods but not found again in our second aerial survey were scored as dead. Only one site had substantial disturbance in the year between samplings, likely due to a large storm off the coast of Christchurch and the resulting storm surge inundating the population for an extended period. This site was excluded.

In addition to mapping small, medium, and large ramets, we also needed to collect information on seedling demography to estimate the survival rate of seedlings and their size the following year, conditional on survival (see vital rate descriptions below for more information). In 2018, we delineated 2 $40cm \times 40cm$ plots and manually counted the number of seedlings. In 2019, we returned and flew brief surveys with the drone at ~ 1.2m above ground level. We generated orthomosaics using the same procedure described above, and then counted surviving seedlings and drew polygons around them to estimate their size. Unfortunately, we did not find seedlings in sufficient quantities at other sites to establish plots, and so our sample size is limited to these two plots at a single site for these parameters.

Our goal was to convert the data contained in orthomosaics into spatially referenced sets of polygons, and then use the polygons to estimate of size- and environmentally-dependent vital rates. Orthosomosaic maps were aligned between the first and second surveys using the Georeferencer GDAL plugin (QGIS Development Team 2021) in QGIS because the georeferencing of the orthomosaics was only accurate to ~5m. This plugin allows users to manually find and mark common points on both images and set them as reference points for transformation between the old coordinate system (map coordinates of survey 2) and the new coordinate system (map coordinates of survey 1). We used the thin plate splines transformation for the coordinate systems and nearest neighbor resampling method (GDAL contributors 2022). The reference points are available in the supplementary materials.

In addition to drawing polygons around the individual plants, we drew polygons around ground truth targets of known sizes (2-8 per site, depending on ease of site access and slope aspect of the terrain within it). We calculated target sizes on the image and then computed the ratio of calculated areas under the polygon vs the known sizes of the targets. We then re-scaled all computed sizes of plants using this ratio. Once all sizes, flowering information, and survival data were digitized, we used the `sf` (Pebesma 2018) and `dplyr` (Wickham

et al. 2022) *R* packages to join all the ramet data into a single dataset.

Table 4.1: Locations of the study populations of Carpobrotus, drone mission dates to monitor their population dynamics, and the number of unique ramets identified in orthomosaic generated from the first mission at each population.

| Site | Lat | Lon | Date Flight 1 | Date Flight 2 | N Ramets |
|---|---|---|---|---|---|
| Colares | 38.81 | -9.48 | 3/24/2019 | 3/18/2020 | 267 |
| Foxton | -40.46 | 175.22 | 10/15/2018 | 10/22/2019 | 1055 |
| Havatselet | 32.36 | 34.86 | 4/19/2018 | 4/26/2019 | 603 |
| Melkboss | -33.71 | 18.45 | 9/24/2018 | 9/23/2019 | 105 |
| Praia de Areao | 40.52 | -8.78 | 3/20/2019 | 3/17/2020 | 320 |
| Rarangi | -41.42 | 174.04 | 10/24/2018 | 11/5/2019 | 1741 |
| Rooisand | -34.35 | 19.09 | 9/12/2018 | 9/15/2019 | 1559 |
| Rough Island | -41.27 | 173.11 | 10/18/2018 | 11/4/2019 | 391 |
| Springfontein | -34.43 | 19.41 | 9/16/2018 | 9/17/2019 | 68 |
| St Francis | -34.18 | 24.82 | 9/10/2018 | 9/10/2019 | 361 |
| Struisbaai | -34.81 | 20.06 | 9/4/2018 | 9/7/2019 | 194 |
| Vogelgat | -34.40 | 19.32 | 9/16/2018 | 9/16/2019 | 64 |
| Whirinaki | -39.38 | 176.89 | 10/29/2018 | 10/19/2019 | 226 |

To understand how environmental variation constrains or facilitates invasion by *Carpobrotus* species, we compiled a data set of georeferenced *Carpobrotus* records from across the globe from GBIF (GBIF 2023). These data were subsequently cleaned with the `CoordinateCleaner` *R* package (Zizka et al. 2019, Figure 4.1) to remove occurrences that are associated with botanical gardens, cities, and other spatial confounds. We downloaded monthly ERA5 climate data from 2018 - 2020 for temperature ($\theta_t$), precipitation ($\theta_p$), and soil water content ($\theta_{s,i}$ where $i$ is the soil layer with values 1-3 to denote 0-7cm, 7-30cm, and 30-100cm deep, respectively). ERA5 data has a resolution of $0.25^{o} \times 0.25^{o}$. We wanted to estimate climate values more precisely for each GBIF occurrence and for the sites we sampled for demographic data. Therefore, environmental covariates were krigged for each occurrence point, plus our site coordinates using the `krigR` *R* package (Kusch & Davy, 2022).

Two key characteristics of Mediterranean biomes is a cool, wet winter and a warm dry summer. We hypothesized that competition for water drives the success of *Carpobrotus* species in their new environment. We expected that differences in the seasonal patterns of temperature and precipitation across sites and regions would also contribute to variance in performance. Therefore, we aggregated data by wet and dry season (i.e. $\theta_s$ values for the wet season, and $\theta_s$ values for the dry season). Finally, we computed how many standard deviations each of our sites was from the global mean value of each environmental covariate to create a standardized environmental value for each site in our data set (Compagnoni et al. 2021).

To test our hypotheses introduced above, we needed to derive fitness estimates for each population to address our hypotheses. Integral Projection Models (IPMs) are mathematical models that track the distribution and abundance of a continuously distributed trait (*e.g.* plant size) in a population through time (Ellner, Childs, & Rees 2016). Additionally, they can be used to estimate (among other quantities) the per-capita growth rate of a population ($\lambda$). Regression modeling is a means of linking trait- and environmentally dependent information to vital rate functions that comprise an IPM. We modeled size-dependent survival (Bernoulli model with logit link, $s_a(z, \theta)$), size at time $t+1$ ($z'$) conditional on survival and size at time $t$ (Gaussian model with identity link and size dependent variance, $G(z'|z, \mu_g, \sigma_g, \theta)$), size-dependent probability of flowering (Bernoulli model with logit link, $p_f(z, \theta)$), and size-dependent flower production (negative binomial model with log link, $r_f(z, \theta)$). In addition to size, we included environmental covariates ($\theta$, described in previous paragraph). We constructed Bayesian generalized linear mixed models (GLMMs) and compared different structures using WAIC, posterior predictive checks, and visual inspection of predictions against the observed data. We selected the best models chosen by WAIC that also made realistic predictions as determined by the latter graphical checks. Each of the potential forms used $\theta_t, \theta_p$ and one of the $\theta_{s,i}$ values as predictors, and

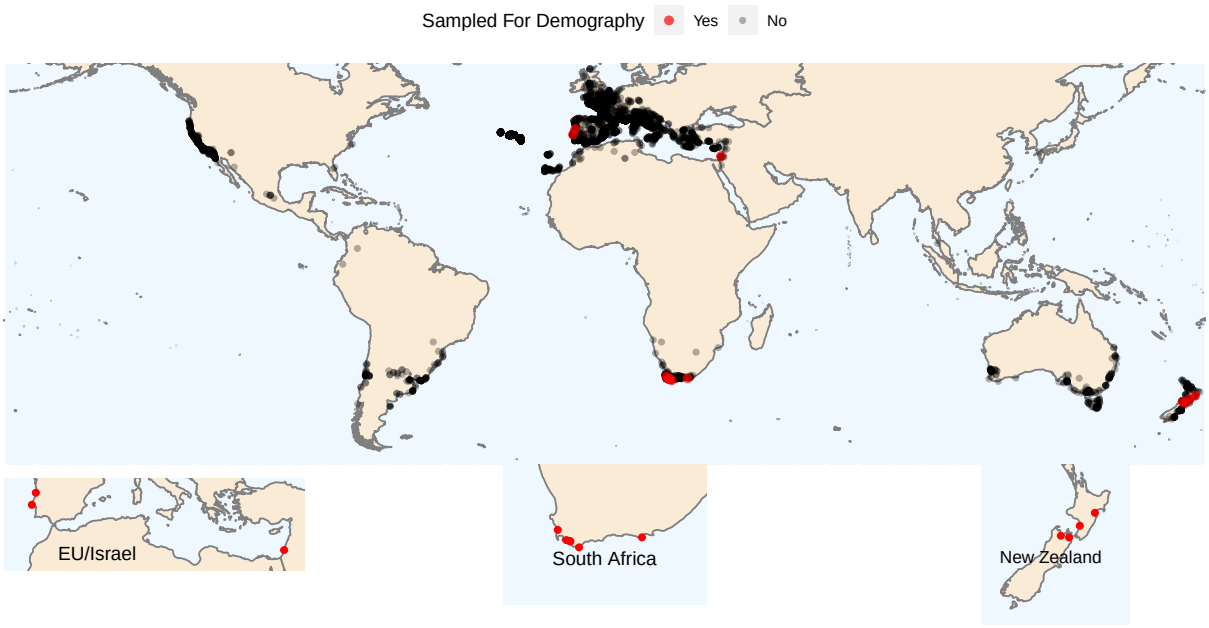



**Figure 4.1**: Geographic distribution of all GBIF occurrences for *Carpobrotus edulis*, *C. acinaciformis*, and *C. chilensis* used for climate envelope calculations (in black) and all sites used for demographic modeling (in red). Insets show demographic sampling locations within each region.

interactions between size and environmental covariates were tested as well. Because the responses of $p_f(z, \theta)$ and $r_f(z, \theta)$ are based on state at time $t$, we included environmental covariates from the year before the initial demographic sampling (i.e. $t-1$ to $t$), whereas $s_a(z, \theta)$ and $G(z'|z, \sigma, \theta)$ to $t+1$ used environmental data from the year between demographic samples. All candidate models included a population-specific intercept term and interaction between size and nativity. All vital rate models were implemented with the `brms` $R$ package (Bürkner et al. 2018). See appendix 7 for a Complete description of vital rate models used in the IPM.

In addition to the vital rates above, we also modeled the size of new recruits at time $t+1$ ($r_a(z')$), and estimated probabilities for flower pod survival ($p_{fv}$) and maturation ($p_m$) to time $t+1$, immediate seed germination ($g_i$), total viability ($v_s$), and seed establishment probability ($p_e$), and seed bank germination ($g_{sb}$) and survival ($s_{sb}$). Data for $r_a(z')$ was collected from the Rooisand site in South Africa using the seedling plot methods described above. We conducted literature reviews to gather estimates of other seed-related parameters, and these were estimated by averaging over the reported estimates. For two vital rates ($p_e$ and $s_{sb}$), we could not find values in the literature. Therefore, we conducted simulations to examine a reasonable range of values ($1^{-5} - 1^{-4}$ and $1^{-2} - 1^{-1}$, respectively) to examine how our results changed as a function of these values. Our conclusions are not sensitive to these values (see Appendix 7).

We used the parameters and functions described above to generate an IPM for each site (Ellner, Childs & Rees 2016). Our IPM included 1 continuous trait, size ($z, z'$), and 3 discrete states - mature fruits ($mf$), seedlings ($sdl$), and seeds in the seed bank ($sb$) (see Appendix 7 for life cycle diagram, complete equations, and parameter distributions describing the IPM). We combined the vital rate functions described above into an IPM in $R$ using the `ipmr` package (Levin et al. 2021). Point estimates of the deterministic per-capita growth rate for each site, $\lambda_i$ were computed from IPMs parameterized with mean values of the posterior distribution for each parameter estimated in the Bayesian regression models described above. We then used the posterior distributions for each regression model to create 4000 $\lambda_i$ values representing a range of plausible IPMs for each site.

Our goal was to determine both the shape and magnitude of a climate driver's effect on $\lambda$ *and* the relative importance of each driver for *Carpobrotus*'s fitness at a global scale, with the expectation that precipitation would be the primary environmental driver of fitness. To address the first goal, we constructed GAMs with the estimated $\lambda$ values as the response and each climate driver as a predictor. GAMs are well suited to this task because population responses to a climate driver may be non-linear across the range of sampling (Ellner, Childs, & Rees 2016). To address the second goal, we conducted an Life Table Response Experiment (LTRE) with $\lambda$ as the response and the spatially varying IPM parameters as predictors (Caswell 2001, Ellner, Childs & Rees 2016). LTREs decompose the contributions of a parameter, vital rate, or specific element of a discretized projection kernel, to $\lambda$ by constructing a (non-) linear model between the response ($\lambda$) and predictor (parameters, vital rates, kernel elements, Caswell 2001, Ellner, Childs & Rees 2016). Specifically, we used a random forest to estimate the relationship between the posterior draws for $\lambda$ and the IPM regression parameters. Random forests are well suited for this analysis because they are non-parametric, non-linear models that can handle high dimensional data (Breiman 2001). We assessed the importance of each variable by randomizing parameter values and computing the change in mean squared error (MSE) of the of the overall regression tree. Larger increases in MSE for a given parameter value indicate that it is more important in predicting $\lambda$, and therefore contributing more to the observed value of $\lambda$ (Ellner, Childs & Rees 2016). We implemented all of the LTRE analyses using the `randomForest` $R$ package (Liaw & Weiner 2002).

**Results**

IPMs yielded $\lambda_i$ values between 0.6 and 1.3 for the 13 studied populations of *Carpobrotus*. GAMs for $\lambda_i$ as a function of each climate variable show differing responses, though no relationship was statistically significant. Climate parameters were generally of low importance in predicting $\lambda$ values, yielding little evidence of a relationship between climate and population performance (Figure 4.2). Dry season precipitation showed the most negative, yet non-significant, effect on $\lambda_i$ ($p = 0.243$, soil water content results not shown here). All other regressions of $\lambda$ on a climate driver were not significant.

The random forest LTRE explained 92% of the variance in $\lambda$ across sites. Decomposing the contributions of

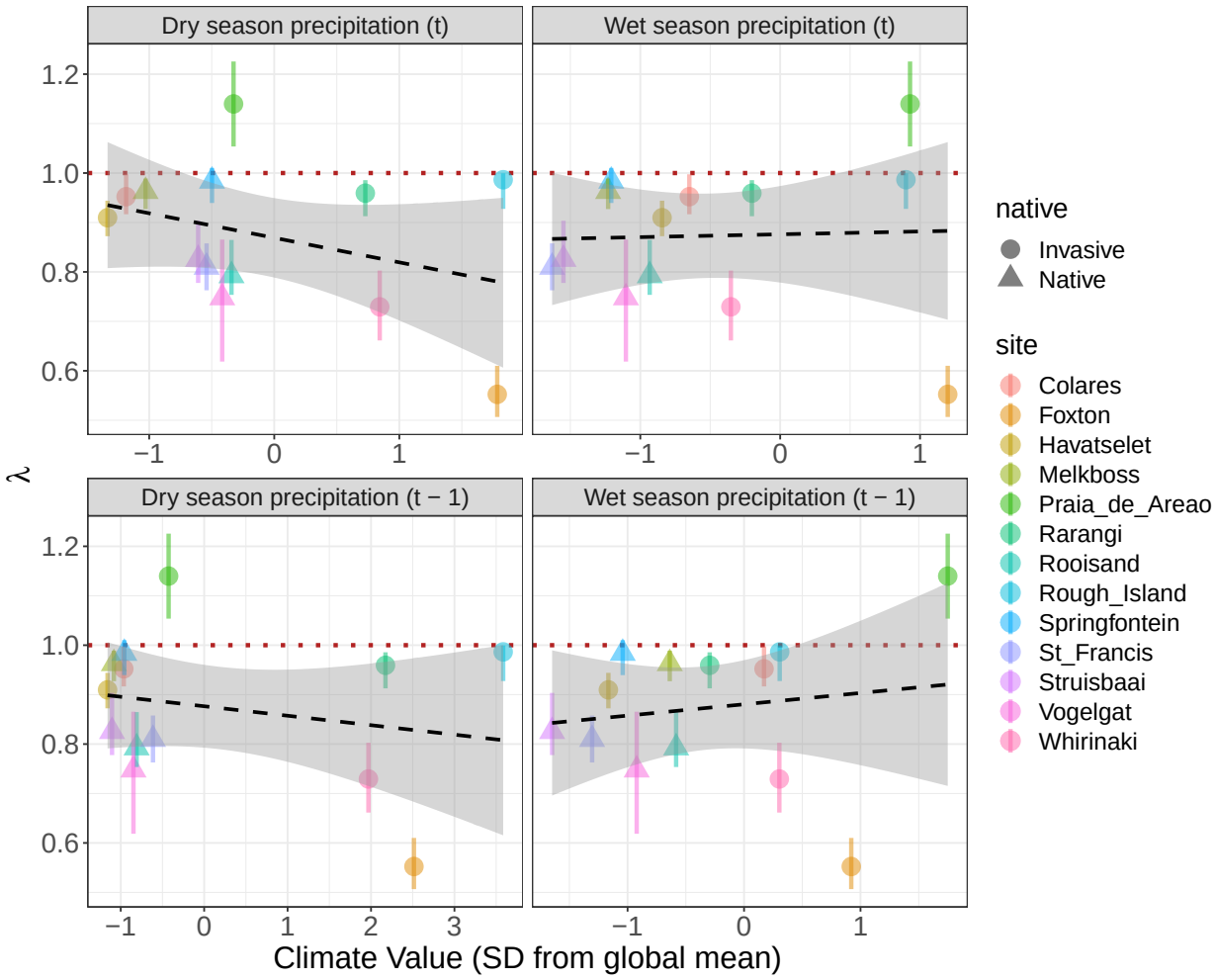



**Figure 4.2**: Relationship between the population growth rate $\lambda$ of our 13 studied Carpobrotus populations and the climate variables included in the IPMs. Black dashed lines represent the fit from the GAMs of `lambda ~ <climate_variable>`, and red dotted lines indicate $\lambda = 1$, meaning the population is demographically stable. None of the relationships shown here are statistically significant.

each parameter showed that site specific random effects were the most important parameters in the model. Climate related vital rate parameters were of substantially lesser importance than these random effects, but many, including wet and dry season precipitation's impact on growth variance and the probability of flowering, were included in the top 20 most important parameters in the IPMs (Figure 4.3).

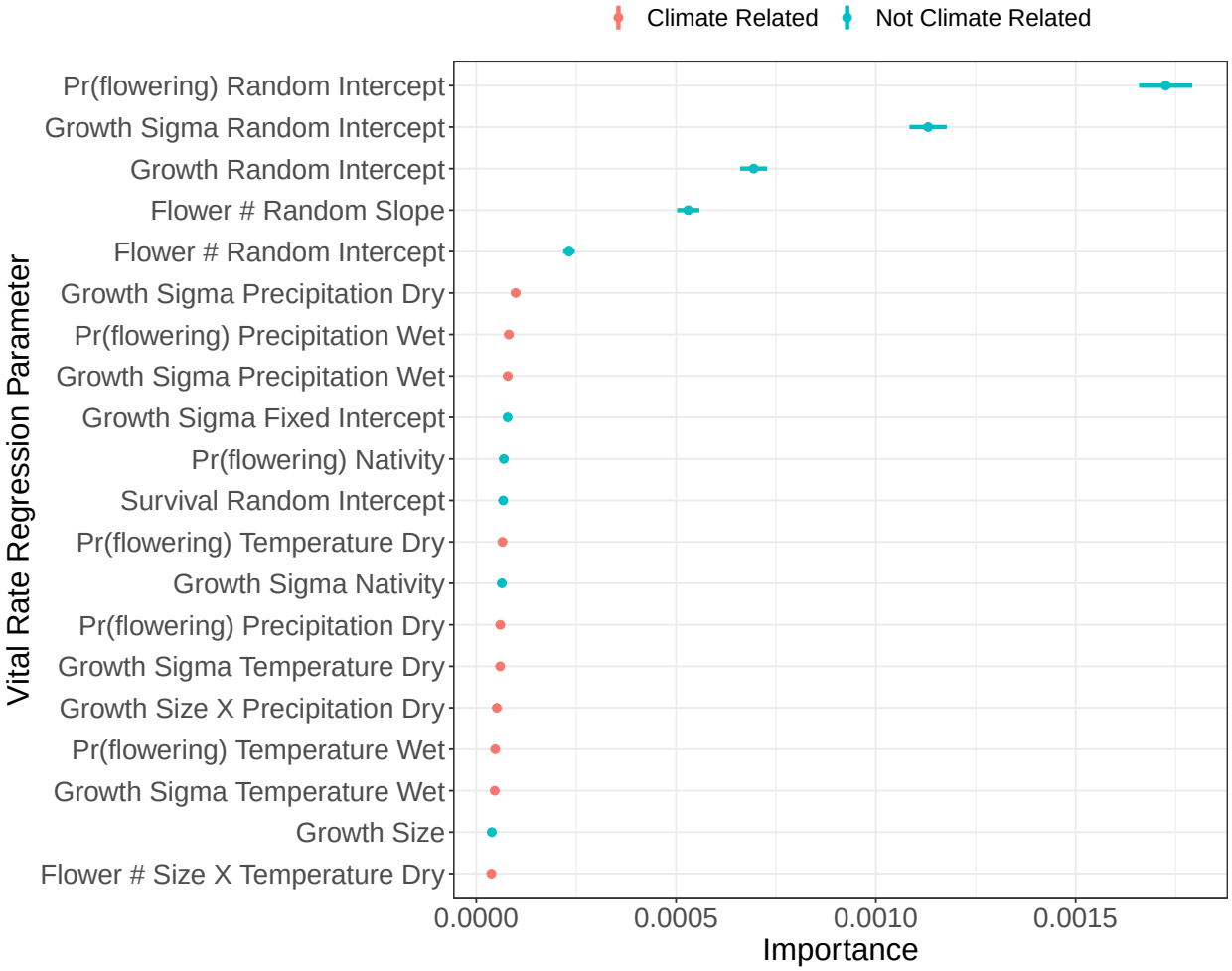



**Figure 4.3**: Top 20 most important variables and their importance in explaining $\lambda$, as determined by the random forest. Climate-related parameters are orders of magnitude less important than site-specific random effects, indicating that unmeasured variation is explaining more variance in $\lambda$.

### Discussion

Understanding how climate shapes the population dynamics of invasive species is one of the missing pieces in the puzzle of conservation biology (Beaury et al. 2020). Here we examined the drivers of population performance in 13 populations of the widespread, invasive species *Carpobrotus spp.* across three continents. Our models report a considerable amount of variation in their population growth rates across populations, ranging from $\lambda \approx 0.55$ at Foxton in New Zealand to $\lambda \approx 1.15$ at Praia de Areao in Portugal (Figure 4.2). However, much of the variation $\lambda$ is explained by unmeasured variables - the LTRE results showed that the random intercepts and slopes contribute the most to explaining variance in $\lambda$ (Figure 4.3). While climate factors were much less important than random variables in the LTRE, sites with lower dry season precipitation appeard to have higher $\lambda$ values (Figure 4.2). This result is not statistically significant and does not yield robust support for our hypothesis that *Carpobrotus* has higher performance in drier sites.

Many factors may contribute to a species success at local scales and might explain why unmeasured variables contribute more to the variation in $\lambda$. First, phylogenetic and functional distinctiveness of Carpobrotus to the local native plant community may have a greater fitness impact than abiotic conditions when species are introduced to novel settings, as this distinctiveness can lead to novel resource use (Mathakutha et al. 2019) and weaker effects of competitors on vital rates and $\lambda$ (Levin et al. 2020). Second, *Carpobrotus* was intentionally introduced in much of its invaded range, largely for ornamental purposes (Portela et al. 2022). Horticultural introductions usually entail human preference for desirable characteristics, such as heavy flower production, which are also desirable characteristics for an invader that escapes cultivation (Caño et al. 2008). Such anthropogenic selection may induce founder effects for more fit phenotypes than are present in its native range, allowing it to become more dominant (Mayr 1942). Third, we note that the introduced range sometimes contains multiple *Carpobrotus* species and their hybrids (Suehs et al. 2004). The high fitness in a few of the populations could be the result of hybridization between multiple introduced populations could yield super-hybrids (Novoa et al. *in review*).

The successional stage of the *Carpobrotus* population might also contribute to the unexplained site variation in the population growth rate $\lambda$. *Carpobrotus* tends to be an early successional species in its native range, losing out to other Fynbos species as the ecosystem matures after fire (T. Masayiti, personal communication). However, the dense litter that forms under plants as they grow inhibits germination of native species and alters the soil geochemistry in the invaded range, which may delay or halt succession in habitats where native species are not adapted to the presence of *Carpobrotus* (Santoro *et al.* 2011, Novoa *et al.* 2013, Compoy *et al.* 2018). An alteration of the successional trajectory of the ecosystem and the associated competitive release that comes with that could easily mask the smaller effect of climate on overall fitness. Documenting the succesional stage of populations cannot be easily done post-hoc as the orthomosaic outputs from the drone do not preserve the vertical structure of the populations. Point clouds generated in the process of orthomosaic construction could provide a suitable alternative (Pix4D SA, 2019), and would not require subjective assessments of successional stage.

In contrast to expectations, fitness was not significantly higher in the introduced range than the native range. Both the best and worst performing populations were present in the introduced range. For example, Praia de Areao was the highest performing, and is situated in full sun along the Portuegese coast. This is a relatively mild winter climate, which may be helpful to the population. In contrast, Foxton is located on the southwestern side of New Zealand's North Island. Strong winds, cooler temperatures, and shifting dunes are all factors that may contribute to this site's relative lack of performance.

Future work investigating climate drivers of invasions should consider that spatial environmental gradients alone are not sufficient for robust statistical inference. We specifically recommend replicating sites both within *and* across climate bands. A recent simulation study illustrated that replicating 10 sites within a climate band for 3 years provides as much power for detecting climate-demography relationships as a 20-30 year data set at a single site (Compagnoni et al. 2022). Our study design only yielded three sites that could serve as climate replicates of each other, while all other sites were spread across a considerable distances and were in different climate bands. This yielded a broad climate gradient, but also did not allow partitioning of site specific variability and climate signal (because $n = 1$ for every level of climate in our data set). Our power to detect climate driven effects would be 0.3-0.5, depending on the inter-site climate correlations (Compagnoni et al. 2022). For 13 spatial replicates, we would likely need to sample for at least 10 transitions to detect a climate effect with confidence, and/or increase our replication within climate bands (Compagnoni et al. 2022).

In general, concerns that climate change may favor the fitness and population growth of a harmful invasive species are not supported by this study. Climate is not found to be a strong predictor of demographic success for *Carpobrotus*. This could be because anthropogenic factors are stronger predictors of invasion success than ecological ones (Wohlwend et al. 2021). While climate change may open up new areas to invasion, long distance dispersal events are still required for a species to realize this new niche space. In the Anthropocene, these dispersal events are far more likely to occur due to economic factors than demographic ones, as evidenced by the fact that initial introductions of *Carpobrotus* were almost exclusively intentional throughout the invaded range. It is therefore more important to focus on limiting trade of these species to curtail spread at the macroecological scale.

**Citations**


1. Bürkner, P.-C. (2018). Advanced Bayesian Multilevel Modeling with the R Package brms. The R Journal, 10(1), 395-411. doi:10.32614/RJ-2018-017
2. Liaw, A. & Wiener, M. (2002). Classification and Regression by randomForest. R News 2(3), 18–22.
3. Breiman, L. (2001). Random Forests. Machine Learning 45: 5-32. DOI: https://doi.org/10.1023/A:1010933404324
4. Pebesma, E., 2018. Simple Features for R: Standardized Support for Spatial Vector Data. The R Journal 10 (1), 439-446, https://doi.org/10.32614/RJ-2018-009
5. Wickham, H., François, R., Henry, L., & Müller, K. (2021). dplyr: A Grammar of Data Manipulation. R package version 1.0.7. https://CRAN.R-project.org/package=dplyr
6. Zizka, A., Silvestro, D., Andermann, T., Azevedo, J., Duarte Ritter, C., Edler, D., *et al.* (2019). "CoordinateCleaner: standardized cleaning of occurrence records from biological collection databases." Methods in Ecology and Evolution 10. DOI: 10.1111/2041-210X.13152 (URL: https://doi.org/10.1111/2041-210X.13152), R package version 2.0-20
7. Portela, R., Barreiro, R., Alpert, P., Xu, C-Y., Webber, B.L., & Roiloa, S.R. (2022). Comparative invasion ecology of *Carpobrotus* from four continents: responses to nutrients and competition. Jourcnal of Plant Ecology, *in press*. https://doi.org/10.1093/jpe/rtac034
8. Wohlwend, M.R., Craven, D., Weigelt, P., Seebens, H., Winter, M., Kreft, H., *et al.* (2021). Anthropogenic and environmental drivers shape diversity of naturalized plants across the Pacific. Diversity and Distributions. https://doi.org/10.1111/ddi.13260
9. Santoro, R., T. Jucker, M. Carranza, and A. T. R. Acosta. (2011). Assessing the Effects of Carpobrotus Invasion on Coastal Dune Soils. Does the Nature of the Invaded Habitat Matter? Community Ecology 12 (2): 234–240. doi:10.1016/j.jnc.2011.08.003.
10. Novoa, A., L. González, L. Moravcová, and P. Pyšek. (2013). Constraints to Native Plant Species Establishment in Coastal Dune Communities Invaded by Carpobrotus edulis: Implications for Restoration. Biological Conservation 164: 1–9. doi:10.1016/j.biocon.2013.04.008.
11. Campoy, J.G., Acosta, A.T.R., Affre, L., Barreiro, R., Brundu, G., Buisson, E. *et al.* (2018). Monographs of invasive plants in Europe: *Carpobrotus*. Botany Letters. https://doi.org/10.1080/23818107.2018.1487884
12. Pimentel D, Zuniga R, Morrison D. 2005 Update on the environmental and economic costs associated with alien-invasive species in the United States. Ecol. Econ. 52, 273–288. (doi:10.1016/j.ecolecon.2004.10.002)
13. Vila M et al. (2011). Ecological impacts of invasive alien plants: a meta-analysis of their effects on species, communities and ecosystems. Ecology Letters 14, 702–708. (doi:10.1111/j.1461-0248.2011.01628.x)
14. Levin, S.C., Crandall, R.M., Pokoski, T., Stein, C., & Knight, T.M. (2020). Functional and phylogenetic distinctiveness explain alien plant population responses to competition. Proceedings B 287: 20201070 http://dx.doi.org/10.1098/rspb.2020.1070
15. Lowry E, Rollinson EJ, Laybourn AJ, Scott TE, Aiello- Lammens ME, Gray SM, Mickley J, Gurevitch J. (2013). Biological invasions: a field synopsis, systematic review, and database of the literature. Ecol. Evol. 3, 182–196. (doi:10.1002/ece3.431)
16. Jeschke JM, Heger T. (2018). Invasion biology: hypotheses and evidence. Boston, MA: CABI Publishing.
17. Novoa, A., L. González, L. Moravcová, and P. Pyšek. (2012). Effects of Soil Characteristics, Allelopathy and Frugivory on Establishment of the Invasive Plant Carpobrotus edulis and a Co-Occuring Native, Malcolmia littorea. PLoS ONE 7 (12): e53166. doi:10.1371/journal.pone.0053166.

18. Novoa, A., and L. González. (2014). Impacts of Carpobrotus edulis (L.) NE Br. On the Germination, Establishment and Survival of Native Plants: A Clue for Assessing Its Competitive Strength. PLoS ONE 9 (9): e107557. doi:10.1371/journal.pone.0107557.

19. D'Antonio, C.M. & Mahall, B.E. (1991). Root Profiles and Competition between the Invasive, Exotic Perennial, Carpobrotus edulis, and Two Native Shrub Species in California Coastal Scrub. American Journal of Botany 78(7): 885-894.

20. Guisan, A. and Thuiller, W. (2005) Predicting Species Distribution: Offering More than Simple Habitat Models. Ecology Letters, 8, 993-1009. http://dx.doi.org/10.1111/j.1461-0248.2005.00792.x

21. Liu, C., Wolter, C., Xian, W., & Jeschke, J. (2020). Species distribution models have limited spatial transferability for invasive species. Ecology Letters 23(11): 1682-1692.

22. Merow, C., Bois, S.T., Allen, J.M., Xie, Y., Silander Jr., J.A. (2017). Climate change both facilitates and inhibits invasive plant ranges in New England. Proceedings of the National Academy of Sciences. E3276–E3284. www.pnas.org/cgi/doi/10.1073/pnas.1609633114

23. Garcia-Callejas, D., Molowny-Horas, R., & Retana, J. (2017). Projecting the distribution and abundance of Mediterranean tree species under climate change: a demographic approach. Journal of Plant Ecology 10(5): 731-743.

24. Broennimann, O. & Guisan, A. (2008). Predicting current and future biological invasions: both native and invaded ranges matter. Biology Letters 4(5):585-589.

25. Elith, J., Kearney, M., & Phillips, S. (2010). The art of modelling range-shifting species. Methods in Ecology and Evolution 1(4): 330-342.

26. QGIS Development Team (2022). QGIS Geographic Information System. Open Source Geospatial Foundation Project. http://qgis.osgeo.org

27. Briscoe, N.J., Elith, J., Salguero-Gomez, R., Lahoz-Monfort, J.J., Carnac, J.S., Glijohann K.M. *et al.* (2019). Forecasting species range dynamics with process-explicit models: matching methods to applications. Ecology Letters 22: 1940-1956.

28. Ellner, S.P., Childs, D.Z., Rees, M. (2016) Data-driven modelling of structured populations: a practical guide to the integral projection model. Basel, Switzerland: Springer International Publishing AG

29. Levin, S.C., Childs, D.Z., Compagnoni, A., Evers, S., Knight, T.M., & Salguero-Gomez, R. (2021). ipmr: flexible implementation of Integral Projection Models. Methods in Ecology and Evolution 12 (10): 1826-1834.

30. Caswell, H. (2001) Matrix population models: construction, analysis, and interpretation, 2nd edn. Sunderland, MA: Sinauer Associates Inc

31. Prentis, P.J., Wilson, J.R.U., Dormontt, E.E., Richardson, D.M., & Lowe, A.J. (2008). Adaptive evolution in invasive species. Trends in Plant Science 13(6): 288-294.

32. Compagnoni A., Pardini, E., & Knight, T.M. (2021). Increasing temperature threatens an already endangered coastal dune plant. Ecosphere 12(3): e03454

33. Klausmeyer, K.R. & Shaw. M.R. (2009). Climate Change, Habitat Loss, Protected Areas and the Climate Adaptation Potential of Species in Mediterranean Ecosystems Worldwide. PLoS ONE 4(7): e6392. https://doi.org/10.1371/journal.pone.0006392

34. Caño, L., Escarre, J., Fleck, I., Blanco-Moreno, J.M., & Sans, F.X. (2008). Increased fitness and plasticity of an invasive species in its introduced range: a study using *Senecio pterophorus*. Journal of Ecology 96: 468-476.

35. Beaury, E.M., Fusco, E.J., Jackson, M.R. et al. Incorporating climate change into invasive species management: insights from managers. Biological Invasions 22, 233–252 (2020). https://doi.org/10.1007/s10530-019-02087-6

36. Allen, J., & Bradley, B.A. (2016). Out of the weeds? Reduced plant invasion risk with climate change in the continental United States. Biological Conservation 203, 306-312. https://doi.org/10.1016/j.biocon.2016.09.015

37. Ziska, L.H. (2022) Plant Invasions, Rising CO2, and Global Climate Change. Chapter in Global Plant Invasions. 71-87. https://link.springer.com/chapter/10.1007/978-3-030-89684-3_4

38. Cavender-Bares, J., Schneider, F.D., Santos, M.J. et al. Integrating remote sensing with ecology and evolution to advance biodiversity conservation. Nature Ecology & Evolution. 6, 506–519 (2022). https://doi.org/10.1038/s41559-022-01702-5

39. Lake, T.A., Runquist, R.D.B, & Moeller, D.A. (2020). Predicting range expansion of invasive species: Pitfalls and best practices for obtaining biologically realistic projections. Diversity and Distributions 26(12): 1767-1779.

40. Innangi, M., Marzialetti, F., Di Febbraro, M., Acosta, A.T.R., De Simone, W., Frate, L., Finizio, M., Perna, P.V., & Carranza, M.L. (2023). Coastal Dune Invaders: Integrative Mapping of Carpobrotus sp. pl. (Aizoaceae) Using UAVs. Remote Sensing. 15, 503. https://doi.org/10.3390/rs15020503

41. GDAL/OGR contributors (2022). GDAL/OGR Geospatial Data Abstraction software Library. Open Source Geospatial Foundation. URL https://gdal.org DOI: 10.5281/zenodo.5884351

42. Muñoz Sabater, J., (2019): ERA5-Land hourly data from 1981 to present. Copernicus Climate Change Service (C3S) Climate Data Store (CDS). (Accessed on < 02-02-2022 >), 10.24381/cds.e2161bac

43. Mathakutha, R., Steyn, C., le Roux, P. C., Blom, I. J., Chown, S. L., Daru, B. H., Ripley, B. S., Louw, A., & Greve, M. (2019). Invasive species differ in key functional traits from native and non-invasive alien plant species. Journal of Vegetation Science 30(5): 994-1006. https://doi.org/10.1111/jvs.12772

44. Compagnoni, A., Evers, S., & Knight, T.M. (2022). Spatial replication should be prioritized to advance our understanding of population responses to climate. BioRxiv https://doi.org/10.1101/2022.06.24.497542

45. Lee-Yaw, J.A., McCune, J.L., Pironon, S., & Sheth, S.N. (2021). Species distribution models rarely predict the biology of real populations. Ecography 2022: e05877, doi: 10.1111/ecog.05877

46. Caswell, H. (1989) The analysis of life table response experiments. I. Decomposition of effects on population growth rate. Ecological Modelling, 46, 221–237.

47. Jackson, J., Lawson, C.S., Adelmant, C., Huhtala, E., Fernandes, P., Hodgson, R., *et al.* (2022). Short-range multispectral imaging is an inexpensive, fast, and accurate approach to estimate biodiversity in a temperate calcareous grassland. Ecology and Evolution 12(12): e9623. https://doi.org/10.1002/ece3.9623

48. Ellner, S.P. & Rees, M. (2006). Integral projection models for species with complex demography. American Naturalist 167(3): 410-428

# Synthesis

There is a long history in plant and animal demography of considering demographic responses across relatively small spatial and temporal grains, and a current need to expand this understanding to broader scales (Crone et al. 2010, Roemer et al. 2021). Macroecologists consider how climate influences the distribution and abundance of species, and use models to predict how these patterns may change with climate change. However, demographic studies that focus on vital rates (survival, growth, fecundity) provide a mechanistic understanding for how species respond to climate (e.g. Compagnoni et al. 2021). Studying demography at macroecological scales can provide a better understanding of how different populations are responding to climate, and what drivers are most important for their vital rates and persistence.

Moving towards broader spatial scales in demography has been limited by having appropriate tools for synthesis and rapid demographic data collection. This dissertation aims to fill these gaps and study populations of the same species that occur across different continents. I develop and apply new tools that can help researchers address questions related to demography and population dynamics across various temporal and spatial scales, and applies them to a focal genus to understand how climate affects the population dynamics of a problematic invasive plant species. The computational framework introduced in chapter 2, `ipmr`, is both a standalone tool for creating Integral Projection Models (IPMs) of any population (Chapter 4, Bogdan et al. 2021) and is an underlying engine for synthesis work (Chapter 3). In Chapter 3, I demonstrate how to use `ipmr` and `Rpadrino` to implement many IPMs across many spatial scales, and the supplementary materials provide a framework for addressing the relationship between range centrality and species fitness, demonstrating their use for macro-ecological applications. In chapter 4, I use a novel data collection techniques (unmanned aerial vehicles) and `ipmr` to study the demographic responses of an invasive succulent plant species across multiple continents that span its native and exotic ranges. I show that climate does not constrain the fitness of this plant species as hypothesized, suggesting that other factors may mediate its ability to colonize and thrive.

### Advances in standalone IPMs

Integral projection models have become a popular tool for researchers across many disciplines to model population performance, evolutionary and life history trajectories, and species fitness in continuously structured populations (Ellner, Childs, & Rees 2016). Following their introduction in 2000 (Easterling, Ellner & Dixon 2000), researchers have developed increasingly complex IPMs to capture the nuanced responses to drivers such as climate (e.g. Hindle et al. 2018), competition (Adler et al. 2010), and changes in continuous structure within discrete groups (e.g. two sex models, Erickson et al. 2017). In parallel with these theoretical and applied developments, researchers created computational tools that help create and analyze IPMs (Metcalf et al. 2013, Shefferson et al. 2020). Despite the uptick in the popularity of IPMs, computational tools to implement IPMs have not kept pace with development of theory.

In chapter 2, I introduced the *R* package, `ipmr`. `ipmr` provides a miniature domain specific language to implement a wide range of IPMs. `ipmr` provides functionality for basic analyses, and extensive documentation to aid researchers in implementing more complex analyses (Appendix 1 & Appendix 2). At present, it can handle both simple and general IPMs (*sensu* Ellner & Rees 2006), deterministic and two types of stochastic models (Rees & Ellner 2009), and density/frequency-independent or -dependent models. Furthermore, `ipmr` handles two-sex models (W. Petry, personal communication), and nested combinations of continuous traits within discrete trait models (e.g. Ellner, Childs & Rees 2016, Chapter 6, Metcalf et al. 2009), which were either exceptionally difficult or impossible to implement with previously written *R* packages.

`ipmr` provides a syntax that mimics the mathematical notation of IPMs, and it does not attempt to abstract over the vital rate modeling step of IPM construction. These two features provide a number of advantages over prior work. First, these two features enable researchers to use any functional form they desire to specify the relationship between a trait value and a vital rate. These two features enable researchers to leverage new developments in regression modeling as they become available, rather than having to wait for `ipmr`'s maintainers to implement a way of integrating them into the package. As long as a researcher can write out the functional form of the response (or the regression technique includes a `predict()` method), then it

is available for use in `ipmr`. Second, there is no limit on the number of vital rates, structuring traits, and there are no rules on how these may relate to each other (though, in practice, having too many of either may result in painfully slow performance - the curse of dimensionality spares no one). This feature enables researchers to model populations of increasing complexity without having to think too hard about model implementation - the researcher can keep adding new sub-kernels to the IPM and `ipmr` will implement them. One final implication of decoupling vital rate modelling from IPM implementation is that raw data are not required to implement an IPM with `ipmr`. One can assume functional forms for different vital rates, work out reasonable ranges for parameter values, and then simulate populations. Prior to this development, researchers who wished to use an *R* package to implement their IPM would have to simulate data, fit regression models to that data, and then construct an IPM. This feature also enables PADRINO (introduced in chapter 3) to drastically reduce the amount of data it has to store to reconstruct published IPMs.

**Summary of new features for synthesis**

Another key piece of `ipmr`'s functionality that sets it apart from prior work is the `proto_ipm` data structure. This data structure provides a unified way of describing every IPM that one can implement with `ipmr`, and so is a common form that can be shared by both user-defined IPMs and ones defined by PADRINO, which is introduced in chapter 3. The `proto_ipm` is essentially a data frame that houses the symbolic definition of the sub-kernels, the parameters associated with this definition, and then the numerical details that implement the model (e.g. domain limits for continuous traits, integration rules, etc.). This means that once a user defines one or more `proto_ipm`s using their own data and/or data from PADRINO, a single set of computational machinery can be applied to implement their models, drastically reducing the amount of time required to conduct multi-model synthesis work.

PADRINO provides a database that stores peer-reviewed, published IPMs. `Rpadrino` leverages the aforementioned features of `ipmr` to reconstruct them for users. Specifically, PADRINO stores the mathematical form of published IPMs as text strings and `Rpadrino` makes use of *R*'s ability to parse text into code as well as modify code's evaluation environments to ensure that IPMs are reconstructed as published. `Rpadrino` breaks this down into two steps: re-generating `proto_ipm`s from PADRINO's symbolic model definitions, and then generating actual IPM objects. This segmenting of the steps enables users to combine their own IPMs with database-generated ones. Furthermore, the consistent structure of the outputs makes programming complex synthesis more straightforward, as there is type-consistency in the outputs.

In Appendix 4 of Chapter 3, I show how to use these larger data sets for synthetic work. Specifically, I use a single interface for implementing multiple models by re-constructing IPMs for 23 species and then computing sensitivities and elasticities of population growth rate, as well as mean lifetime recruit production. This appendix further demonstrates the advantages of using an expression based framework by demonstrating how to alter vital rate functions and compute function value perturbations. For example, I provide a case study for how to correct unrealistic functions in an IPM. This functionality allows users to make more complete use of the data stored in the database by correcting issues with published IPMs that would otherwise be unusable for a particular analysis. The combination of `Rpadrino` and PADRINO also provides a framework for theoreticians to experiment with the consequences of altering the relationships between traits and vital rates using realistic demographic information.

Appendix 5 of Chapter 3 demonstrates another important result for researchers interested in synthesis. This appendix combines data from user-defined models as well as other open access databases to informally address a long-standing question in ecology. This appendix leverages range maps from BIEN (Maitner *et al.* 2017) and COMPADRE (Salguero-Gomez *et al.* 2014) to estimate the relationship between range centrality and population growth rate ($\lambda$). In doing so, I provide a code template that can be repurposed by other researchers to clean and filter publicaly accessible data and conduct larger scale syntheses than were previously possible.

### Applications

Thus far, I have detailed only of what might be done with the novel tools I have developed, rather than what has been done with them. `ipmr` has been used to construct a simple model for a population of *Carpobrotus* species in Israel (Bogdan et al. 2020), examine the effects of harvesting and fire on three economically important tree species in the Western Ghats of India (Neeraja *et al.* 2022), explore the role of climate change on perennial grasslands (Andrzejak et al. 2023), and to construct an age and size structured model of the impacts of temperature on the invasive Asian Carp (Brook 2021).

In chapter 4, I use `ipmr` to generate an IPM that examines population performance at the local scale and species performance at a macro-ecological scale to investigate how climate affects the *Carpobrotus* genus across its native and invaded range. Contrary to my expectations, this work revealed that climate is not very important in determining the success of this invasive genus. There are a myriad of reasons why this may be the case, ranging from the statistical limitations (*i.e.* a lack of power) to the biological drivers (*e.g.* biotic interactions are more important and mask the relatively smaller climate effects). Climate's lack of importance in explaining $\lambda$ reveals the importance of both temporal and spatial scale in studies like this. Regarding the former, single years are often not sufficient to capture a relationship between climate variables and population performance, due to, for example, single year deviations from longer-term weather patterns may confound signal (Compagnoni et al. 2023). Regarding biological explanations, populations may experience different biotic interactions, and genetic bottlenecks that are imposed on a species via introduction to a new region (founder effects and subsequent adaptation).

It has been suggested that space might be substituted for time when studying various facets of biodiversity and population biology (Blois et al. 2013, but see Dammand 2019). As such, sampling multiple populations that experience different climates in a single year might provide insight into how a single population would respond to climate changes that occur across longer time scales. However, I found that sampling 13 populations spread across 4 continents was not sufficient to detect climate effects on demography. I argue that a more successful approach, in hindsight, would have been to increase replication within each climate band, rather than sampling populations that span across a broad climate gradient. This alternative approach would have allowed me to disentangle the within-band, across site variation on one hand, and the variation arising across climate bands on the other hand (Compagnoni et al. 2022).

### Limitations of these new frameworks

In addition to the limitations of the space for time substitutions, which are discussed in chapter 4 and in the preceding paragraph, there are other limitations to this dissertation's research. The *R* packages `ipmr` and `Rpadrino`, and the PADRINO database, for all of their flexibility, are not capable of implementing all possible IPMs. For example, `ipmr` is not able to accommodate time-lagged models (i.e. historical IPMs *sensu* Shefferson *et al.* 2020) or periodic models (Letcher *et al.* 2014). Despite `ipmr`'s ability to handle semi- and non-parametric models for vital rates, a general syntax for describing them in PADRINO has not yet been developed either. Thus, IPMs that include, for example, generalized additive models (Wood 2017) are not yet available for synthesis. Another, more general, limitation of PADRINO is that the data contained therein may not be biologically correct - the only criteria for inclusion in that database is that the model has received some sort of peer review (i.e. are either published in a peer reviewed journal, are part of an MSc. or PhD thesis, or a technical report). Therefore, the models may be all biologically unrealistic, even if they have passed peer review. For an example of how this limitation looks in practice (and how to overcome it provided it is recognized by the user), see Appendix 4 of Chapter 3.

### Future directions for research

Ecology faces the challenge of scaling our understanding of ecological systems across different spatial and temporal grains. To tackle this challenge, ecologists must develop methods and frameworks for integrating data and observations across scales. This dissertation contributes to advancing ecology by filling key gaps

in our ability to study demography at broad spatial and temporal scales. There are some key areas that are important to develop along this line in future research. The first recommendation for future research is with regard to chapters 2 and 3. The *R* packages `ipmr` and `Rpadrino` are promising tools for generating point estimate of demographic quantities. However, quantifying uncertainty is an essential exercise for every scientist, regardless of field. Currently, neither framework provides off the shelf functionality to propagate uncertainty in parameters/vital rates into uncertainty in the demographic quantities of interest. At the moment, an `ipmr` vignette is the only resource provided to users to demonstrate how one might generate a posterior distribution for the population growth rate $\lambda$. This documentation alone is certainly insufficient given that no robust inference can be made without an understanding of how the outcome of interest may vary. This is an important area of future development for both tools to reach their full potential.

The outcome of interest in most studies extends beyond $\lambda$. While `ipmr` contains functionality to calculate deterministic and stochastic eigenvectors, this *R* package does not contain functionality to calculate many other metrics. The documentation provides some overview of how to compute other metrics, though it is far from exhaustive (see Ellner, Childs, & Rees 2016). This limitation is largely by design, as combining too much into a single piece of software makes it harder to maintain (the principle of conscious decoupling). A third package is needed that contains functionality to conduct more complex analyses on `ipmr` and `Rpadrino` outputs.

There is an ever growing list of publications that make use of IPMs, and PADRINO will require constant attention to keep it updated. One priority for the digitization team should be to balance out the phylogenetic representation of the database. Currently, it is quite plant-heavy, which reflects my own personal interests but is not reflective of the sampling effort made by the field as whole. Synthesis requires consideration of sample sizes and the balance of each factor under consideration, so it would be ideal to allow users to make full use of data without having to correct for these issues as much as they currently may have to.

Ecologists are aware that large scale drivers may exhibit small effect sizes when studied at finer spatial or temporal grains. Chapter 4 highlights this phenomenon. There are two key next steps for this work. First, to ensure that future sampling adds additional sites that are close to existing ones, rather than attempting to add sites that occupy additional climate space. Second, consider other drivers of invasiveness simultaneously to understand relative importance of these drivers. These two improvements will likely yield far more insight than the broad scale sampling of climate alone discussed in Chapter 4.

### Concluding remarks

The computational framework introduced in chapters 2 and 3 provides a new way for researchers to generate and synthesize demographic data across a variety grains and scales. Their flexibility will hopefully provide researchers with a toolkit to address broader questions with the appropriate methods and statistical power. Despite their limitations, they have already found use in the literature. As they are open source and open access, both using and contributing to them can be a collaborative endeavor.

In practice, devising an appropriate sampling scheme for detecting climate drivers on demographic responses is far more nuanced and challenging than simply sampling far and wide, as I did in Chapter 4. Careful consideration must be given to the spatial and temporal scales of the expected effects in question, and to the level of replication required to detect them. On the other hand, regardless of the presence or absence of a detectable climate effect on fitness, we do know that invasive species are harmful to biodiversity and economic interests. We should not take the absence of an observed climate effect to indicate that a species poses no threat beyond its current distribution, and policy should reflect this existing knowledge.

### Citations

1. Bogdan, A., Levin, S.C., Salguero-Gomez, R. (2020) Demographic analysis of an Israeli Carpobrotus population. PLoSONE. https://doi.org/10.1371/journal.pone.0250879

2. Ellner, S.P., Childs, D.Z., Rees, M. (2016) Data-driven modelling of structured populations: a practical guide to the integral projection model. Basel, Switzerland: Springer International Publishing AG

3. Easterling, M.R., Ellner, S.P., & Dixon, P.M. (2000). Size specific sensitivity: applying a new structured population model. Ecology 81(3): 694-708.

4. Ellner, S.P. & Rees, M. (2006). Integral Projection Models for species with complex demography. The American Naturalist 167(3): 410-428.

5. Rees, M., & Ellner, S. P. (2009). Integral projection models for populations in temporally varying environments. Ecological Monographs, 79(4), 575–594. https://doi.org/10.1890/08-1474.1.

6. Maitner, B., Boyle B., Casler N., Condit R., Donoghue J., Duran S.M., et al. (2017) The bien r package: A tool to access the Botanical Information and Ecology Network (BIEN) database. Methods in Ecology and Evolution 9(2): 373-379. https://doi.org/10.1111/2041-210X.12861

7. Levin, S.A. (1989). The problem of pattern and scale in ecology. Ecology 73(6): 1943-1967.

8. Erickson, R.A., Eager, E.A., Brey, M.B., Hansen, M.J., & Kocovsky, P.M. (2017). An integral projection model with YY-males and application to evaluating grass carp control. Ecological Modelling 361: 14-25. https://doi.org/10.1016/j.ecolmodel.2017.07.030

9. Metcalf, C. J. E., McMahon, S. M., Salguero-Gómez, R., & Jongejans, E. (2013). IPMpack: An R package for integral projection models. Methods in Ecology and Evolution., 4(2), 195–200. https://doi.org/10.1111/2041-210x.12001

10. Shefferson, R. P., Kurokawa, S., & Ehrlen, J. (2020). LEFKO3: Analysing individual history through size-classified matrix population models. Methods in Ecology and Evolution. https://doi. org/10.1111/2041-210X. 13526

11. Metcalf, C. J. E., Horvitz, C. C., Tuljapurkar, S., & Clark, D. A. (2009). A time to grow and a time to die: a new way to analyze the dynamics of size, light, age, and death of tropical trees. Ecology, 90(10), 2766-2778.

12. Salguero-Gómez, R, Jones, O.R., Archer, C.A., Buckley, Y.M., Che-Castaldo, J., Caswell, C., Hodgson, D., Scheuerlein, A., Conde, D.A., Brinks, E., de Buhr, H., Farack, C., Gottschalk, F., Hartmann, A., Henning, A., Hoppe, G., Roemer, G., Runge, J., Ruoff, T., etal. (2014) The COMPADRE Plant Matrix Database: an online repository for plant population dynamics. Journal of Ecology 103: 202-218. https://doi.org/10.1111/1365-2745.12334

13. Neeraja, U. V., Saneesh, C. S., Dyda, V., Reddy, H., Yadama, G. N., & Knight, T. M. (2022). Harvesting has variable effects on demographic rates and population growth across three dry forest tree species. Biotropica, 54(6), 1376-1389.

14. Brook, M. (2021). Impacts of temperature on Asian Carp life history and population growth (Master's thesis, University of Waterloo).

15. Letcher, B.H., Schueller, P., Bassar, R.D., Nislow, K.H., Coombs, J.A.,Sakrejda, K. et al. (2014). Robust estimates of environmental effects on population vital rates: an integrated capture-recapture model of seasonal brook trout growth, survival and movement in a stream network. Journal of Animal Ecology 84(2): 337-352. https://doi.org/10.1111/1365-2656.12308

16. Wood, S.N. (2017). Generalized Additive Models: An Introduction with R (2nd edition). Chapman and Hall/CRC.

17. Compagnoni, A., Levin, S., Childs, D.Z. et al. Herbaceous perennial plants with short generation time have stronger responses to climate anomalies than those with longer generation time. Nat Commun 12, 1824 (2021). https://doi.org/10.1038/s41467-021-21977-9

18. Hindle, B., Rees, M., Sheppard, A.W., Quintana-Ascensio, P.F., Menges, E.S., & Childs, D.Z. (2018). Exploring population responses to environmental change when there's never enough data: a factor

analytic approach. Methods in Ecology and Evolution 9(11): 2283-2293. https://doi.org/10.1111/2041-210X.13085

19. Adler, P.B., Ellner, S.P., & Levine, J.M. (2010). Coexistence of perennial plants: an embarrassment of niches. Ecology Letters 13(8): 1019-1029. https://doi.org/10.1111/j.1461-0248.2010.01496.x

20. Compagnoni, A., Evers, S., & Knight, T.M. (2022). Spatial replication should be prioritized to advance our understanding of population responses to climate. BioRxiv https://doi.org/10.1101/2022.06.24.497542

21. Crone, E.E., Menges, E.S., Ellis, M.M., Bell, T., *et al.* (2010). How do plant ecologists use matrix population models? Ecology Letters 14(1): 1-8.

22. Roemer, G., Dahlgren, J.P., Saluero-Gomez, R., Stott, I.M., Jones, O.R. (2023). Plant demographic knowledge is biased towards short-term studies of temperate-region herbaceous perennials. BioRXiv. https://doi.org/10.1101/2021.04.25.441327

23. Andrzejak, M., Knight, T.M., Plos, C., & Korell, L. (2023). Reproductive success mediates the effects of climate change and grassland management on plant populations dynamics. BioRXiv. https://doi.org/10.1101/2023.04.26.538388

24. Blois, J.L., Williams, J.W., Fitzpatrick, M.C., Jackson, S.T., & Ferrier, S. (2013). Space can substitute for time in predicting climate-change effects on biodiversity. Proceedings of the National Academy of Sciences 110(23): 9374-9379. https://doi.org/10.1073/pnas.1220228110

25. Dammand, C. (2019). A Critique of the Space-for-Time Substitution Practice in Community Ecology. Trends in Ecology and Evolution 34(5): 416-421. https://doi.org/10.1016/j.tree.2019.01.013

## Acknowledgements

I could not, nor would not, have done this work alone. I first want to thank my supervisors, Tiffany Knight and Roberto Salgeuro-Gomez. Your mentorship has been invaluable, and I cannot think of two more supportive advisors in academia. I also want to thank the members of the Knight Lab over the last 6 years. None of these ideas would have happened were it not for the unrelenting support, intellectual brilliance, and comic relief you provided. So thanks to Aldo Compagnoni, Sanne Evers, Neeraja Venkataraman, Valentin Stefan, Dylan Craven, Joanne Bennett, Leana Zoller, Neeraja Venkataraman, Elena Motivans, Martin Andrzejak, Amibeth Thompson, Demetra Rakosy, Lotte Korell, and Michael Wohlwend. Finally, I owe a debt to T. Michael Anderson and Miles Silman, who introduced me to ecological research, and encouraged me to continue with it after undergrad.

I also owe a great debt to those who kept me sane throughout this whole ordeal, particularly Andrea Pacheco, Eduardo Arle, the three Leana's, Mike, Sanne, Max, Valentin, Martin, and Amibeth. Thanks for the impromptu nights out, the parties, and the Hops nights with Franz and Stine (and also to the latter for putting up with us all these years).

I would be remiss in skipping over my friends back home: Alex, Henry, Ryan, Christian, Andy, Keith, Drew, and the rest of the Denver team. Home was always a reset switch for me, and it was largely because of you that it felt that way.

Speaking of home, I now have to thank my family. To my parents, Steve and Chris, my brothers, Daniel and M, and my girlfriend, Larissa - I have no words to convey how grateful I am for your love and support, and how impossible this would have been without it. Finally, I must thank my grandparents Betty and Buddy Levin for their dedication to education and their support of mine, and to Robert and Lib Conner for inspiring my love of nature.

# Appendix 1: `ipmr` Case Study 1

# Case Study 1: Bogdan et al. 2020

## Two versions of a simple model

The first case study in this manuscript creates a model for *Carpobrotus spp.* The dataset used in this case study was collected in Havatselet Ha'Sharon, a suburb of Tel Aviv, Israel. The data were collected by drones taking aerial imagery of the population in successive years. Images were combined into a single high-resolution orthomosaic and georeferenced so the map from year 2 laid on top of the map from year 1. Flowers on each plant were counted using a point layer, and polygons were drawn around each ramet to estimate sizes and survival from year to year. Plants that had 0 flowers were classified as non-reproductive, and any plant with 1 or more flowers was classified as reproductive. This led to four regression models - survival, growth conditional on survival, probability of flowering, and number of flowers produced conditional on flowering. Finally, plants present in year 2 that were not present in year 1 were considered new recruits. The mean and variance of their sizes were computed, and this was used to model the recruit size distribution.

The resulting IPM is a simple IPM (i.e. no discrete states, one continuous state variable). The data that the regressions are fit to are included in the `ipmr` package, and can be accessed with `data(iceplant_ex)` (the name comes from the common name for *Carpobrotus* species, which is "iceplants").

The IPM can be written on paper as follows:

1. $n(z', t+1) = \int_L^U K(z', z) n(z, t) dz$
2. $K(z', z) = P(z', z) + F(z', z)$
3. $P(z', z) = s(z) * G(z', z)$
4. $F(z', z) = p_f(z) * r_s(z) * p_r * r_d(z')$

The components of each sub-kernel are either regression models or constants. Their functional forms are given below:

5. $Logit(s(z)) = \alpha_s + \beta_s * z$
6. $G(z', z) = f_G(z', \mu_G(z), \sigma_G)$
7. $\mu_G(z) = \alpha_G + \beta_G * z$
8. $Logit(p_f(z)) = \alpha_{p_f} + \beta_{p_f} * z$
9. $Log(r_s(z)) = \alpha_{r_s} + \beta_{r_s} * z$
10. $r_d(z') = f_{r_d}(z', \mu_{r_d}, \sigma_{r_d})$

$\alpha s$ and $\beta s$ correspond to intercepts and slopes from regression models, respectively. Here, $f_G$ and $f_{r_d}$ are used to denote normal probability density functions. The other parameters are constants derived directly from the data itself.

```
library(ipmr)
```

```
## Warning: package 'ipmr' was built under R version 4.2.3
```

```
## Welcome to `ipmr`! `browseVignettes('ipmr')` to get started.
```

```
data(iceplant_ex)

# growth model.
```

```r
grow_mod <- lm(log_size_next ~ log_size, data = iceplant_ex)
grow_sd  <- sd(resid(grow_mod))

# survival model

surv_mod <- glm(survival ~ log_size, data = iceplant_ex, family = binomial())

# Pr(flowering) model

repr_mod <- glm(repro ~ log_size, data = iceplant_ex, family = binomial())

# Number of flowers per plant model

flow_mod <- glm(flower_n ~ log_size, data = iceplant_ex, family = poisson())

# New recruits have no size(t), but do have size(t + 1)

recr_data <- subset(iceplant_ex, is.na(log_size))

recr_mu  <- mean(recr_data$log_size_next)
recr_sd  <- sd(recr_data$log_size_next)

# This data set doesn't include information on germination and establishment.
# Thus, we'll compute the realized recruitment parameter as the number
# of observed recruits divided by the number of flowers produced in the prior
# year.

recr_n   <- length(recr_data$log_size_next)

flow_n   <- sum(iceplant_ex$flower_n, na.rm = TRUE)

recr_pr  <- recr_n / flow_n


# Now, we put all parameters into a list. This case study shows how to use
# the mathematical notation, as well as how to use predict() methods

all_params <- list(
  surv_int = coef(surv_mod)[1],
  surv_slo = coef(surv_mod)[2],
  repr_int = coef(repr_mod)[1],
  grow_int = coef(grow_mod)[1],
  grow_slo = coef(grow_mod)[2],
  grow_sdv = grow_sd,
  repr_slo = coef(repr_mod)[2],
  flow_int = coef(flow_mod)[1],
  flow_slo = coef(flow_mod)[2],
  recr_n   = recr_n,
  flow_n   = flow_n,
  recr_mu  = recr_mu,
  recr_sd  = recr_sd,
  recr_pr  = recr_pr
)
```

The next chunk generates a couple constants used to implement the model. We add 20% to the smallest and largest observed sizes to minimize eviction, and will implement the model with 100 meshpoints.

NB: `L` is multiplied by 1.2 because the log of the minimum observed size is negative, and we want to extend the size range to make it more negative. If `L` were positive, we'd multiply by 0.8.

```
L <- min(c(iceplant_ex$log_size,
           iceplant_ex$log_size_next),
         na.rm = TRUE) * 1.2

U <- max(c(iceplant_ex$log_size,
           iceplant_ex$log_size_next),
         na.rm = TRUE) * 1.2

n_mesh_p <- 100
```

We now have the parameter set prepared, and have the boundaries for our domains set up. We are ready to implement the model.

We start with the function `init_ipm()`. This function has five arguments: `sim_gen`, `di_dd`, `det_stoch`, `kern_param`, and `uses_age`. For now, we will ignore the last argument, as it is covered in case study 2. The first 4 arguments specify the type of IPM we are building:

1. `sim_gen`: `"simple"`/`"general"`
    - A. **simple**: This describes an IPM with a single continuous state variable and no discrete stages.
    - B. **general**: This describes and IPM with either more than one continuous state variable, one or more discrete stages, or both of the above. Basically, anything other than an IPM with a single continuous state variable.
2. `di_dd`: `"di"`/`"dd"`
    - A. **di**: This is used to denote a **d**ensity-**i**ndependent IPM.
    - B. **dd**: This is used to denote a **d**ensity-**d**ependent IPM.
3. `det_stoch`: `"det"`/`"stoch"`
    - A. **det**: This is used to denote a deterministic IPM. If this is the third argument of `init_ipm`, `kern_param` must be left as `NULL`.
    - B. **stoch**: This is used to denote a stochastic IPM. If this is the third argument of `init_ipm`, `kern_param` must be specified.

This particular model is deterministic, as there are no data on temporal or spatial changes in vital rates. An introduction to stochastic models is available here. This example does not make use of the final argument, `kern_param`, because it is not a stochastic model, so we'll ignore it for now.

Once we've decided on the type of model we want, we create the model class using one of the two options for each argument. Since there is no stochasticity, we can leave the fourth argument empty (its default is `NULL`). This case study is a simple, density independent, deterministic IPM, so we use the following:

```
carpobrotus_ipm <- init_ipm(sim_gen = "simple", di_dd = "di", det_stoch = "det")
```

After we have initialized our IPM, we need to start adding sub-kernels using the `define_kernel()` function. These correspond to equations 3 and 4 above. We'll start with the `P` kernel. It contains functions that describe survival of individual ramets, and, if they survive, their new sizes. Note that in `ipmr`, the order in which we define kernels for an IPM makes no difference, so we could also start with the `F` if we wanted to.

1. Survival is modeled with a logistic regression to predict the probability of survival to $t + 1$ based on the size of the ramet at $t$ (`surv_mod`). In order to use the coefficients from that model to generate a

survival probability, we need to know the inverse logit transformation, or, a function that performs it for us based on the linear predictor.

2. Size at $t+1$ is modeled with a Gaussian distribution with two parameters: the mean and standard deviation from the mean. The mean value of size at $t+1$ (`mu_G`) is itself a linear function of size at $t$ and is parameterized with coefficients from the linear model (`grow_mod`). The standard deviation is a constant derived from the residual variance from the linear model we fit.

We start providing information on the `P` kernel by giving it a `name`. The name is important because we can use it to reference this kernel in higher level expressions later on. It can have any name we want, but `P` is consistent with the literature in this field (e.g. Easterling, Ellner & Dixon 2000, Ellner & Rees 2006). Next, we write the `formula`. The `formula` is the form of the kernel, and should look like Equation 3, without the $z$ and $z'$ arguments.

```
carpobrotus_ipm <-  define_kernel(
  proto_ipm = carpobrotus_ipm,
  name      = "P",
  formula   = s * G,
  ...
)
```

The `family` comes after `formula`. It describes the type of transition the kernel is implementing. `family` can be one of 4 options:

1. `"CC"`: Continuous state -> continuous state.
2. `"DC"`: discrete state -> continuous state.
3. `"CD"`: continuous state -> discrete state.
4. `"DD"`: discrete state -> discrete state.

Since this is a simple IPM with only 1 continuous state variable and 0 discrete state variables, the `family` will always be `"CC"`. In general IPMs, this will not always be true.

```
carpobrotus_ipm <-  define_kernel(
  proto_ipm = carpobrotus_ipm,
  name      = "P",
  formula   = s * G,
  family    = "CC",
  ...
)
```

We've now reached the `...` section of `define_kernel()`. The `...` part takes a set of named expressions that represent the vital rate functions we described in equations 5-7 above. The names on the left hand side of the `=` should appear either in the `formula` argument, or in other parts of the `....` The expressions on the right hand side should generate the values that we want to plug in. For example, Equation 5 ($Logit(s(z)) = \alpha_s + \beta_s * z$) makes use of the `plogis` function in the `stats` package to compute the survival probabilities from our linear model. The names of the coefficients match the names in the `all_params` object we generated above. Another thing to note is the use of `z_1` and `z_2`. These are place-holders for $z, z'$ in the equations above. `ipmr` will generate values for these internally using information that we provide in some of the next steps.

```
carpobrotus_ipm <-  define_kernel(
  proto_ipm = carpobrotus_ipm,
  name      = "P",
  formula   = s * G,
  family    = "CC",
  G         = dnorm(z_2, mu_g, grow_sdv),
  mu_g      = grow_int + grow_slo * z_1,
```

```
  s         = plogis(surv_int + surv_slo * z_1),
  ...
)
```

After setting up our vital rate functions, the next step is to provide a couple more kernel-specific details:

1. `data_list`: this is the `all_params` object we created above. It contains the names and values of all the constants in our model.
2. `states`: A list that contains the names of the state variables in the kernel. In our case, we've just called them `"z"`. The `states` argument controls the names of the variables `z_1` and `z_2` that are generated internally. We could just as easily call them something else - we would just have to change the vital rate expressions to use those names instead. For example, in this model, $z, z'$ is the log-transformed surface area of ramets. We could abbreviate that with `"log_sa"`. In that case, `z_1,z_2` would become `log_sa_1, log_sa_2` in the vital rate expressions.
3. `evict_cor`: Whether or not to correct for eviction (Williams et al. 2012).
4. `evict_fun`: If we decide to correct for eviction, then a function that will correct it. In this example, we use `ipmr`'s `truncated_distributions` function. It takes two arguments: `fun`, which is the abbreviated form of the probability function family (e.g. "norm" for Gaussian, "lnorm" for log-normal, etc.), and `target`, which is the name in `...` that it modifies.

```
carpobrotus_ipm <-  define_kernel(
  proto_ipm = carpobrotus_ipm,
  name      = "P",
  formula   = s * G,
  family    = "CC",
  G         = dnorm(z_2, mu_g, grow_sdv),
  mu_g      = grow_int + grow_slo * z_1,
  s         = plogis(surv_int + surv_slo * z_1),
  data_list = all_params,
  states    = list(c("z")),
  evict_cor = TRUE,
  evict_fun = truncated_distributions(fun    = "norm",
                                      target = "G")
)
```

We've now defined our first sub-kernel. The next step is to repeat this process for the `F` kernel, which is Equations 4 and 8-10.

```
carpobrotus_ipm <-  define_kernel(
  proto_ipm = carpobrotus_ipm,
  name      = "F",
  formula   = recr_pr * r_s * r_d * p_f,
  family    = "CC",
  r_s       = exp(flow_int + flow_slo * z_1),
  r_d       = dnorm(z_2, recr_mu, recr_sd),
  p_f       = plogis(repr_int + repr_slo * z_1),
  data_list = all_params,
  states    = list(c("z")),
  evict_cor = TRUE,
  evict_fun = truncated_distributions(fun    = "norm",
                                      target =  "r_d")
)
```

We've defined our sub-kernels. The next step is tell `ipmr` how to implement it numerically, and pro-

vide the information needed to generate the correct iteration kernel. To do this, we use `define_impl()`, `define_domains()`, and `define_pop_state()`.

The first function tells `ipmr` which integration rule to use, which state variable each kernel acts on (`state_start`), and which state variable each kernel produces (`state_end`). The format of the list it takes in the `kernel_impl_list` argument can be tricky to implement right, so the helper function `make_impl_args_list()` makes sure everything is formatted properly. The `kernel_names` argument can be in any order. The `int_rule`, `state_start`, and `state_end` arguments are then matched to kernels in the `proto_ipm` based on the order in the `kernel_names`. Note that, at the moment, the only integration rule that's implemented is `"midpoint"`. `"b2b"` (bin to bin) and `"cdf"` (cumulative density functions) are in the works, and others can be implemented by popular demand.

```
carpobrotus_ipm <-  define_impl(
  proto_ipm = carpobrotus_ipm,
  make_impl_args_list(
    kernel_names = c("P", "F"),
    int_rule     = rep('midpoint', 2),
    state_start    = rep('z', 2),
    state_end      = rep('z', 2)
  )
)
```

Next, we define the range of values that our state variable, $z$/`z` can take on. This is done using `define_domains`. The `...` argument should have named vectors. The name should match the name of the `state`/`domain`. The first value in the vector is lower boundary, the second entry is the upper boundary, and the third entry is the number of bins to divide that range into.

```
carpobrotus_ipm <-  define_domains(
  proto_ipm = carpobrotus_ipm,
  z         = c(L, U, n_mesh_p)
)
```

Finally, we define the initial population state. In this case, we just use a uniform vector, but we could also use custom functions we defined on our own, or pre-specified vectors. The name of the population vector should be the name of the `state`/`domain`, with an `"n_"` attached to the front.

```
carpobrotus_ipm <-  define_pop_state(
  proto_ipm = carpobrotus_ipm,
    n_z     = rep(1/100, n_mesh_p)
)
```

Up until this point, all we've done is add components to the `proto_ipm`. We now have enough information in `proto_ipm` object to build a model, iterate it, and compute some basic quantities. `make_ipm()` is the next function we need. It generates the vital rate functions from the parameters and integration details we provided, and then builds the sub-kernels. At this point, it checks to make sure that everything makes numerical sense (e.g. there are no negative values or `NA`s generated). If we set `iterate = TRUE`, `make_ipm()` also generates expressions for iterating the model internally, and then evaluates those for the number of iterations supplied by `iterations`. There are a number of other arguments to `make_ipm()` that can prove helpful for subsequent analyses. `return_main_env` is one of these. The `main_env` object contains, among other things, the integration mesh and bin width information specified in `define_domains()`. We'll need the meshpoints and bin width for the analyses we'll do in the Further Analyses section, so we'll set `return_main_env = TRUE`.

```
carpobrotus_ipm <-  make_ipm(
  proto_ipm       = carpobrotus_ipm,
  iterate         = TRUE,
  iterations      = 100,
```

```
  return_main_env = TRUE
)


asymp_grow_rate <- lambda(carpobrotus_ipm)
asymp_grow_rate
```

```
##    lambda
## 0.9759257
```

We see that the population is projected to shrink slightly. `ipmr` computes all values by iteration. Our measure of the asymptotic growth rate is the ratio $\frac{N_{t+1}}{N_t}$ for the final iteration of the model. If we are concerned about whether or not we've iterated our model enough to trust this value, we have two options: check for convergence using the helper `is_conv_to_asymptotic()`, or create the full iteration kernel, compute the dominant eigenvalue of that, and compare our estimate with the value obtained by iteration.

```
# Option 1: is_conv_to_asymptotic

is_conv_to_asymptotic(carpobrotus_ipm)
```

```
## lambda
##   TRUE
```

```
# Option 2: generate iteration kernel and compute eigenvalues

K <- make_iter_kernel(carpobrotus_ipm)

lam_eigen <- Re(eigen(K$mega_matrix)$values[1])

# If we've iterated our model enough, this should be approximately 0 (though
# maybe a little off due to floating point errors).

asymp_grow_rate - lam_eigen
```

```
##       lambda
## 3.463896e-14
```

We can also inspect our sub-kernels, the time series of the population trait distribution, and make alterations to our model using some helpers from `ipmr`.

```
# Sub-kernels have their own print method to display the range of values
# and some diagnotic information.

carpobrotus_ipm$sub_kernels
```

```
## $P
##
##  Minimum value: 0, maximum value: 0.08763
## All entries greater than or equal to 0: TRUE
##
## $F
##
##  Minimum value: 0, maximum value: 0.02512
## All entries greater than or equal to 0: TRUE
```

```
# Extract the time series of the population state (n_z),
# and the n_t+1/n_t values (lambda)
```

```
pop_time_series    <- carpobrotus_ipm$pop_state$n_z
lambda_time_series <- carpobrotus_ipm$pop_state$lambda

# Next, we'll tweak the intercept of the p_f function and re-fit the model.

new_proto_ipm      <- carpobrotus_ipm$proto_ipm

# The parameters setter function takes a list. It can replace single values,
# create new values, or replace the entire parameter list, depending on how you
# set up the right hand side of the expression.

parameters(new_proto_ipm) <- list(repr_int = -0.3)

new_carp_ipm <- make_ipm(new_proto_ipm,
                         iterations = 100)

lambda(new_carp_ipm)
```

```
##    lambda
## 0.9720439
```

Next, we'll go through an alternative implementation of the model using `predict(surv_mod)` instead of the mathematical form of the linear predictors. After that, we'll explore a couple additional analyses to see what is going on with this population of iceplants.

## Using predict methods instead

We can simplify the code a bit more and get rid of the mathematical expressions for each regression model's link function by using `predict()` methods instead. The next chunk shows how to do this. Instead of extracting parameter values, we put the model objects themselves into the `data_list`. Next, we specify the `newdata` object where the name corresponds to the variable name(s) used in the model in question, and the values are the domain you want to evaluate the model on.

Above, we added parts to the `carpobrotus_ipm` object in a stepwise fashion. However, every `define_*` function in `ipmr` takes a `proto_ipm` as the first argument and returns a `proto_ipm` object. Thus, we can also use the `%>%` operator from the `magrittr` package to chain together the model creation pipeline. The `%>%` is included in `ipmr`, so we don't need to load any additional packages to access it. This example will demonstrate that process as well.

```
pred_par_list <- list(
  grow_mod = grow_mod,
  grow_sdv = grow_sd,
  surv_mod = surv_mod,
  repr_mod = repr_mod,
  flow_mod = flow_mod,
  recr_n   = recr_n,
  flow_n   = flow_n,
  recr_mu  = recr_mu,
  recr_sd  = recr_sd,
  recr_pr  = recr_pr
)

predict_method_carpobrotus <- init_ipm(sim_gen = "simple",
                                       di_dd = "di",
```

```
                                    det_stoch = "det") %>%
define_kernel(
  name      = "P",
  formula   = s * G,
  family    = "CC",
  G         = dnorm(z_2, mu_g, grow_sdv),
  mu_g      = predict(grow_mod,
                      newdata = data.frame(log_size = z_1),
                      type = 'response'),
  s         = predict(surv_mod,
                      newdata = data.frame(log_size = z_1),
                      type = "response"),
  data_list = pred_par_list,
  states    = list(c('z')),
  evict_cor = TRUE,
  evict_fun = truncated_distributions("norm", "G")
) %>%
define_kernel(
  name      = "F",
  formula   = recr_pr * r_s * r_d * p_f,
  family    = "CC",
  r_s       = predict(flow_mod,
                      newdata = data.frame(log_size = z_1),
                      type = "response"),
  r_d       = dnorm(z_2, recr_mu, recr_sd),
  p_f       = predict(repr_mod,
                      newdata = data.frame(log_size = z_1),
                      type = "response"),
  data_list = pred_par_list,
  states    = list(c("z")),
  evict_cor = TRUE,
  evict_fun = truncated_distributions("norm", "r_d")
) %>%
define_impl(
  make_impl_args_list(
    kernel_names = c("P", "F"),
    int_rule     = rep('midpoint', 2),
    state_start    = rep('z', 2),
    state_end      = rep('z', 2)
  )
) %>%
define_domains(
  z = c(L, U, n_mesh_p)
)  %>%
define_pop_state(
  n_z = rep(1/100, n_mesh_p)
) %>%
make_ipm(iterate    = TRUE,
         iterations = 100)
```

## Further analyses

Many research questions require a bit more than just computing asymptotic growth rate ($\lambda$). Below, we will compute the kernel sensitivity, elasticity, $R_0$, and generation time. First, we will define a couple of helper functions. These are not included in `ipmr`, but will eventually be implemented in a separate package that can handle the various classes that `ipmr` works with.

The first is sensitivity of $\lambda$ to perturbations in the projection kernel. Here, we can use the `right_ev` and `left_ev` functions in `ipmr` to get the right and left eigenvectors, and then compute the sensitivity surface.

**Technical note:** `right_ev` and `left_ev` both compute eigenvectors via iteration. `left_ev` generates a transpose iteration using the `state_start` and `state_end` information contained in the `proto_ipm` object (defined in `define_impl`, for a full overview of transpose iteration, see Ellner & Rees, 2006, Appendix A). Because the form of for left iteration is different from the default of right iteration, `left_ev()` will always have to iterate a model. On the other hand, `right_ev` will always check to see if the model is already iterated. If so, and the population's trait distribution has converged to its asymptotic state, then it will just pull out the final distribution from the `ipm` object, scale it to sum to 1, and then return that without re-iterating anything. If not, it will use the final trait distribution from the `ipm` object as the starting point and iterate the model for 100 iterations (this can be adjusted as needed using the `iterations` argument to `right_ev`). If this fails to converge, it will return `NA` with a warning.

It is also important to note that we have a second argument here named `d_z`. This is the width of the integration bins. We'll see how to get that from our IPM below.

```
sens <- function(ipm_obj, d_z) {

  w <- right_ev(ipm_obj)[[1]]
  v <- left_ev(ipm_obj)[[1]]

  return(
    outer(v, w) / sum(v * w * d_z)
  )

}
```

Next, we can define a function to compute the elasticity of $\lambda$ to kernel perturbations. This uses the `sens` function from above, and the `lambda()` function from `ipmr`.

```
elas <- function(ipm_obj, d_z) {

  K           <- make_iter_kernel(ipm_obj)$mega_matrix

  sensitivity <- sens(ipm_obj, d_z)

  lamb        <- lambda(ipm_obj)

  out         <- sensitivity * (K / d_z) / lamb

  return(out)

}
```

We may also want to compute the per-generation population growth rate. The function below uses the sub-kernels contained in the `carpobrotus_ipm` object to do that.

```
R_nought <- function(ipm_obj) {
```

```
  Pm <- ipm_obj$sub_kernels$P
  Fm <- ipm_obj$sub_kernels$F

  I  <- diag(dim(Pm)[1])

  N  <- solve(I - Pm)

  R  <- Fm %*% N

  return(
    Re(eigen(R)$values)[1]
  )

}
```

Finally, generation time is a useful metric in many analyses. Below, we make use of our `R_nought` function to compute one version of this quantity (though other definitions exist. Covering those is beyond the scope of this case study).

```
gen_time <- function(ipm_obj) {

  lamb     <- unname(lambda(ipm_obj))

  r_nought <- R_nought(ipm_obj)

  return(log(r_nought) / log(lamb))
}
```

We need to extract the `d_z` value and meshpoints from the IPM we built. We can extract this information in a list form using the `int_mesh()` function from `ipmr` on our IPM object. The `d_z` in this case will be called `d_z` because we named our domain `"z"` when we implemented the model. However, it will have a different name if the `states` argument in `define_kernel` has different values. Once we have that, we can begin computing all the values of interest. For example, if `states = list(c("dbh", "height"))`, then `int_mesh()` would a return a list with `d_dbh` and `d_height`.

```
mesh_info <- int_mesh(carpobrotus_ipm)

sens_mat <- sens(carpobrotus_ipm, mesh_info$d_z)
elas_mat <- elas(carpobrotus_ipm, mesh_info$d_z)

R0    <- R_nought(carpobrotus_ipm)
gen_T <- gen_time(carpobrotus_ipm)

R0
```

```
## [1] 0.5079748
```

```
gen_T
```

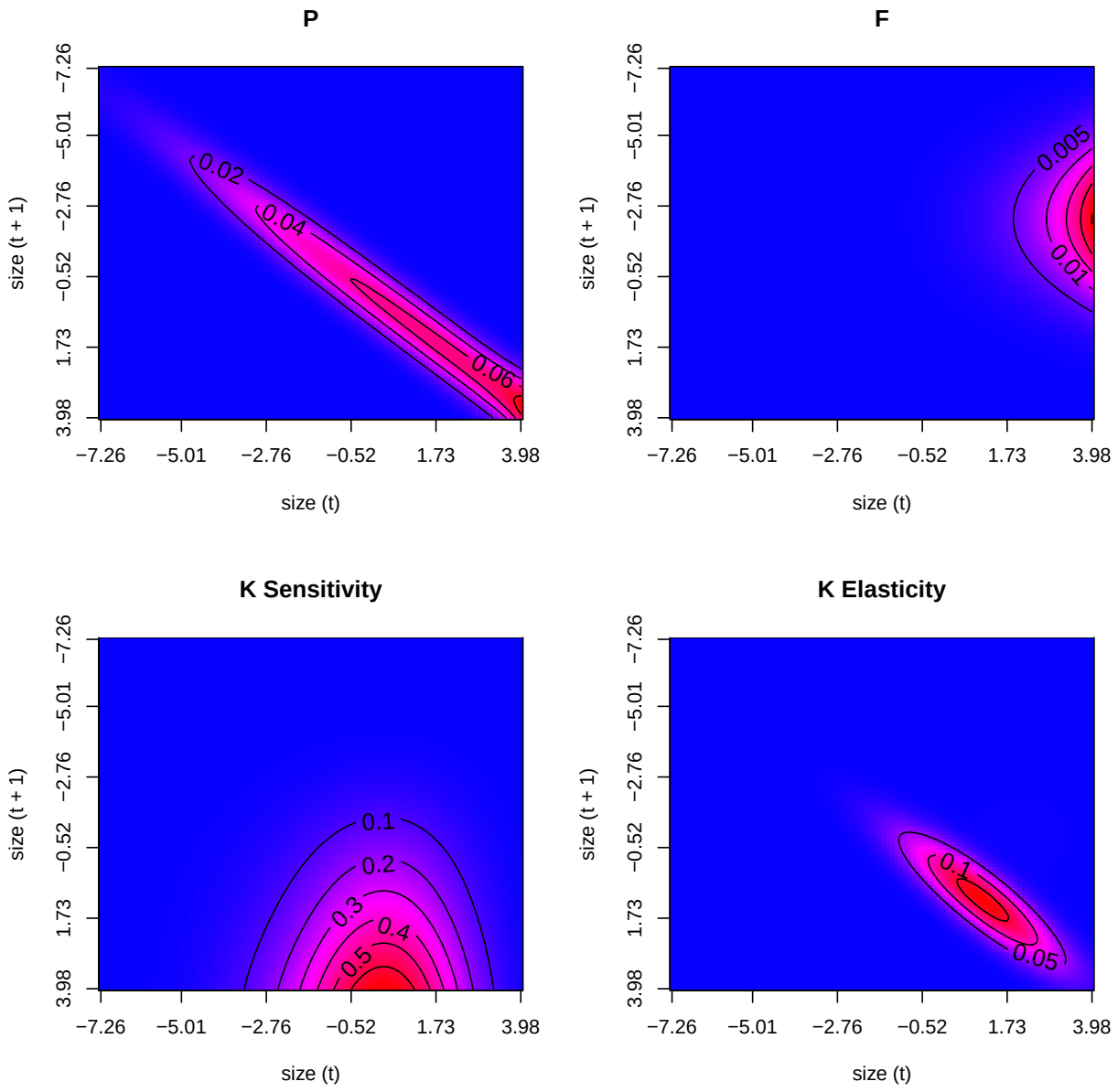
```
## [1] 27.79469
```

We may want to visualize our sub-kernels, iteration kernel, and the results of our sensitivity and elasticity analyses. We'll go through two options: one using the `graphics` package and one using the `ggplot2` package.

First, the `graphics` package.

```
lab_seq  <- round(seq(L, U, length.out = 6), 2)
tick_seq <- c(1, 20, 40, 60, 80, 100)
```

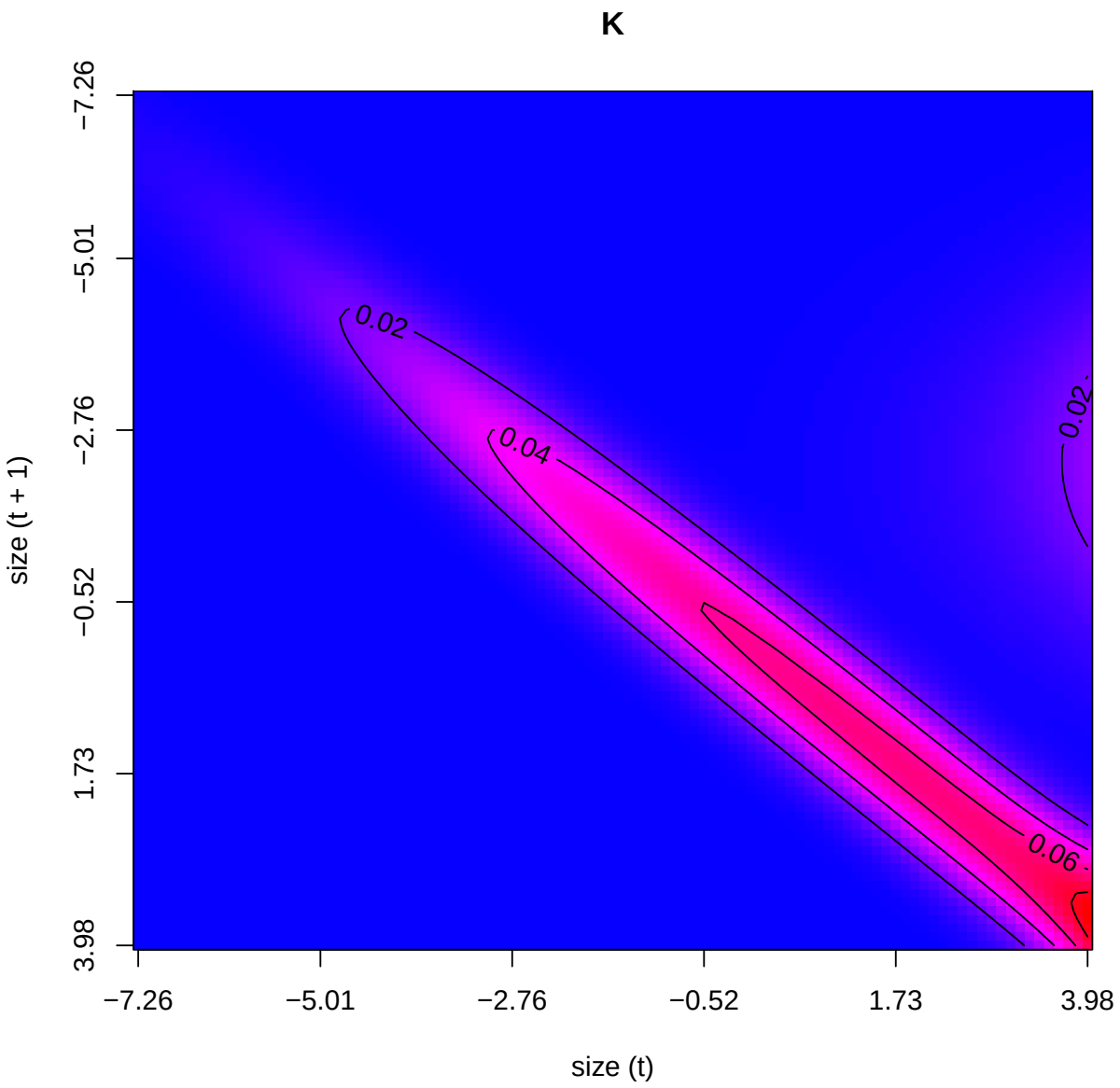
```
par(mfrow = c(2, 2))

# Sub-kernels - ipmr contains plot methods for sub-kernels

plot(carpobrotus_ipm$sub_kernels$P,
     do_contour = TRUE,
     main       = "P",
     xlab       = "size (t)",
     ylab       = "size (t + 1)",
     yaxt       = "none",
     xaxt       = "none")
axis(1, at = tick_seq, labels = as.character(lab_seq))
axis(2, at = tick_seq, labels = as.character(lab_seq))

plot(carpobrotus_ipm$sub_kernels$F,
     do_contour = TRUE,
     main       = "F",
     xlab       = "size (t)",
     ylab       = "size (t + 1)",
     yaxt       = "none",
     xaxt       = "none")
axis(1, at = tick_seq, labels = as.character(lab_seq))
axis(2, at = tick_seq, labels = as.character(lab_seq))



# Sensitivity and elasticity

class(sens_mat) <- c("ipmr_matrix", class(sens_mat))
class(elas_mat) <- c("ipmr_matrix", class(elas_mat))

plot(sens_mat,
     do_contour = TRUE,
     main       = "K Sensitivity",
     xlab       = "size (t)",
     ylab       = "size (t + 1)",
     yaxt       = "none",
     xaxt       = "none")
axis(1, at = tick_seq, labels = as.character(lab_seq))
axis(2, at = tick_seq, labels = as.character(lab_seq))


plot(elas_mat,
     do_contour = TRUE,
     main       = "K Elasticity",
     xlab       = "size (t)",
     ylab       = "size (t + 1)",
     yaxt       = "none",
     xaxt       = "none")
axis(1, at = tick_seq, labels = as.character(lab_seq))
axis(2, at = tick_seq, labels = as.character(lab_seq))
```




If we want to plot the iteration kernel, we can use `ipmr`'s `make_iter_kernel()` function to create one, and then the `plot()` method to plot that as well.

```
par(mfrow = c(1, 1))
K <- make_iter_kernel(carpobrotus_ipm)

plot(K$mega_matrix,
     do_contour = TRUE,
     main       = "K",
     xlab       = "size (t)",
     ylab       = "size (t + 1)",
     yaxt       = "none",
     xaxt       = "none")
axis(1, at = tick_seq, labels = as.character(lab_seq))
```

```
axis(2, at = tick_seq, labels = as.character(lab_seq))
```



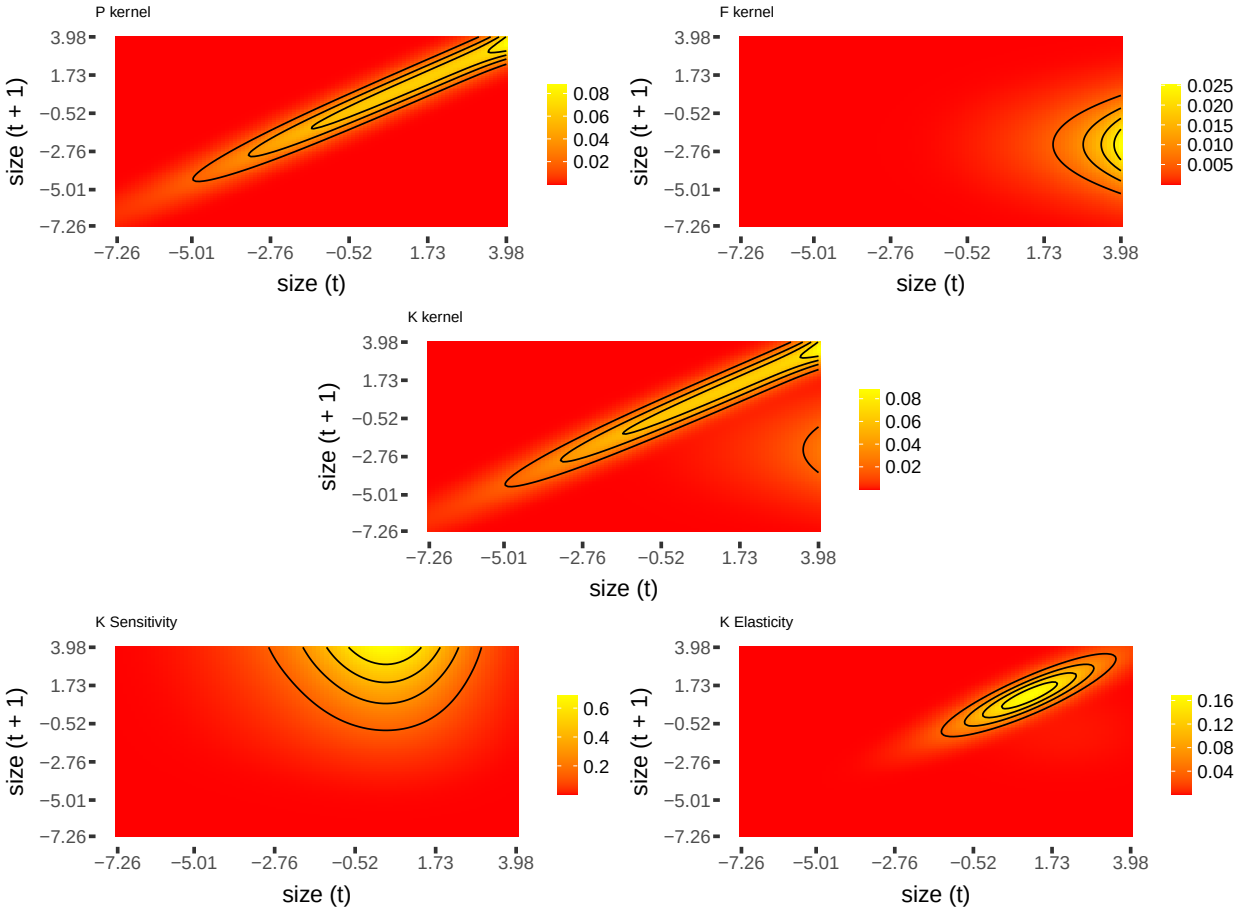

Now, for the `ggplot2` version. First, we create a long format of the matrix using `ipmr`'s `ipm_to_df` function. `ipm_to_df` can handle either bare matrices, or objects produced by `make_ipm`. The latter case is useful for plotting kernels directly using `ggplot2`. Once we've generated the long format sensitivity and elasticity matrices, we can use `geom_tile` and `geom_contour` to generate the `ggplots`, and `grid.arrange` from the `gridExtra` package to put them side by side.

```
library(ggplot2)
library(gridExtra)

p_df    <- ipm_to_df(carpobrotus_ipm$sub_kernels$P)
f_df    <- ipm_to_df(carpobrotus_ipm$sub_kernels$F)
k_df    <- ipm_to_df(K$mega_matrix)
```

```r
sens_df <- ipm_to_df(sens_mat)
elas_df <- ipm_to_df(elas_mat)

# Create a default theme for our plots

def_theme <- theme(
  panel.background  = element_blank(),
  axis.text         = element_text(size = 16),
  axis.ticks        = element_line(size = 1.5),
  axis.ticks.length = unit(0.08, "in"),
  axis.title.x      = element_text(
    size   = 20,
    margin = margin(
      t = 10,
      r = 0,
      l = 0,
      b = 2
    )
  ),
  axis.title.y = element_text(
    size   = 20,
    margin = margin(
      t = 0,
      r = 10,
      l = 2,
      b = 0
    )
  ),
  legend.text = element_text(size = 16)
)

p_plt <- ggplot(p_df) +
  geom_tile(aes(x    = t,
                y    = t_1,
                fill = value)) +
  geom_contour(aes(x = t,
                   y = t_1,
                   z = value),
               color = "black",
               size = 0.7,
               bins = 5) +
  scale_fill_gradient("Value",
                      low = "red",
                      high = "yellow") +
  scale_x_continuous(name = "size (t)",
                     labels = lab_seq,
                     breaks = tick_seq) +
  scale_y_continuous(name = "size (t + 1)",
                     labels = lab_seq,
                     breaks = tick_seq) +
  def_theme +
  theme(legend.title = element_blank()) +
  ggtitle("P kernel")
```

```
f_plt <- ggplot(f_df) +
  geom_tile(aes(x    = t,
                y    = t_1,
                fill = value)) +
  geom_contour(aes(x = t,
                   y = t_1,
                   z = value),
               color = "black",
               size = 0.7,
               bins = 5) +
  scale_fill_gradient("Value",
                      low = "red",
                      high = "yellow") +
  scale_x_continuous(name = "size (t)",
                     labels = lab_seq,
                     breaks = tick_seq) +
  scale_y_continuous(name = "size (t + 1)",
                     labels = lab_seq,
                     breaks = tick_seq) +
  def_theme +
  theme(legend.title = element_blank()) +
  ggtitle("F kernel")

k_plt <- ggplot(k_df) +
  geom_tile(aes(x    = t,
                y    = t_1,
                fill = value)) +
  geom_contour(aes(x = t,
                   y = t_1,
                   z = value),
               color = "black",
               size = 0.7,
               bins = 5) +
  scale_fill_gradient("Value",
                      low = "red",
                      high = "yellow") +
  scale_x_continuous(name = "size (t)",
                     labels = lab_seq,
                     breaks = tick_seq) +
  scale_y_continuous(name = "size (t + 1)",
                     labels = lab_seq,
                     breaks = tick_seq) +
  def_theme +
  theme(legend.title = element_blank()) +
  ggtitle("K kernel")

sens_plt <- ggplot(sens_df) +
  geom_tile(aes(x    = t,
                y    = t_1,
                fill = value)) +
  geom_contour(aes(x = t,
                   y = t_1,
                   z = value),
```

```
                color = "black",
                size = 0.7,
                bins = 5) +
  scale_fill_gradient("Value",
                      low = "red",
                      high = "yellow") +
  scale_x_continuous(name = "size (t)",
                     labels = lab_seq,
                     breaks = tick_seq) +
  scale_y_continuous(name = "size (t + 1)",
                     labels = lab_seq,
                     breaks = tick_seq) +
  def_theme +
  theme(legend.title = element_blank()) +
  ggtitle("K Sensitivity")

elas_plt <- ggplot(elas_df) +
  geom_tile(aes(x    = t,
                y    = t_1,
                fill = value)) +
  geom_contour(aes(x = t,
                   y = t_1,
                   z = value),
                color = "black",
                size = 0.7,
                bins = 5) +
  scale_fill_gradient("Value",
                      low = "red",
                      high = "yellow") +
  scale_x_continuous(name = "size (t)",
                     labels = lab_seq,
                     breaks = tick_seq) +
  scale_y_continuous(name = "size (t + 1)",
                     labels = lab_seq,
                     breaks = tick_seq) +
  def_theme +
  theme(legend.title = element_blank()) +
  ggtitle("K Elasticity")


grid.arrange(
  p_plt,    f_plt, k_plt,
  sens_plt, elas_plt,
             layout_matrix = matrix(c(1, 1, 2, 2,
                                      NA, 3, 3, NA,
                                      4, 4, 5, 5),
                                    nrow = 3,
                                    byrow = TRUE))
```


P kernel
size (t + 1)
size (t)
3.98
1.73
−0.52
−2.76
−5.01
−7.26
0.08
0.06
0.04
0.02
F kernel
0.025
0.020
0.015
0.010
0.005
K kernel


K Sensitivity
K Elasticity
0.6
0.4
0.2
0.16
0.12
0.08
0.04

# Appendix 2: `ipmr` Case Study 2

# Case Study 2: Ellner, Childs & Rees 2016

## A more complicated, age and size structured model

Many life cycles cannot be described by a single, continuous state variable. For example, some plants are best modeled using height or diameter at breast height (DBH) as a state variable, and may also form a seed bank. Seeds in the seed bank can't have a value for height or DBH, but may lose their viability as they age. Thus, we require a discrete state to capture the dynamics of the seed bank. Models that include discrete states and/or multiple continuous state variables are *general IPMs*.

Many species exhibit age-dependent demography. Age may interact with a measure of size, for example body mass, resulting in neither single variable reliably predicting demography on its own. Data sets for which individuals are cross-classified by age and size represent an interesting opportunity for demographic research.

This case study will use an age and size structured population to illustrate how to implement general IPMs in `ipmr`. We will use an age and size structured model from Ellner, Childs, & Rees (2016) to explore how to work with general IPMs in `ipmr`. The data are from a long term study of Soay sheep (*Ovis aries*) on St. Kilda. The population has been studied in detail since 1985 (Clutton-Brock & Pemberton 2004). Individuals are caught, weighed, and tagged shortly after birth, and re-weighed each subsequent year. Maternity can be inferred from field observations, so we can also model the link between parental state and offspring state. More detailed methods are provided in Clutton-Brock & Pemberton (2004) and Childs et al. 2011.

In addition to computing the per-capita growth rate ($\lambda$) and right and left eigenvectors ($w_a(z)$ and $v_a(z)$, respectively), we will also show how to compute age specific survival and fertility for these models.

With size denoted $z, z'$, age denoted $a$, and the maximum age an individual can have denoted $M$, the model can be written as:

1. $n_0(z', t+1) = \sum_{a=0}^{M} \int_L^U F_a(z', z) n_a(z, t) dz$
2. $n_a(z', t+1) = \int_L^U P_{a-1}(z', z) n_{a-1}(z, t) dz$ for $a = 1, 2, ..., M$

In this case, there is also an "greybeard" age class $M+1$ defined as $a \geq M+1$. We need to define one more equation to indicate the number of individuals in that age group.

3. $n_{M+1}(z', t+1) = \int_L^U [P_M(z', z) n_M(z, t) + P_{M+1}(z', z) n_{M+1}(z, t)] dz$

Below, $f_G$ and $f_B$ denote normal probability density functions. The sub-kernel $P_a(z', z)$ is comprised of the following functions:

4. $P_a(z', z) = s(z, a) * G(z', z, a)$
5. Survival: $Logit(s(z, a)) = \alpha_s + \beta_{s,z} * z + \beta_{s,a} * a$
6. Growth: $G(z', z, a) = f_G(z', \mu_G(z, a), \sigma_G)$
7. Mean Growth: $\mu_G(z, a) = \alpha_G + \beta_{G,z} * z + \beta_{G,a} * a$

and the sub-kernel $F_a(z', z)$ is comprised of the following functions:

8. $F_0 = 0$
9. $F_a = s(z, a) * p_b(z, a) * p_r(a) * B(z', z) * 0.5$

This model only follows females, and we assume that the population is half female. Thus, we multiply the $F_a$ kernel by 0.5. If needed, we could adjust the sex ratio based on observed data, and update this multiplication accordingly.

10. Probability of reproducing: $Logit(p_b(z,a)) = \alpha_{p_b} + \beta_{p_b,z} * z + \beta_{p_b,a} * a$
11. Probability of recruiting: $Logit(p_r(a)) = \alpha_{p_r} + \beta_{p_r,a} * a$
12. Recruit size distribution: $B(z',z) = f_B(z', \mu_B(z), \sigma_B)$
13. Mean recruit size: $\mu_B(z) = \alpha_B + \beta_{B,z} * z$

Equations 5-7 and 10-13 are parameterized from regression models. The parameter values are taken from Ellner, Childs & Rees, Chapter 6 (2016). These can be found here. In the code from the book, these parameters were used to simulate an individual based model (IBM) to generate a data set. The data were then used to fit regression models and an age×size IPM. We are going to skip the simulation and regression model fitting steps steps and just use the "true" parameter estimates to generate the IPM.

## Model Code

First, we will define all the model parameters and a function for the $F_a$ kernels.

```
library(ipmr)
```

```
## Warning: package 'ipmr' was built under R version 4.2.3
```

```
## Welcome to `ipmr`! `browseVignettes('ipmr')` to get started.
```

```
# Set parameter values and names

param_list <- list(
  ## Survival
  surv_int  = -1.70e+1,
  surv_z    =  6.68e+0,
  surv_a    = -3.34e-1,
  ## growth
  grow_int  =  1.27e+0,
  grow_z    =  6.12e-1,
  grow_a    = -7.24e-3,
  grow_sd   =  7.87e-2,
  ## reproduce or not
  repr_int  = -7.88e+0,
  repr_z    =  3.11e+0,
  repr_a    = -7.80e-2,
  ## recruit or not
  recr_int  =  1.11e+0,
  recr_a    =  1.84e-1,
  ## recruit size
  rcsz_int  =  3.62e-1,
  rcsz_z    =  7.09e-1,
  rcsz_sd   =  1.59e-1
)

# define a custom function to handle the F kernels. We could write a rather
# verbose if(age == 0) {0} else {other_math} in the define_kernel(), but that
# might look ugly. Note that we CANNOT use ifelse(), as its output is the same
# same length as its input (in this case, it would return 1 number, not 10000
```

```
# numbers).

r_fun <- function(age, s_age, pb_age, pr_age, recr) {

  if(age == 0) return(0)

  s_age * pb_age * pr_age * recr * 0.5

}
```

Next, we set up the $P_a$ kernels (Equations 4-7 above). Because this is a general, deterministic IPM, we use `init_ipm(sim_gen = "general", di_dd = "di", det_stoch = "det", uses_age = TRUE)`. We set `uses_age = TRUE` to indicate that our model has age structure as well as size structure. There are 3 key things to note:

1. the use of the suffix `_age` appended to the names of the `"P_age"` kernel and the `mu_g_age` variable.
2. the value `age` used in the vital rate expressions.
3. the list in the `age_indices` argument.

The values in the `age_indices` list will automatically get substituted in for `"age"` each time it appears in the vital rate expressions and kernels. We add a second variable to this list, `"max_age"`, to indicate that we have a "greybeard" class. If we wanted our model to kill all individuals above age 20, we would simply omit the `"max_age"` slot in the `age_indices` list.

This single call to `define_kernel()` will result in 22 actual kernels, one for each value of `age` from 0-21. For general IPMs that are not age-structured, we would use `uses_par_sets` and `par_set_indices` in the same way we're using `age` below.

The `plogis` function is part of the `stats` package in *R*, and performs the inverse logit transformation.

```
age_size_ipm <- init_ipm(sim_gen = "general",
                         di_dd = "di",
                         det_stoch = "det",
                         uses_age = TRUE) %>%
  define_kernel(
    name          = "P_age",
    family        = "CC",
    formula       = s_age * g_age * d_z,
    s_age         = plogis(surv_int + surv_z * z_1 + surv_a * age),
    g_age         = dnorm(z_2, mu_g_age, grow_sd),
    mu_g_age      = grow_int + grow_z * z_1 + grow_a * age,
    data_list     = param_list,
    states        = list(c("z")),
    uses_par_sets = FALSE,
    age_indices   = list(age = c(0:20), max_age = 21),
    evict_cor     = FALSE
  )
```

The $F_a$ kernel (equations 8-13) will follow a similar pattern - we append a suffix to the `name` paramter, and then make sure that our functions also include `_age` suffixes and `age` values where they need to appear.

```
age_size_ipm <-   define_kernel(
  proto_ipm     = age_size_ipm,
  name          = "F_age",
  family        = "CC",
  formula       = r_fun(age, s_age, pb_age, pr_age, recr) * d_z,
```

```
  s_age         = plogis(surv_int + surv_z * z_1 + surv_a * age),
  pb_age        = plogis(repr_int + repr_z * z_1 + repr_a * age),
  pr_age        = plogis(recr_int + recr_a * age),
  recr          = dnorm(z_2, rcsz_mu, rcsz_sd),
  rcsz_mu       = rcsz_int + rcsz_z * z_1,
  data_list     = param_list,
  states        = list(c("z")),
  uses_par_sets = FALSE,
  age_indices   = list(age = c(0:20), max_age = 21),
  evict_cor     = FALSE
)
```

Once we've defined the $P_a$ and $F_a$ kernels, we need to define starting and ending states for each kernel. Age-size structured populations will look a little different from other models, as we need to ensure that all fecundity kernels produce `age-0` individuals, and all survival-growth kernels produce `age` individuals. We define the implementation arguments using `define_impl()`, and set each kernel's `state_start` to `"z_age"`. Because the fecundity kernel produces age-0 individuals, regardless of the starting age, its `state_end` is `"z_0"`.

```
age_size_ipm <-  define_impl(
  proto_ipm = age_size_ipm,
  make_impl_args_list(
    kernel_names = c("P_age", "F_age"),
    int_rule     = rep("midpoint", 2),
    state_start    = c("z_age", "z_age"),
    state_end      = c("z_age", "z_0")
  )
)
```

We define the domains using `define_domains()` in the same way we did for Case Study 1.

```
age_size_ipm <- age_size_ipm %>%
  define_domains(
    z = c(1.6, 3.7, 100)
  )
```

Our definition of the initial population state will look a little different though. We want to create 22 copies of the initial population state, one for each age group in the model. We do this by appending `_age` to the `n_z` in the `...` part of `define_pop_state`. We'll also set `make_ipm(...,return_all_envs = TRUE)` so we can access the computed values for each vital rate function in the model.

```
age_size_ipm <-  define_pop_state(
  proto_ipm = age_size_ipm,
  n_z_age = rep(1/100, 100)
  ) %>%
  make_ipm(
    usr_funs = list(r_fun = r_fun),
    iterate  = TRUE,
    iterations = 100,
    return_all_envs = TRUE
  )
```

## Basic analysis

We see that the population is projected to grow by about 1.5% each year. As in Case Study 1, we can check for convergence using the `is_conv_to_asymptotic()` function.

```
lamb <- lambda(age_size_ipm)
lamb
```

```
##   lambda
## 1.014833
```

```
is_conv_to_asymptotic(age_size_ipm)
```

```
## lambda
##   TRUE
```

For some analyses, we may want to get the actual vital rate function values, rather than the sub-kernels and/or iteration kernels. We can access those with `vital_rate_funs()`. Note that right now, this function always returns a full bivariate form of the vital rate function (i.e. for survival, it returns a $n$ X $n$ kernel, rather than a 1 x $n$ vector, where $n$ is the number of meshpoints). It is also important to note that these functions are **not yet discretized**, and so need to be treated as such (i.e. any vital rate with a probability density function will contain probability densities, not probabilities).

```
vr_funs <- vital_rate_funs(age_size_ipm)

# Age 0 survival and growth vital rate functions

vr_funs$P_0
```

```
## s_0 (not yet discretized): A 100 x 100 kernel with minimum value: 0.0019 and maximum value: 0.9995
## g_0 (not yet discretized): A 100 x 100 kernel with minimum value: 0 and maximum value: 5.0691
## mu_g_0 (not yet discretized): A 100 x 100 kernel with minimum value: 2.2556 and maximum value: 3.528
```

```
# Age 12 fecundity functions

vr_funs$F_12
```

```
## s_12 (not yet discretized): A 100 x 100 kernel with minimum value: 0 and maximum value: 0.9744
## pb_12 (not yet discretized): A 100 x 100 kernel with minimum value: 0.0217 and maximum value: 0.9345
## pr_12 (not yet discretized): A 100 x 100 kernel with minimum value: 0.965 and maximum value: 0.965
## recr (not yet discretized): A 100 x 100 kernel with minimum value: 0 and maximum value: 2.5091
## rcsz_mu (not yet discretized): A 100 x 100 kernel with minimum value: 1.5038 and maximum value: 2.97
```

We can also update the model to use a new functional form for a vital rate expression. For example, we could add parent size dependence for the probability of recruiting function. This requires 3 steps: extract the `proto_ipm` object, set the new functional form, and update the parameter list. We have to wrap the assignment in `new_fun_form()` to prevent parsing errors.

```
new_proto <- age_size_ipm$proto_ipm

vital_rate_exprs(new_proto,
                 kernel = "F_age",
                 vital_rate = "pr_age") <-
  new_fun_form(plogis(recr_int + recr_z * z_1 + recr_a * age))

parameters(new_proto) <- list(recr_z = 0.05)

new_ipm <- make_ipm(new_proto,
                    return_all_envs = TRUE)
```

```
lambda(new_ipm)
```

```
##   lambda
## 1.017868
```

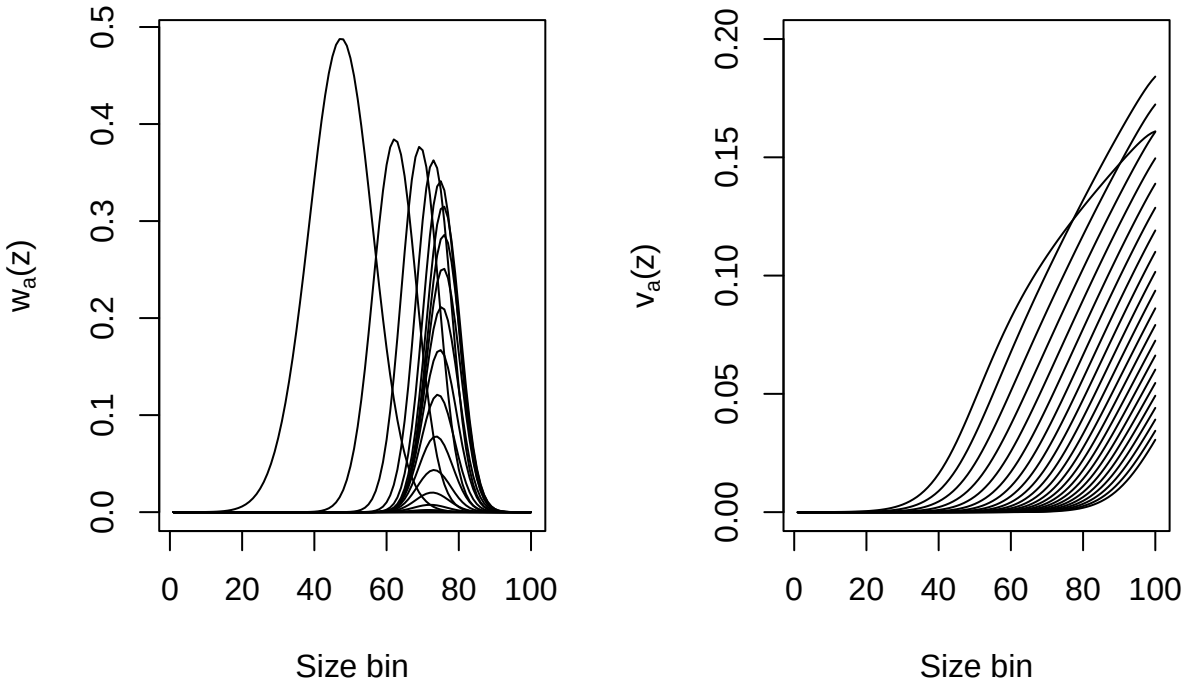
Next, we'll extract and visualize eigenvectors and compute age specific fertility and survival.

## Further analyses

The `right_ev` and `left_ev` functions also work for age × size models. We can use extract these, and plot them using a call to `lapply`. We will use the notation from Ellner, Childs, & Rees (2016) to denote the left and right eigenvectors ($v_a(z)$ and $w_a(z)$, respectively).

NB: we assign the `lapply` call to a value here because `lines` returns `NULL` invisibly, and this clogs up the console. You probably don't need to do this for interactive use. We'll also divide the $dz$ value back into each eigenvector so that they are continuous distributions, rather than discretized vectors.

```
d_z <- int_mesh(age_size_ipm)$d_z

stable_dists <- right_ev(age_size_ipm)

w_plot <- lapply(stable_dists, function(x, d_z) x / d_z,
         d_z = d_z)

repro_values <- left_ev(age_size_ipm)

v_plot <- lapply(repro_values, function(x, d_z) x / d_z,
                 d_z = d_z)


par(mfrow = c(1, 2))
plot(w_plot[[1]], type = 'l',
     ylab = expression(paste("w"[a],"(z)")),
     xlab = "Size bin")

x <- lapply(w_plot[2:22], function(x) lines(x))

plot(v_plot[[1]], type = 'l',
     ylab = expression(paste("v"[a],"(z)")),
     xlab = "Size bin",
     ylim = c(0, 0.2))

x <- lapply(v_plot[2:22], function(x) lines(x))
```




Next, we'll compute age-specific survival ($\tilde{l}_a$/`l_a`) and fecundity ($\tilde{f}_a$/`f_a`) values. These are defined as follows:

$$\tilde{l}_a = eP^a c$$

$$\tilde{f}_a = (eFP^a c)/l_a$$

where $c$ is some distribution of newborns.

We'll initialize a cohort using the stable size distribution for age-0 individuals that we obtained above. Next, we'll iterate them through our model for 100 years, and see who's left, and how much they reproduced.

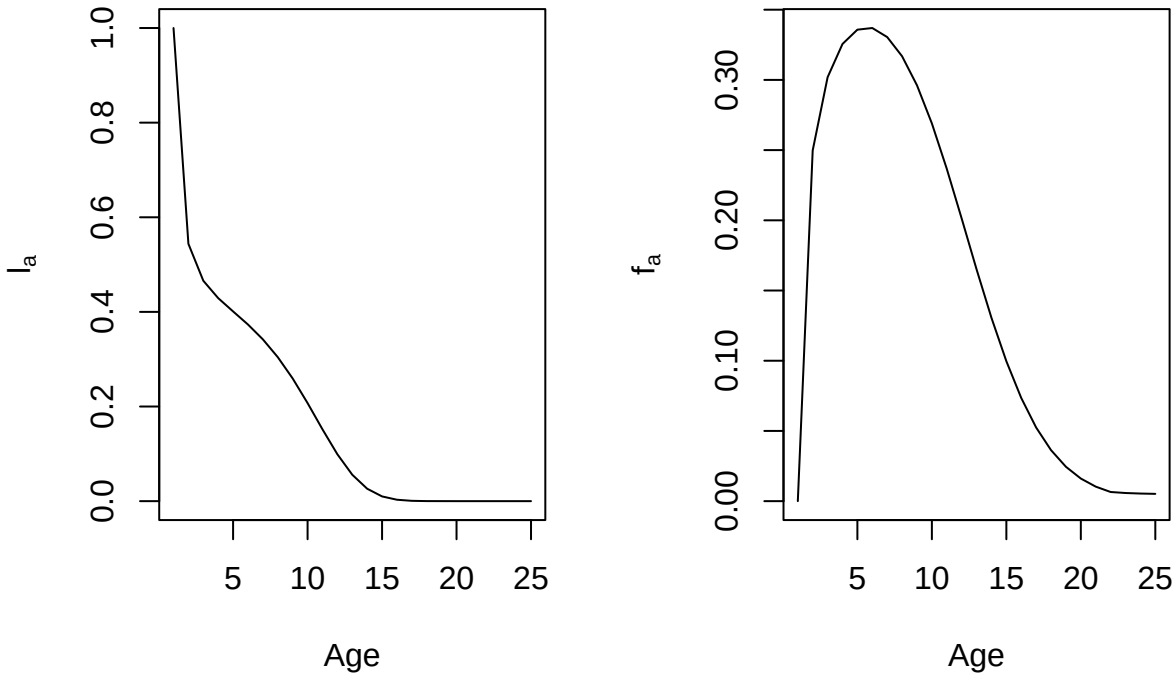
NB: do not try to use this method for computing $R_0$ - it will lead to incorrect results because in this particular model, parental state affects initial offspring state. For more details, see Ellner, Childs & Rees (2016), Chapters 3 and 6.

A couple technical notes:

1. We are going to split out our P and F sub-kernels into separate lists so that indexing them is easier during the iteration process.
2. We need to set our `max_age` variable to 22 now, so that we don't accidentally introduce an "off-by-1" error when we index the sub-kernels in our IPM object.
3. We use an identity matrix to compute the initial value of `l_a`, because by definition all age-0 individuals must survive to age-0.

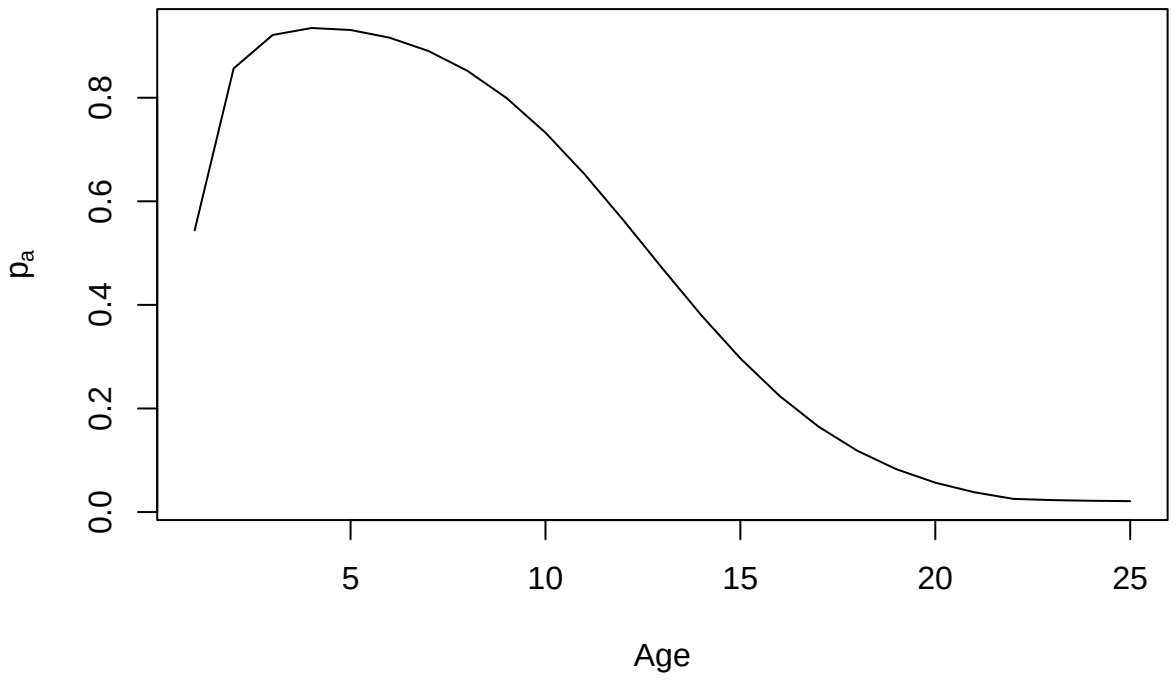
```
# Initialize a cohort and some vectors to hold our quantities of interest.

init_pop <- stable_dists[[1]] / sum(stable_dists[[1]])
```

```r
n_yrs    <- 100L

l_a <- f_a <- numeric(n_yrs)

P_kerns <- age_size_ipm$sub_kernels[grepl("P", names(age_size_ipm$sub_kernels))]
F_kerns <- age_size_ipm$sub_kernels[grepl("F", names(age_size_ipm$sub_kernels))]

# We have to bump max_age because R indexes from 1, and our minimum age is 0.

max_age <- 22

P_a <- diag(length(init_pop))

for(yr in seq_len(n_yrs)) {

  # When we start, we want to use age-specific kernels until we reach max_age.
  # after that, all survivors have entered the "greybeard" class.

  if(yr < max_age) {

    P_now <- P_kerns[[yr]]
    F_now <- F_kerns[[yr]]

  } else {

    P_now <- P_kerns[[max_age]]
    F_now <- F_kerns[[max_age]]

  }

  l_a[yr] <- sum(colSums(P_a) * init_pop)
  f_a[yr] <- sum(colSums(F_now %*% P_a) * init_pop)

  P_a <- P_now %*% P_a
}

f_a <- f_a / l_a

# Looks like most are dead at after 25 years, so we'll restrict our
# plot range to that time span

par(mfrow = c(1, 2))

plot(l_a[1:25], type = 'l',
     ylab = expression(paste("l"[a])),
     xlab = "Age")
plot(f_a[1:25], type = 'l',
     ylab = expression(paste("f"[a])),
     xlab = "Age")
```




We can also calculate the age-specific survival probability $p_a$ as $\frac{l_{a+1}}{l_a}$. We'll restrict our calculations to the first 25 years of life, as we'll see that almost no sheep live longer than that.

```
p_a <- l_a[2:26] / l_a[1:25]

plot(p_a, type = 'l',
     ylab = expression(paste("p"[a])),
     xlab = "Age")
```




## Citations

Childs DZ, Coulson T, Pemberton JM, Clutton-Brock TH, & Rees M (2011). Predicting trait values and measuring selection in complex life histories: reproductive allocation decisions in Soay sheep. Ecology Letters 14: 985-992.

Clutton-Brock TH & Pemberton JM (2004). Soay Sheep: Dynamics and Selection in an Island Population. Cambridge University Press, Cambridge.

# Appendix 3: Supplementary Information for Chapter 2

**Figure S3.1**

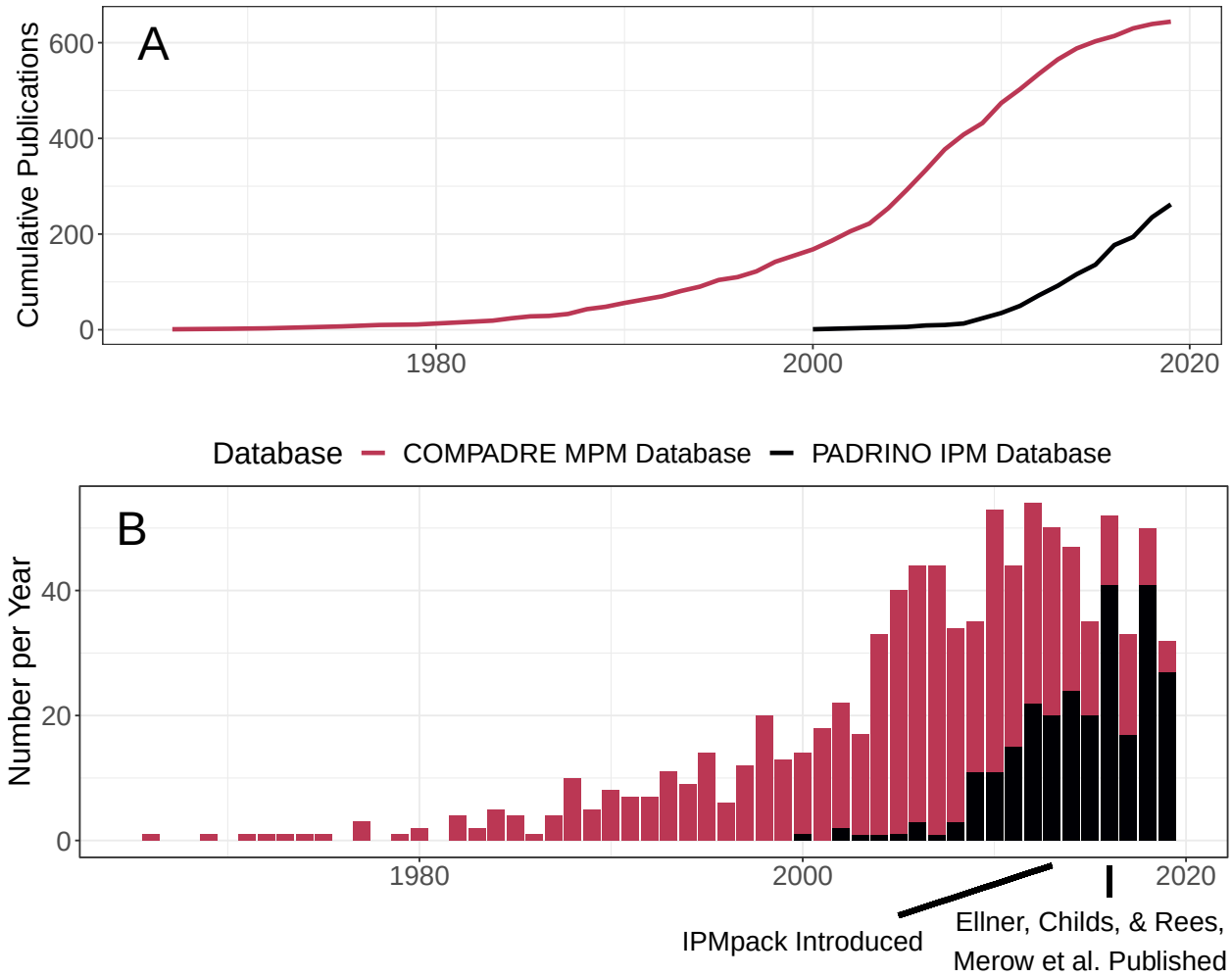



Figure S3.1: The usage of integral projection models (IPMs) has increased rapidly since their introduction. Cumulative number of publications using matrix projection models (MPMs, red) and IPMs (black) (A) and number of publications per year for each type of model (B). IPMs have been adopted rapidly since their introduction in 2000. Unfortunately, software packages to assist with their implementation have not kept pace with their theoretical advancements and applications to ever more complex demographic data.

# Appendix 4: PADRINO Case Study 1

# Stand-alone analyses with PADRINO and Rpadrino

## PADRINO and Rpadrino

We have created *Rpadrino* to streamline the process of interacting with PADRINO from *R*. The goal of *Rpadrino* is data management and model construction - not necessarily to do analyses for you. Thus, there is still some minimum amount of programming knowledge required to use it. It will also be helpful to understand how *ipmr*, the engine that powers model reconstruction, creates model objects and the things that it returns. *ipmr* is extensively documented here, and reading at least the introduction to that will certainly help understand the code that follows here. Eventually, we plan to create the *ipmtools* package, which will house functions designed to work with the IPMs stored in PADRINO to conduct more extensive analyses (*e.g.* perturbations, LTREs, life history traits).

## Usage

This case study makes use of *dplyr* to help with data transformation. If you do not already have it, install it (and *Rpadrino*) with:

```
install.packages(c("dplyr", "Rpadrino"))
```

We will show how to compute sensitivity and elasticity for simple models, and then derive some demographic quantities from models housed in PADRINO. The first step is to identify models that have the information we want. Sensitivity and elasticity computations will make use of functions contained in *ipmr*, and we will define a couple of our own to help tie it all together.

After perturbation analyses, we will compute the mean lifetime output of recruits as a function of initial size $z_0$, $\bar{r}(z_0)$. This is defined as $\bar{r}(z_0) = eFN$, where $F$ represents a the fecundity kernel, and $N$ is the fundamental operator. The fundamental operator can be thought of as the expected amount of time spent in any state $z'$ prior to death given an initial state $z_0$ (Caswell 2001), and is computed as $N = (I - P)^{-1}$ (where $P$ is the survival/growth kernel, and $I$ is an identity operator such that $IP = PI = P$).

Finally, we will compute mean size at death ($\bar{\omega}(z_0) = (\mathbf{i} \circ (1 - s))N$), and the size at death kernel ($\Omega(z', z_0) = (1 - s(z'))N(z', z_0)$. Thus, we need models that contain information on survival and growth, and sexual reproduction. To keep things simpler, we will restrict ourselves to simple IPMs.

> NB: The above formulae are from Ellner, Childs, & Rees (2016), Table 3.3. Their derivations are described in detail in Chapter 3 of the book.

### Subsetting using Metadata and other tables

We can find simple IPMs in PADRINO using a combination of *dplyr* (Wickham et al. 2021) and *Rpadrino* code. *Rpadrino* provides the `pdb_subset()` function. `pdb_subset()` currently only takes `ipm_id`s that we want to keep. The functionality will get expanded, but it's surprisingly complicated to manage that in a user-friendly interface (see the PADRINO explorer app for additional help). This means that we have to work out which `ipm_id`s correspond to the models we want, and then pass those to `pdb_subset()`.

It is a good idea to consult the table guide so that you are familiar with table and variable names, and what information they provide. This case study will introduce some of the variables and tables, but it does not cover them all!

```
library(dplyr)
library(Rpadrino)
```

```
## Warning: package 'ipmr' was built under R version 4.2.3
```

```
pdb <- pdb_download(save = FALSE)

# Simple models only make use of 1 trait/state variable. Therefore, if a model
# has more than 1, it is, by definition, not a simple model. The code below
# calculates the number of traits per "ipm_id", and then filters out those
# that have more than 1 trait. The final piece with the square brackets makes sure
# the final result is a character vector containing only ipm_id's, rather than
# a data.frame

simple_mod_ind <- pdb$StateVariables %>%
  group_by(ipm_id) %>%
  summarise(N = n()) %>%
  filter(N < 2) %>%
  .[ , "ipm_id", drop = TRUE]

simple_pdb <- pdb_subset(pdb, simple_mod_ind)
```

We have quite a few to choose from! However, a number of these may be stochastic and/or density-dependent models. We will want to get rid of those too, as sensitivity analyses can be trickier and more time consuming for them. This process will use the `Metadata`, `EnvironmentalVariables`, and `ParSetIndices` tables to find those.

```
# The first piece of stoch_ind examines the EnvironmentalVariables table. This
# contains information on IPMs that include continuous environmental variation.
# Rpadrino treats these as stochastic by default, because  PADRINO almost
# always uses random number generators to sample the distributions of environmental
# values. Therefore, there is not really a way to sample these in a way that makes
# them deterministic while still preserving the published model.

stoch_ind <- unique(simple_pdb$EnvironmentalVariables$ipm_id)

# The second piece of stoch_ind examines the ParSetIndices table. This table
# describes discrete environmental variation. These models do not have to be
# stochastic, but they will make the analysis a bit more complicated, so we are
# going to drop those for now.

stoch_ind <- c(stoch_ind, unique(simple_pdb$ParSetIndices$ipm_id))

# The final piece of stoch_ind checks for density dependence. This information
# is stored in the 'has_dd' column of the Metadata table.

stoch_ind <- c(stoch_ind,
               unique(simple_pdb$Metadata$ipm_id[simple_pdb$Metadata$has_dd]))

det_pdb <- pdb_subset(simple_pdb, setdiff(simple_pdb$Metadata$ipm_id,
                                          stoch_ind))
```

For simple models in PADRINO, the kernels are pretty consistently named with respect to the broader IPM literature: `P` denotes survival and growth, `F` denotes sexual reproduction, and `C` denotes asexual reproduction. The `IpmKernels` table stores the names, functional forms of the kernels, as well as other information needed to implement them. We can use the kernel names column, `kernel_id`, to do a quick sanity check to make sure our subsetting produced only these kernels like so:

```
unique(det_pdb$IpmKernels$kernel_id)
```

```
## [1] "P" "F"
```

Great, all `P`s and `F`s! Finally, we are going to make sure we only have one IPM per species in our analysis. This is certainly not required for any analyses, just to keep things tractable for now. *Rpadrino*'s metadata access functions return the `ipm_id`'s as names of each value, so we can use `names()` to get the IDs we need.

```
keep_ind <- pdb_species_accepted(det_pdb) %>%
  .[!duplicated(.)] %>%
  names()

my_pdb <- pdb_subset(det_pdb, keep_ind)
```

## Re-building IPMs

Now that we have our data subsetted, we can start making IPMs. The first step is always to create `proto_ipm` objects. These are an intermediate step between the database and a set of usable kernels. Because *ipmr* also uses these as an intermediate step, we can combine models from PADRINO with ones that we create ourselves. There is an example of this in the second case study.

### Under the hood

If you're interested in how PADRINO actually stores IPMs, and specifically the expressions that comprise them, keep reading. If not, skip to the next heading.

Lets have a look at how PADRINO stores kernels, vital rates, and parameters. These are in the `IpmKernels`, `VitalRateExpr`, and `ParameterValuess` tables, respectively.

```
head(my_pdb$IpmKernels[ , 1:3])
```

```
##    ipm_id kernel_id                                                formula
## 43 aaaa34         P                                    P = s * g * d_lnsize
## 44 aaaa34         F                          F = r * fn * pE * d  * d_lnsize
## 47 aaaa36         P                                    P = s * g * d_lnsize
## 48 aaaa36         F                          F = r * fn * pE * d  * d_lnsize
## 92 aaa144         P                                      P = s * g * d_size
## 93 aaa144         F F = rep_p * es_p * sdl_s *n_infl * n_fl * n_seed * d_size
```

```
head(my_pdb$VitalRateExpr[ , 1:3])
```

```
##     ipm_id demographic_parameter                                formula
## 147 aaaa34              Survival s = 1/(1+exp(-(s_b + s_m * lnsize_1)))
## 148 aaaa34                Growth                g = Norm(g_mean, g_var)
## 149 aaaa34                Growth          g_mean = g_b + g_m * lnsize_1
## 150 aaaa34                Growth   g_var = sqrt(gv_b + gv_m * lnsize_1)
## 151 aaaa34             Fecundity r = 1/(1+exp(-(r_b + r_m * lnsize_1)))
## 152 aaaa34             Fecundity       fn = exp(fn_b + fn_m * lnsize_1)
```

```
head(my_pdb$ParameterValues)
```

```
##     ipm_id demographic_parameter state_variable parameter_name parameter_value
## 619 aaaa34              Survival         lnsize            s_b      -0.5612335
## 620 aaaa34              Survival         lnsize            s_m       0.4628431
## 621 aaaa34                Growth         lnsize            g_b       1.1088198
## 622 aaaa34                Growth         lnsize            g_m       0.5148672
## 623 aaaa34                Growth         lnsize           gv_b       0.9504887
## 624 aaaa34                Growth         lnsize           gv_m       0.0000000
```

We can see that the kernels and vital rate expressions are all defined symbolically, and the parameter values are stored elsewhere. This helps us reuse parameters that appear in multiple expressions without re-typing them, reducing the risk of errors. Additionally, it'll make it easier for us to modify parameter values, vital rate expressions, and kernel formulae if we want to. However, the syntax in the tables is probably not the easiest to work with directly. Therefore, *Rpadrino* provides the `pdb_make_proto_ipm()` function. This takes a `pdb` object and produces a list of `proto_ipm`s. In the chunk after this one, we will see that it translates the syntax in `IpmKernels` and `VitalRateExpr` into usable `R` code. There are additional options that we can pass to this, but we will ignore those for now, and just focus on creating and understanding what the outputs are.

### Creating the proto_ipm list

The following line generates a set of `proto_ipm`'s for the species in our subsetted database:

```
simple_det_list <- pdb_make_proto_ipm(my_pdb)
```

```
## 'ipm_id' aaa310 has the following notes that require your attention:
## aaa310: 'Geo and time info retrieved from COMPADRE (v.X.X.X.4)'

## 'ipm_id' aaa323 has the following notes that require your attention:
## aaa323: 'Simulated demographic data derived from Nicole J Ecol 2011'

## 'ipm_id' aaa326 has the following notes that require your attention:
## aaa326: 'Demographic data from Metcalf Funct Ecol 2006'

## 'ipm_id' aaa385 has the following notes that require your attention:
## aaa385: 'Same data as AAA385. State variable Height (Cm)'

## 'ipm_id' ddddd3 has the following notes that require your attention:
## ddddd3: 'Frankenstein IPM'

## 'ipm_id' ddddd4 has the following notes that require your attention:
## ddddd4: 'assumes mean surface temp of 10.34 °C, and constant survival probability of
## large pike'

## 'ipm_id' dddd24 has the following notes that require your attention:
## dddd24: 'Assumes no external recruitment'

## 'ipm_id' dddd26 has the following notes that require your attention:
## dddd26: '1 ipm digitized, additional ipms taking into account dispersal still
## possible to digitize'

## 'ipm_id' dddd30 has the following notes that require your attention:
## dddd30: 'Frankenstein IPM'

## 'ipm_id' dddd37 has the following notes that require your attention:
## dddd37: 'MS contains 2 det and 2 stoch IPMs, only 1 det inlcuded here'

## 'ipm_id' dddd39 has the following notes that require your attention:
## dddd39: 'Only deterministic model included here: assumes precipitation = 104mm'
```

```
## 'ipm_id' dddd40 has the following notes that require your attention:
## dddd40: 'DEB-IPM - these vital rates assume NO shrinking. 1 model digitized: assumes
## that scaled functional response E_Y = 0.65, with var(E_Y) = 0.1 - see paper for
## details'

## 'ipm_id' dddd41 has the following notes that require your attention:
## dddd41: 'DEB-IPM - these vital rates shrinking IS possible.  1 model digitized:
## assumes that scaled functional response E_Y = 0.65, with var(E_Y) = 0.1 - see paper
## for details'
```

First, we note that the building process threw out a few messages. The first is that the coordinates and duration information come from COMPADRE, not necessarily the original publication. This is not really alarming - COMPADRE is pretty trustworthy. The next few are related to demographic data sources and GPS location. "Frankenstein IPM" refers to a situation where some vital rates are measured directly from demographic data the authors collected, while other vital rates were retrieved from the literature (*i.e.* the object is cobbled together from disparate sources, Shelley 1818). Again, not necessarily alarming, though we'd want to know this info if our study question required that all vital rates come from one place (*e.g.* matching environmental conditions to demographic performance). The next few tell us that the publication actually contains more IPMs, but that our PADRINO digitization team hasn't finished entering all of them yet. And finally, there is a note about the assumptions contained by the model.

we will inspect a couple of the objects in this list to get a feel for what a `proto_ipm` contains:

```
simple_det_list
```

```
## This list of 'proto_ipm's contains the following species:
## Poa alsodes
## Poa sylvestris
## Aeonium haworthii
## Cotyledon orbiculata
## Aconitum noveboracense
## Dracocephalum austriacum
## Cirsium arvense
## Lonicera maackii
## Mimulus cardinalis
## Reynoutria japonica
## Carpobrotus spp
## Crocodylus niloticus
## Esox lucius
## Sisturus catenatus catenatus
## Ovis aries
## Oncorhynchus clarkii
## Tridacna maxima
## Gadus morhua
## Podarcis lilfordi
## Nerodia sipedon
## Ostrea edulis
## Dipsastraea favus
## Platygyra lamellina
## Ficedula hypoleuca
## Testudo graeca
## Manta alfredi
## Rhizoglyphus robini
##
## You can inspect each model by printing it individually.
```

```
simple_det_list$aaaa34
```

```
## A simple, density independent, deterministic proto_ipm with 2 kernels defined:
##  P, F
##
## Kernel formulae:
##
## P: s * g
## F: r * fn * pE * d
##
## Vital rates:
##
## s: 1/(1 + exp(-(s_b + s_m * lnsize_1)))
## g_mean: g_b + g_m * lnsize_1
## g_var: sqrt(gv_b + gv_m * lnsize_1)
## g: dnorm(lnsize_2, g_mean, g_var)
## r: 1/(1 + exp(-(r_b + r_m * lnsize_1)))
## fn: exp(fn_b + fn_m * lnsize_1)
## d: dexp(lnsize_2, 1/d_mean)
##
## Parameter names:
##
##  [1] "s_b"    "s_m"    "g_b"    "g_m"    "gv_b"   "gv_m"   "r_b"    "r_m"
##  [9] "fn_b"   "fn_m"   "t_r"    "d_mean" "pE"
##
## All parameters in vital rate expressions found in 'data_list':  TRUE
##
## Domains for state variables:
##
## lnsize: lower_bound = 0, upper_bound = 5, n_meshpoints = 500
##
## Population states defined:
##
## n_lnsize: Pre-defined population state.
##
## Internally generated model iteration procedure:
##
## n_lnsize_t_1: right_mult(kernel = P, vectr = n_lnsize_t) + right_mult(kernel = F,
##     vectr = n_lnsize_t)
```

```
simple_det_list$dddd30
```

```
## A simple, density independent, deterministic proto_ipm with 2 kernels defined:
##  P, F
##
## Kernel formulae:
##
## P: s * g
## F: s * r * pg * 0.5 * d
##
## Vital rates:
##
## s: 1/(1 + exp(-(aS + bS * svl_1 + cS * svl_1^2)))
## muG: svl_1 + (Linf - svl_1) * (1 - exp(-k * tg))
```

```
## g: dnorm(svl_2, muG, sigmaG)
## r: exp(aR + bR * svl_1)
## pg: ifelse(svl_1 < svlM, 0, 1)
## d: dnorm(svl_2, muD, sigmaD)
##
## Parameter names:
##
##  [1] "aS"     "bS"     "cS"     "Linf"   "k"      "tg"     "sigmaG" "aR"
##  [9] "bR"     "svlM"   "muD"    "sigmaD"
##
## All parameters in vital rate expressions found in 'data_list':  TRUE
##
## Domains for state variables:
##
## svl: lower_bound = 120, upper_bound = 1200, n_meshpoints = 1000
##
## Population states defined:
##
## n_svl: Pre-defined population state.
##
## Internally generated model iteration procedure:
##
## n_svl_t_1: right_mult(kernel = P, vectr = n_svl_t) + right_mult(kernel = F,
##     vectr = n_svl_t)
```

We can see that *Rpadrino* has translated PADRINO's syntax into a set of *R* expressions that correspond to the vital rate functions and sub-kernel functional forms, as well as checked that the model can be implemented with the parameter values that are present in PADRINO. Finally, it has generated the model iteration expression, which shows how the sub-kernels interact with each trait distribution at time $t$ to produce new trait distributions at time $t+1$. We can now build the actual IPM objects. We will also check for convergence to asymptotic dynamics using the `is_conv_to_asymptotic` function.

```
all_ipms   <- pdb_make_ipm(simple_det_list)

check_conv <- is_conv_to_asymptotic(all_ipms)
```

```
## The following IPMs did not converge: aaa310, aaa341, ccccc1, ddddd3, ddddd5,
## dddd10, dddd24, dddd26, dddd30, dddd33, dddd35, dddd36, dddd37, dddd39, dddd40,
## dddd41
```

```
check_conv
```

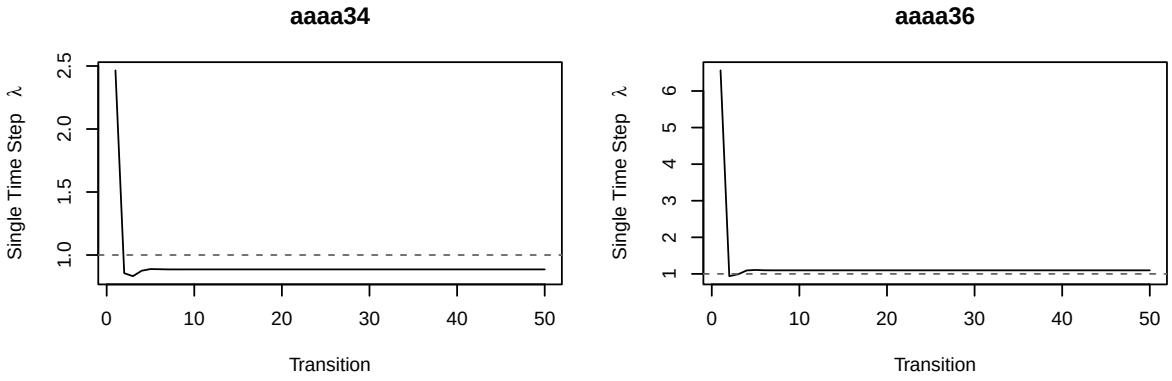
```
## [1] FALSE
```

We can see that a few of these need more than the default number of iterations to converge to asymptotic dynamics. All $\lambda$ values are computed via iteration, rather than computing eigenvalues. Since we need correct $\lambda$ values to compute elasticity, we will need to re-run those models until they converge (or at least come very close to convergence). `pdb_make_ipm()` contains the `addl_args` argument that tells the function how to deviate from the default behavior of `ipmr::make_ipm()`. It's accepts nested lists with the following format:

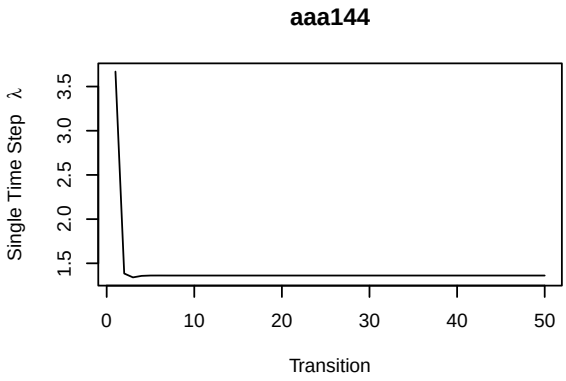
```
list(<ipm_id_1> = list(<make_ipm_arg_name_1> = <XXX>,
                       <make_ipm_arg_name_2> = <YYY>),
     <ipm_id_2> = list(<make_ipm_arg_name_1> = <XXX>,
                       <make_ipm_arg_name_5> = <ZZZ>))
```

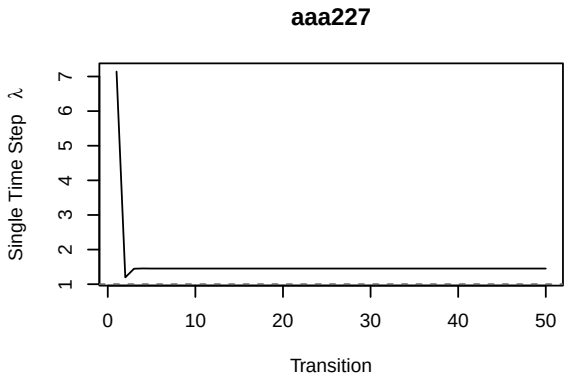
We replace the values in `<>` with the actual `ipm_id`s, argument names, and values we want them to have. We can do this many models a bit more concisely:

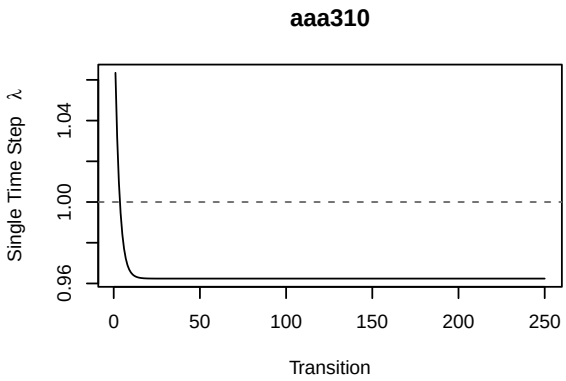
```r
# Create an empty list with names that correspond to ipm_id's that we want to add
# additional iterations for.

ind_conv <- c(paste0("aaa", c(310,341)),
              paste0("ccccc", 1),
              paste0("ddddd", c(3,5)),
              paste0("dddd", c(10, 24, 26, 30, 33, 35, 36, 37, 39, 40, 41)))

# Next, we set create an entry in each list with iterations = <some number>
# we will use 250 for this example. We need to set the names of the list to be
# the ipm_id's, so that pdb_make_ipm() knows which models to use the additional
# arguments with.

arg_list <- lapply(ind_conv,
                   function(x, n_iter) list(iterations = n_iter),
                   n_iter = 250) %>%
  setNames(ind_conv)

new_ipms <- pdb_make_ipm(simple_det_list, addl_args = arg_list)

# Check for convergence out to 5 digits. This should be close enough for what
# we want to do.

check_conv <- is_conv_to_asymptotic(new_ipms, tolerance = 1e-5)

# We can also plot the lambda time series using conv_plot methods for pdb_ipms.
# The last two models may not have converged
par(mfrow = c(4, 2))

conv_plot(new_ipms)
```

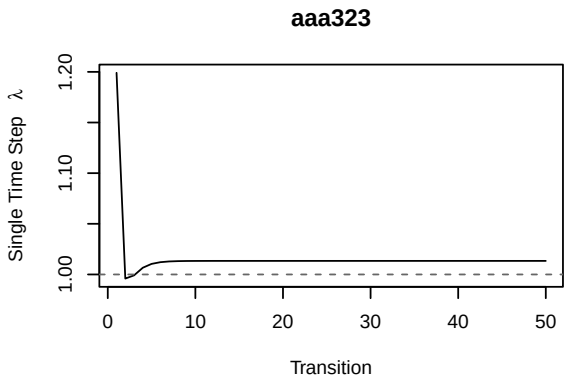

aaaa34
Single Time Step λ
1.0
1.5
2.0
2.5
0
10
20
30
40
50
Transition

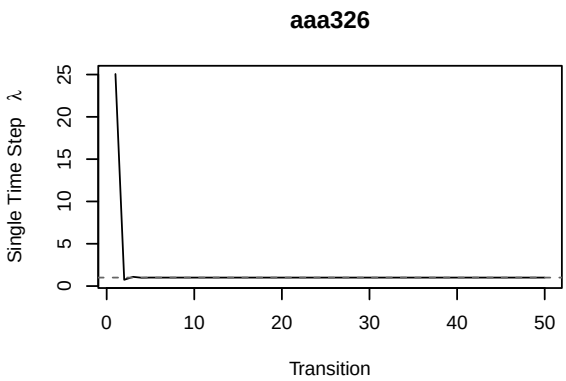

aaaa36
Single Time Step λ
1
2
3
4
5
6
0
10
20
30
40
50
Transition

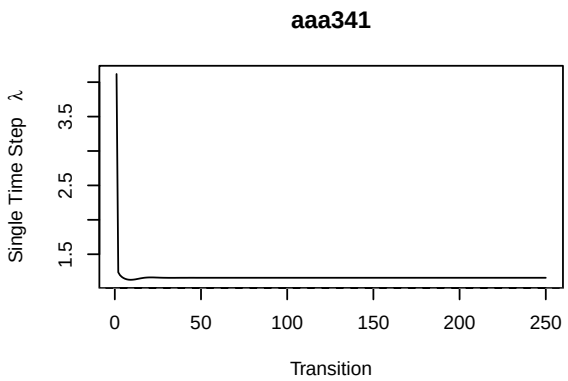

aaa144
Single Time Step λ
1.5
2.0
2.5
3.0
3.5
0
10
20
30
40
50
Transition

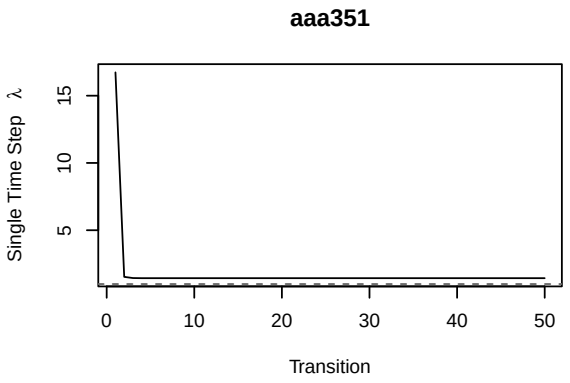

aaa227
Single Time Step λ
1
2
3
4
5
6
7
0
10
20
30
40
50
Transition

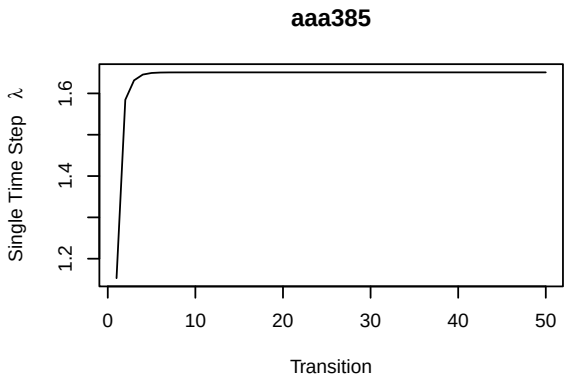

aaa310
Single Time Step λ
0.96
1.00
1.04
0
50
100
150
200
250
Transition

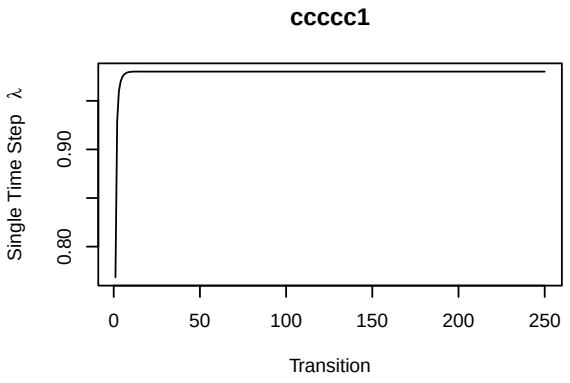

aaa323
Single Time Step λ
1.00
1.10
1.20
0
10
20
30
40
50
Transition

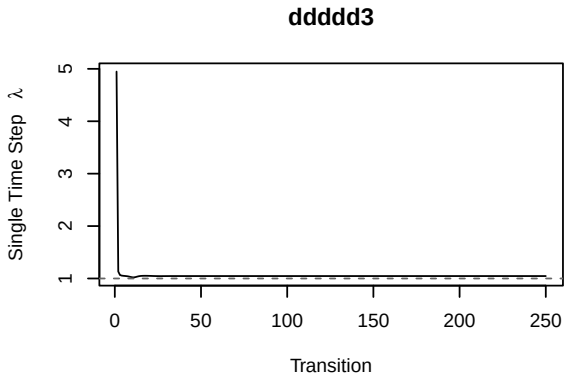

aaa326
Single Time Step λ
0
5
10
15
20
25
0
10
20
30
40
50
Transition

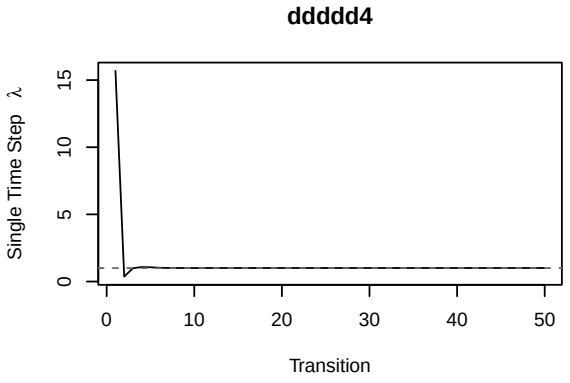

aaa341
Single Time Step λ
1.5
2.5
3.5
0
50
100
150
200
250
Transition

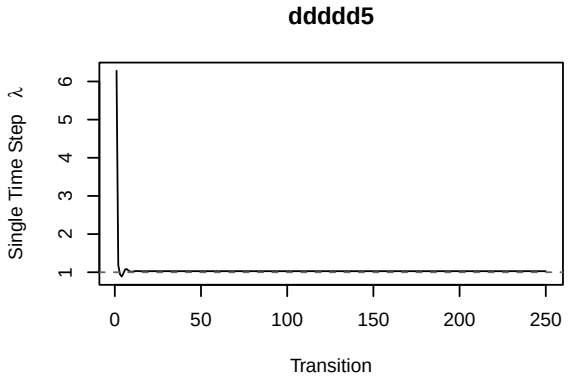

aaa351
Single Time Step λ
5
10
15
0
10
20
30
40
50
Transition

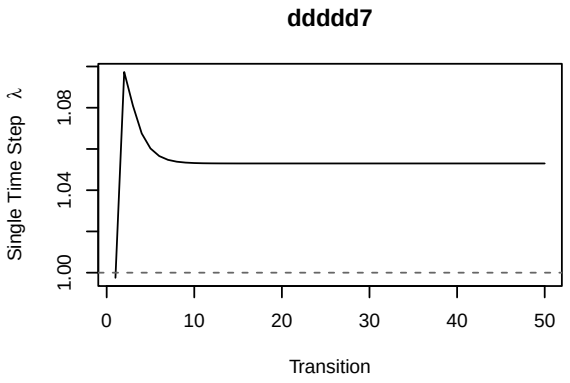

aaa385
Single Time Step λ
1.2
1.4
1.6
0
10
20
30
40
50
Transition

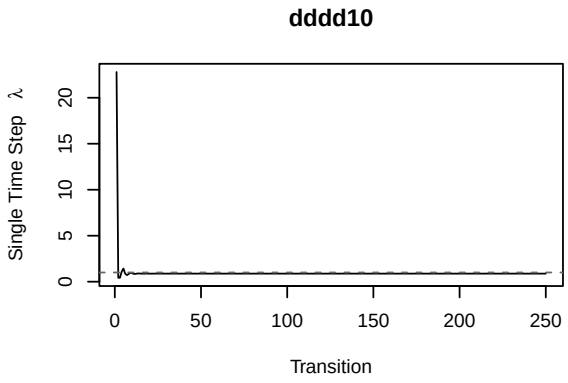

ccccc1
Single Time Step λ
0.80
0.90
0
50
100
150
200
250
Transition

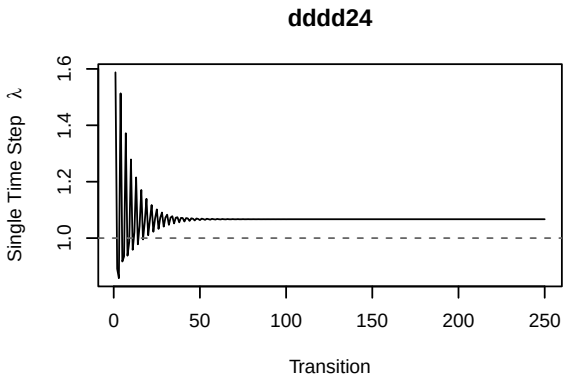

ddddd3
Single Time Step λ
1
2
3
4
5
0
50
100
150
200
250
Transition

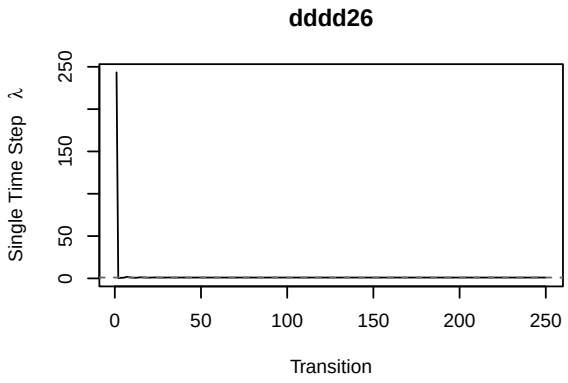

ddddd4
Single Time Step λ
0
5
10
15
0
10
20
30
40
50
Transition

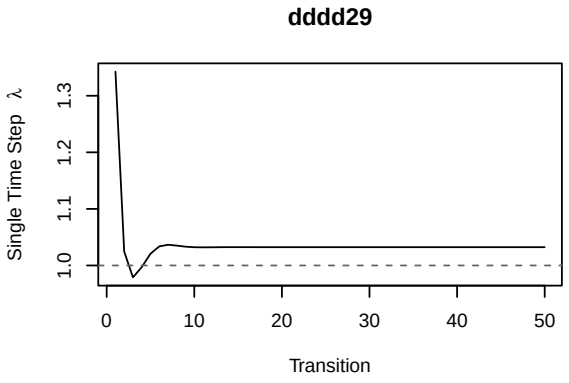

ddddd5
Single Time Step λ
1
2
3
4
5
6
0
50
100
150
200
250
Transition

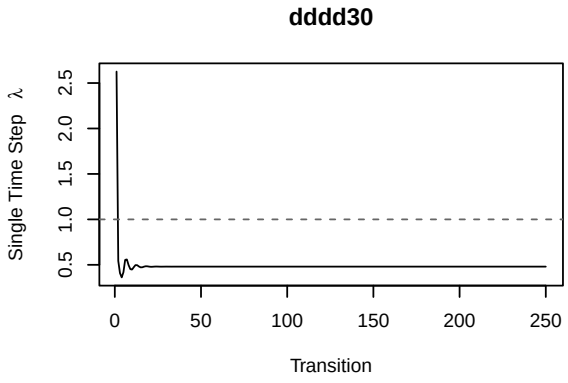

ddddd7
Single Time Step λ
1.00
1.04
1.08
0
10
20
30
40
50
Transition

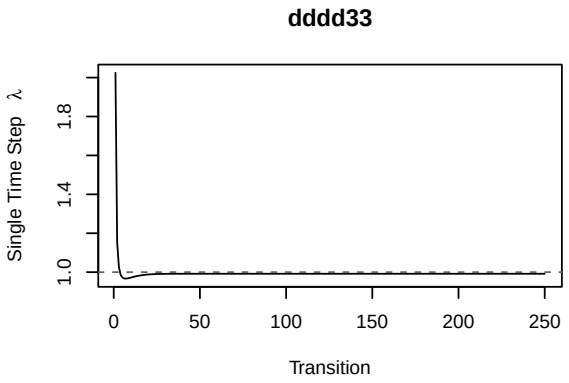

dddd10
Single Time Step λ
0
5
10
15
20
0
50
100
150
200
250
Transition

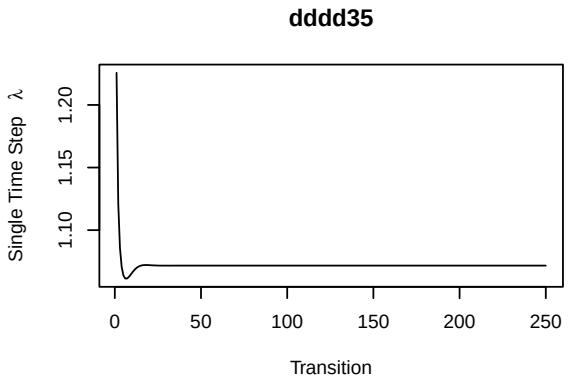

dddd24
Single Time Step λ
1.0
1.2
1.4
1.6
0
50
100
150
200
250
Transition

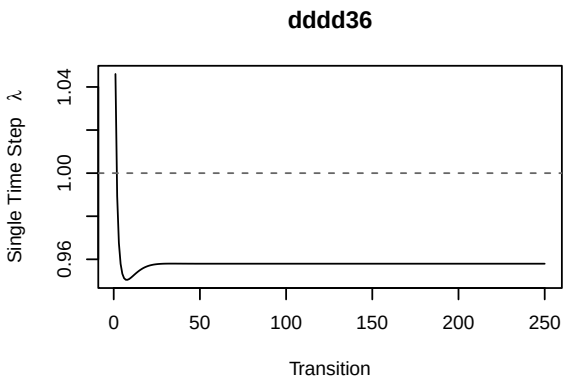

dddd26
Single Time Step λ
0
50
150
250
0
50
100
150
200
250
Transition

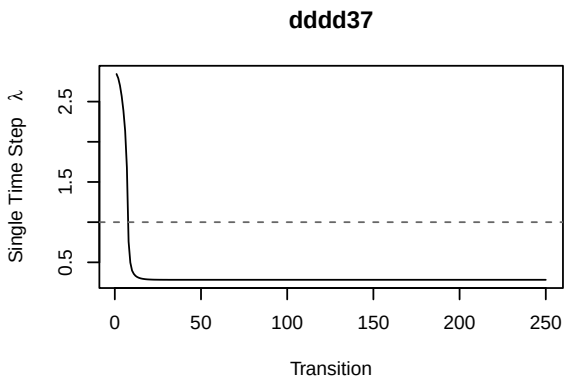

dddd29
Single Time Step λ
1.0
1.1
1.2
1.3
0
10
20
30
40
50
Transition

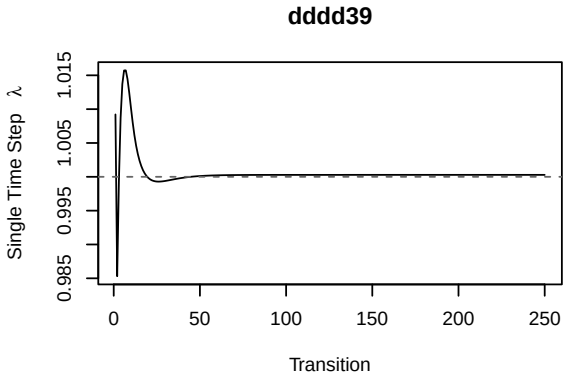

dddd30
Single Time Step λ
0.5
1.0
1.5
2.0
2.5
0
50
100
150
200
250
Transition

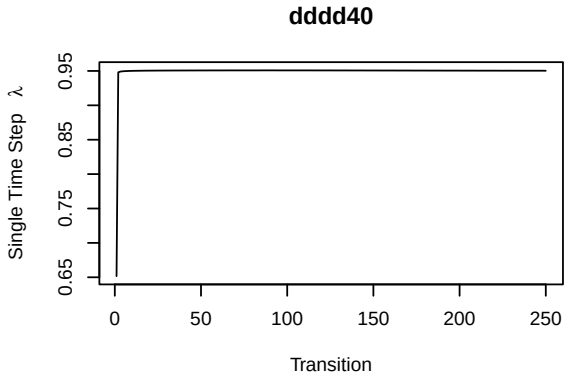

dddd33
Single Time Step λ
1.0
1.4
1.8
0
50
100
150
200
250
Transition

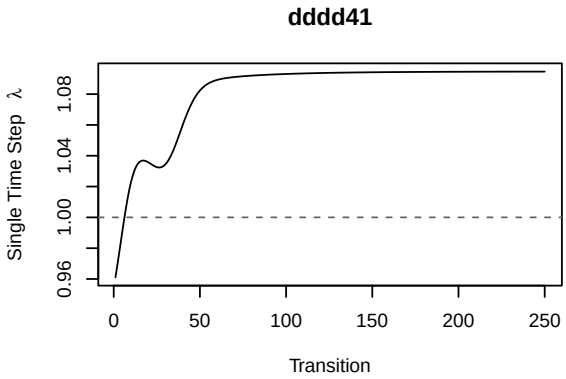

dddd35
Single Time Step λ
1.10
1.15
1.20
0
50
100
150
200
250
Transition


dddd36
Single Time Step λ
0.96
1.00
1.04
0
50
100
150
200
250
Transition


dddd37
Single Time Step λ
0.5
1.5
2.5
0
50
100
150
200
250
Transition


dddd39
Single Time Step λ
0.985
0.995
1.005
1.015
0
50
100
150
200
250
Transition


dddd40
Single Time Step λ
0.65
0.75
0.85
0.95
0
50
100
150
200
250
Transition


dddd41
Single Time Step λ
0.96
1.00
1.04
1.08
0
50
100
150
200
250
Transition

`dddd24`, `dddd26`, `dddd40` and `dddd41` are still not converging. We will remove those from our further analyses:

```
keep_ind <- setdiff(names(new_ipms), c("dddd24", "dddd26",
                                       "dddd40", "dddd41"))

new_ipms <- new_ipms[keep_ind]
```

# Further analyses

*Rpadrino* contains methods for most of *ipmr*'s analysis functions. These include (but are not limited to!) `lambda`, `left_ev`, and `right_ev`. We need all three of these to compute sensitivity and elasticity. We also need the binwidth of the integration mesh so we can perform integrations. *Rpadrino*'s has the `int_mesh()` function for that, and the binwidth is always the first element in the list that it returns. We can extract them like so:

```
lambdas    <- lambda(new_ipms)
repro_vals <- left_ev(new_ipms, tolerance = 1e-5)
ssd_vals   <- right_ev(new_ipms, tolerance = 1e-5)

d_zs <- lapply(new_ipms, function(x) int_mesh(x, full_mesh = FALSE)[[1]])
```

## Sensitivity

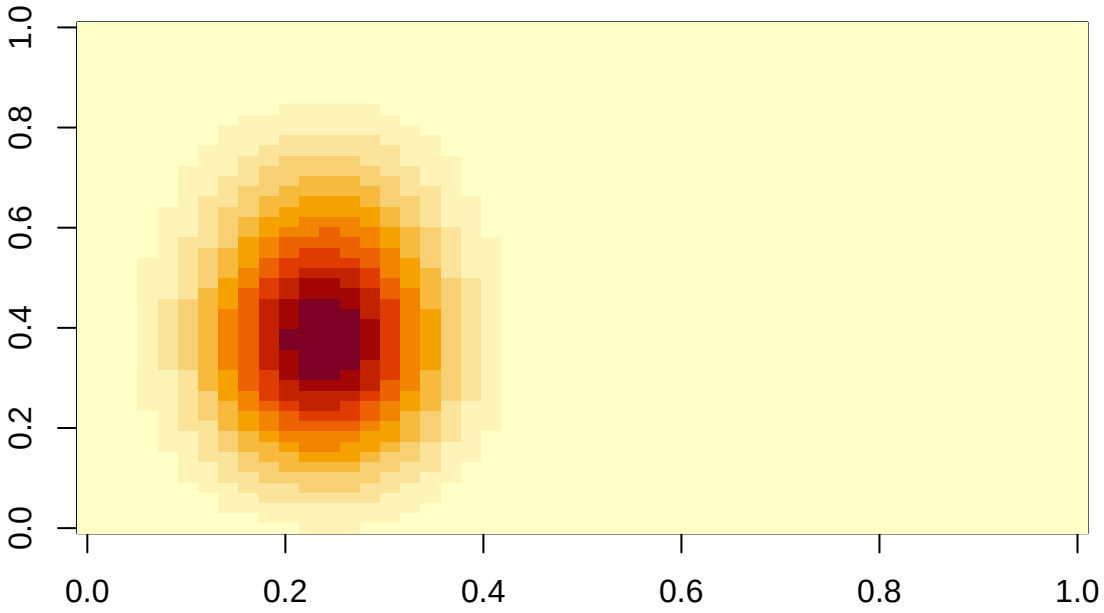
With these, we can now compute sensitivity. This is given by $\mathbf{s}(z_0', z_0) = \frac{v(z_0')w(z_0)}{\langle v,w \rangle}$, where $v(z_0')$ is the left eigenvector and $w(z_0)$ is the right eigenvector. It will be helpful to write a function that takes these values as arguments and returns the sensitivity kernel. We will use `lapply(seq_along())` to iterate over each model.

```
# r_evs: right eigenvectors
# l_evs: left eigenvectors
# d_zs:  binwidths
# lapply(seq_along()) generates a sequence of numbers that correspond to indices
# in the list of eigenvectors and binwidths. Since each of these objects is a
# list of lists, we need to use [[index]][[1]]. The first[[]] gets the correct
# list entry, and the second [[1]] converts it to a numeric vector by unlisting
# the second layer of the list.
# NB: Some tidyverse-oriented users may wish to substitute this with pmap() and
# pluck(). This is fine, just not demonstrated here.

sens <- function(r_evs, l_evs, d_zs) {

  lapply(seq_along(r_evs),
         function(ind, r_ev, l_ev, d_z){

           outer(l_ev[[ind]][[1]], r_ev[[ind]][[1]]) /
             (sum(l_ev[[ind]][[1]] * r_ev[[ind]][[1]] * d_z[[ind]]))

         },
         r_ev = r_evs,
         l_ev = l_evs,
         d_z = d_zs)
}
```

```
sens_list <- sens(ssd_vals, repro_vals, d_zs) %>%
  setNames(names(lambdas))
```

We can plot these using `image()`:

```
par(mfrow = c(1, 1))
image(t(sens_list$aaa385))
```




## Elasticity

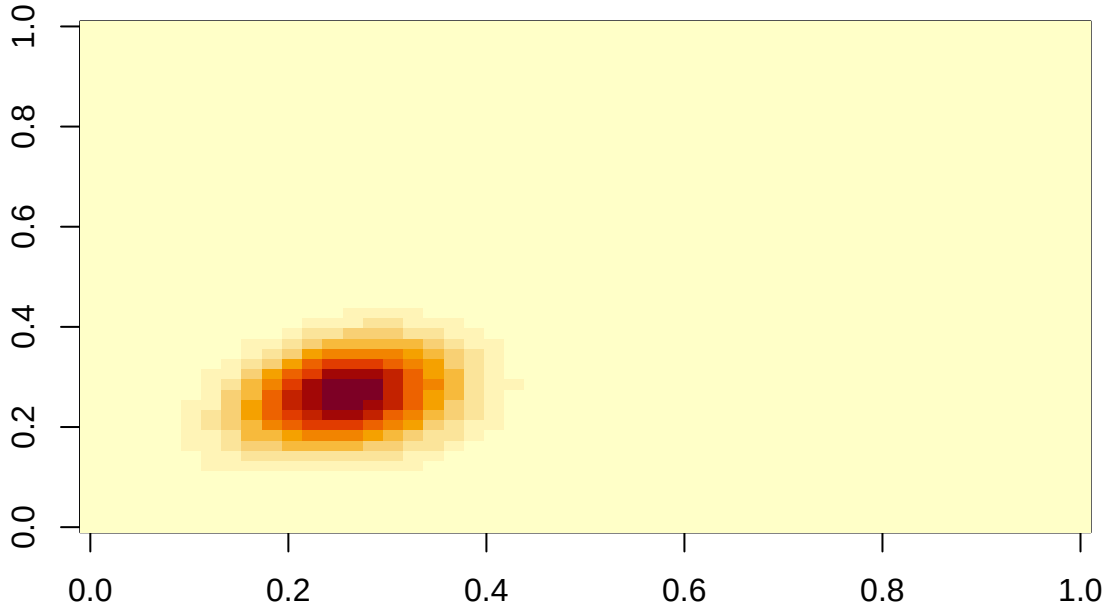
We can compute elasticity without much more effort. We need one more piece of information from the IPM list that we have not extracted - the iteration kernel. We can get those using `make_iter_kernel()` on the `new_ipms` object, and then computing the elasticity using $\mathbf{e}(z_0', z_0) = \frac{K(z_0', z_0)}{\lambda}\mathbf{s}(z_0', z_0)$.

```
iter_kerns <- make_iter_kernel(new_ipms)

elas_list <- lapply(seq_along(iter_kerns),
                    function(ind, iter_kernels, sens_kernels, lambdas, d_zs) {

                      (iter_kernels[[ind]][[1]] / d_zs[[ind]] / lambdas[[ind]]) *
                        sens_kernels[[ind]]

                    },
                    iter_kernels = iter_kerns,
                    sens_kernels = sens_list,
```

```
                    lambdas = lambdas,
                    d_zs = d_zs) %>%
  setNames(names(lambdas))
```

Similarly, we can plot this using `image()`:

```
image(t(elas_list$aaa385))
```




## Mean lifetime recruit production

We can calculate the expected number of recruits produced over an individuals lifetime as function of its initial size. This is defined as $\bar{r}(z_0) = eFN$. $F$ is the fecundity kernel, and we can get these kernels from each IPM in our list using:

```
F_kerns <- lapply(new_ipms, function(x) x$sub_kernels$F)
```

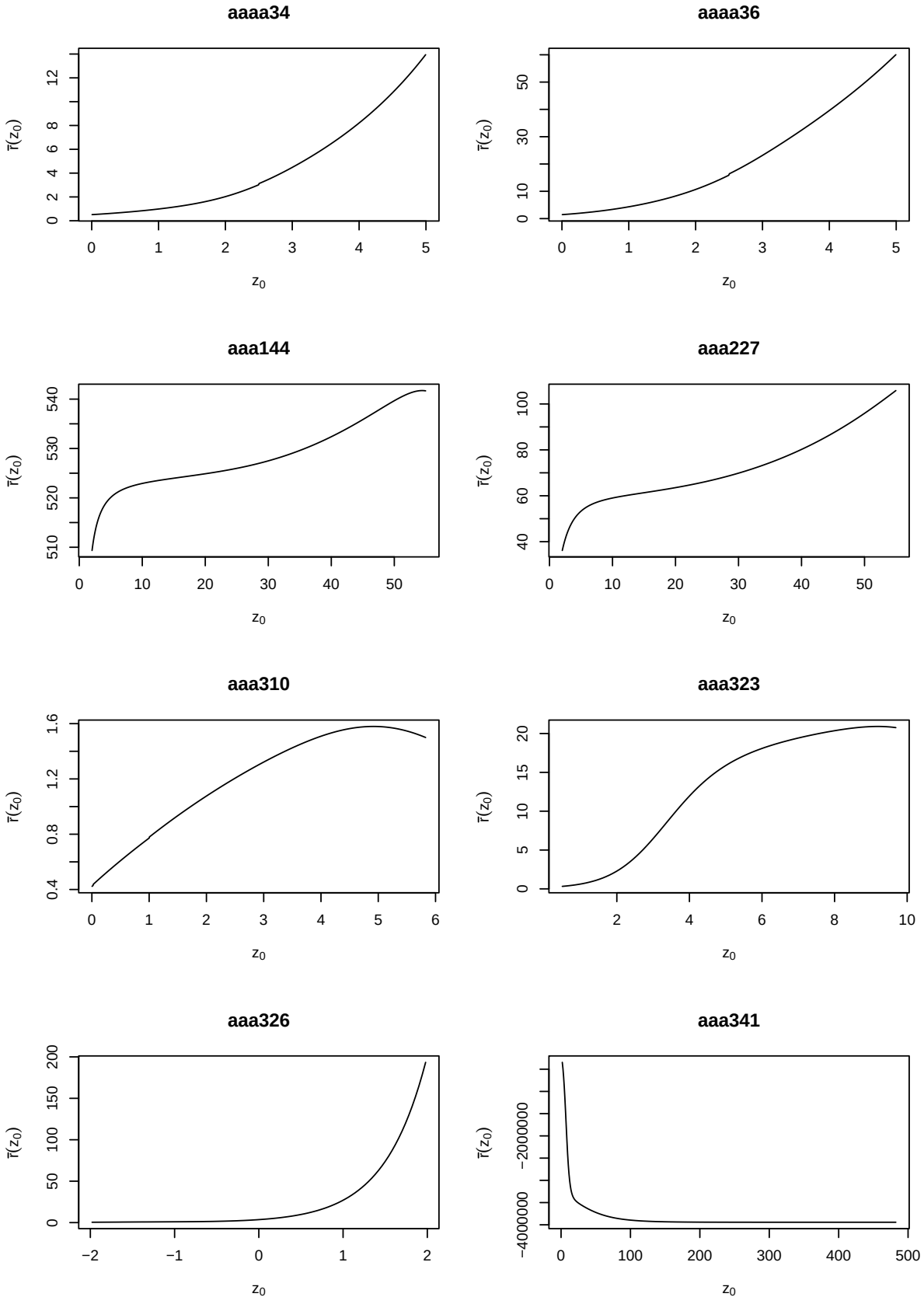
$N$ is the fundamental operator. This tells us the expected amount of time an individual will spend in state $z'$ given an initial state $z_0$. The fundamental operator is defined as $(I - P)^{-1}$ (see Ellner, Childs, & Rees 2016 Chapter 3 for the derivation of this). $I$ is an identity kernel ($I(z', z) = 1$ for $z' = z$, and $I(z', z) = 0$ for $z' \neq z$). This code is only a bit more complicated:

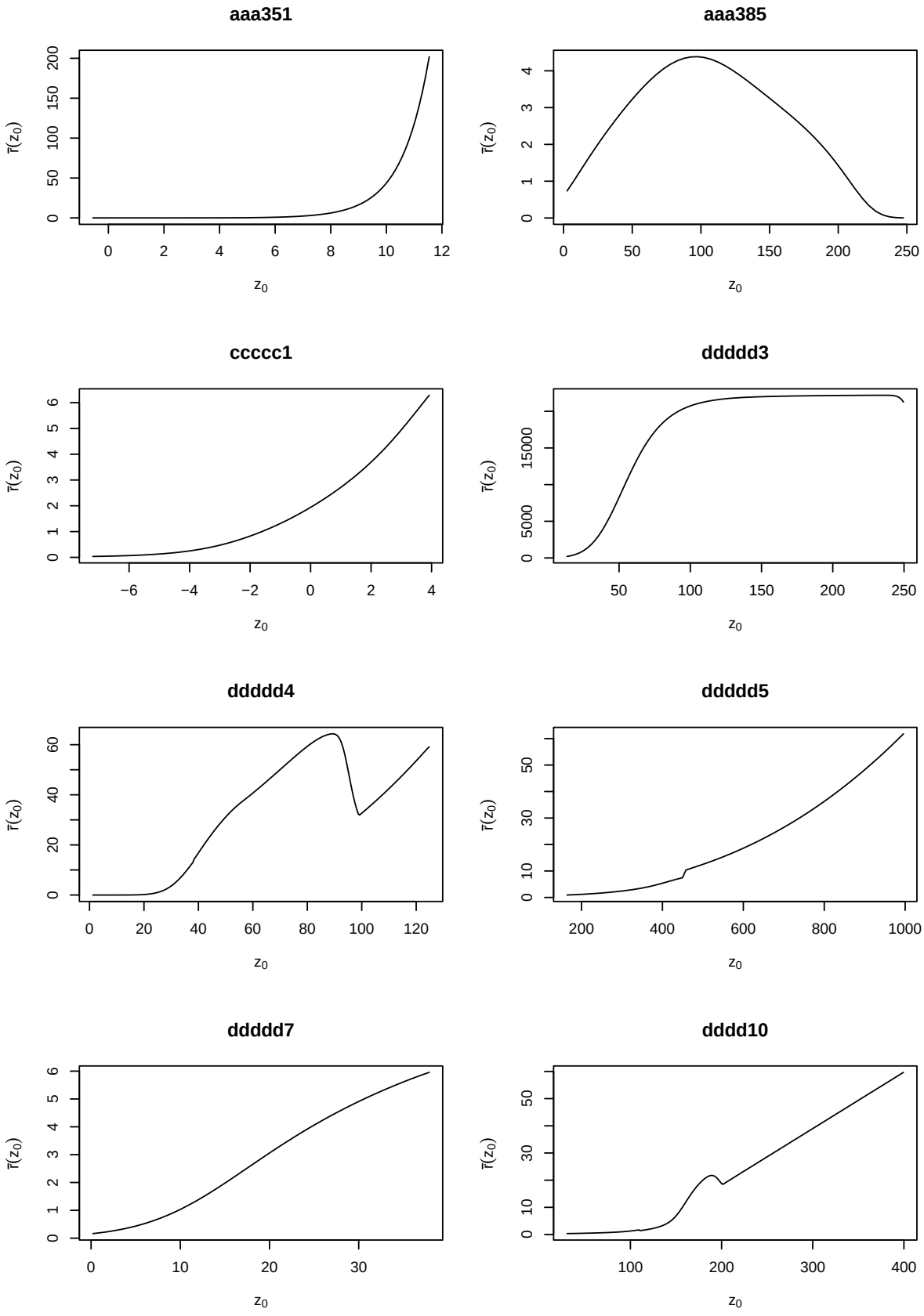
```
# Function to create an identity kernel with dimension equal to P
make_i <- function(P) {
  return(
    diag(nrow(P))
  )
```

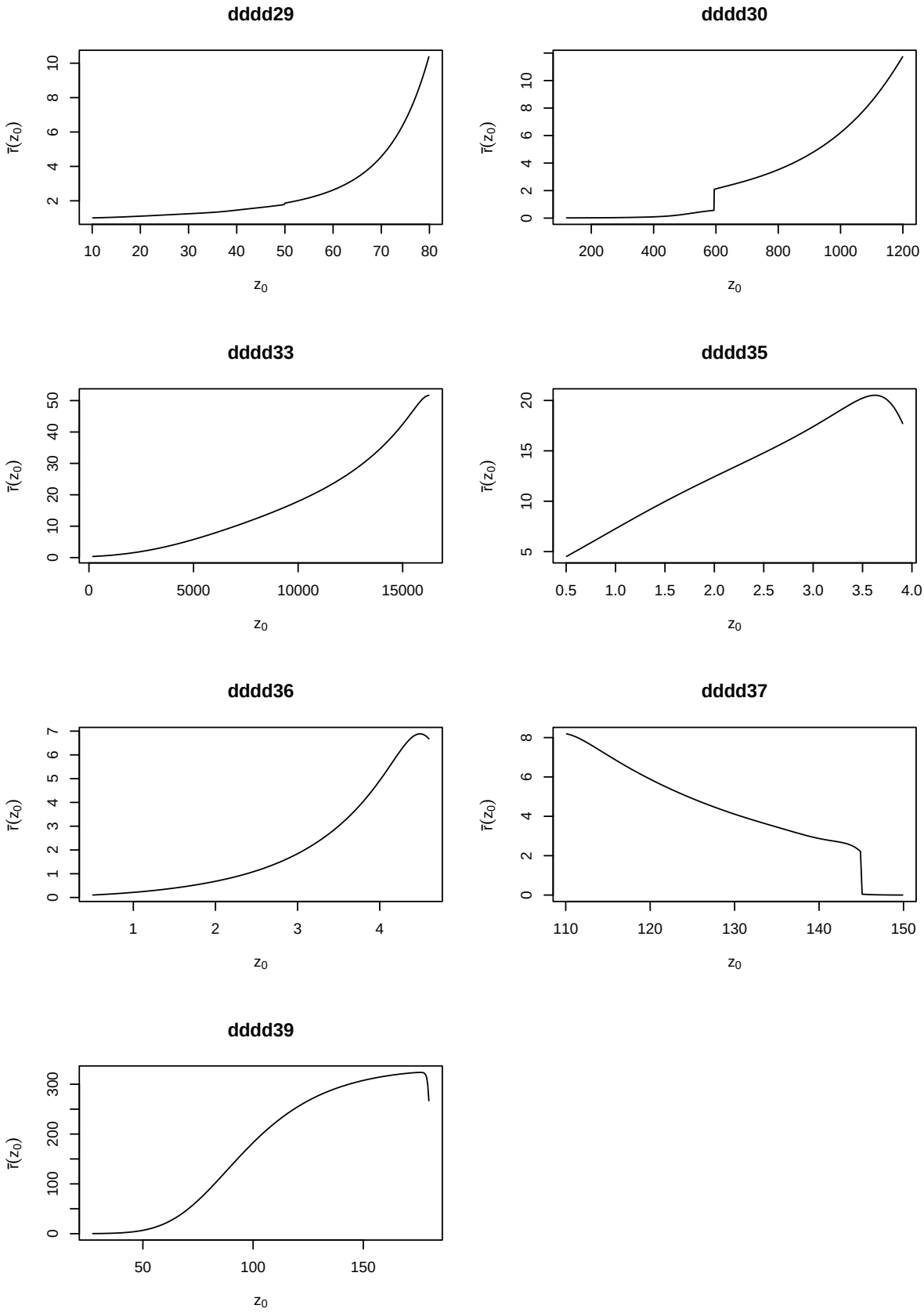
```
}

N_kerns <- lapply(new_ipms, function(x) {

  P <- x$sub_kernels$P
  I <- make_i(P)

  # solve() inverts the matrix for us (the ^(-1) part of the equation)
  solve(I - P)

})
```

$e$ is a constant function $e(z) \equiv 1$. In practice, the left multiplication of $eF$ has the effect of computing the column sums of $F$. We will replace the $e$ with a call to `colSums()` in our code below (this will run faster than doing the multiplication). We now have everything we need to compute and visualize the expected lifetime reproductive output:

```
# We wrap the computation in as.vector so that it returns a simple numeric vector
# rather than a 1 x N matrix

r_bars <- lapply(seq_along(N_kerns),
                 function(idx, Fs, Ns) {

                   as.vector(colSums(Fs[[idx]]) %*% Ns[[idx]])

                 },
                 Fs = F_kerns,
                 Ns = N_kerns) %>%
  setNames(names(F_kerns))

# we will extract the meshpoint values so that the x-axes on our plots look
# prettier.
x_seqs <- lapply(new_ipms,function(x) int_mesh(x, full_mesh = FALSE)[[2]])


# Finally, we can plot the data by looping over the lists and creating a
# a simple line plot (type = "l")

par(mfrow = c(4, 2))

for(i in seq_along(r_bars)) {


  plot(r_bars[[i]], x = x_seqs[[i]], type = 'l', main = names(r_bars)[i],
       ylab = expression(bar(r)(z[0])),
       xlab = expression(z[0]))

}
```


aaaa34
aaaa36
aaa144
aaa227
aaa310
aaa323
aaa326
aaa341
r̄(z0)
z0


aaa351
aaa385
ccccc1
ddddd3
ddddd4
ddddd5
ddddd7
dddd10
r̄(z0)
z0


dddd29
r̄(z0)
z0
dddd30
r̄(z0)
z0
dddd33
r̄(z0)
z0
dddd35
r̄(z0)
z0
dddd36
r̄(z0)
z0
dddd37
r̄(z0)
z0
dddd39
r̄(z0)
z0

## Troubleshooting and Modifying IPMs

The plot of the output above provides a cautionary tale: $\bar{r}(z_0)$ is negative in model `aaa341`! This is not biologically possible - we can not make negative recruits. We know there is some quirk in that model that produces these results. PADRINO would flag a model with negative numbers in the $F$ kernel when it's built, so we need to check the $N$ kernel for that model for negative values.

```
range(N_kerns$aaa341)
```

```
## [1] -5008.038563     1.020386
```

We have found the problem in the $N$ kernel, but what causes that? It turns out that the $P$ kernel, when discretized, is either close to or exactly singular, and so the determinant $(I - P)$ is very close to singular as well (and negative!). This can happen when the survival function is exactly 1 for some range of initial trait values (Ellner, Childs & Rees 2016, Chapter 3).

```
P <- new_ipms$aaa341$sub_kernels$P
I <- make_i(P)
det(I - P) # negative and very close to 0
```

```
## [1] -1.29379e-09
```

It is important to remember that PADRINO provides models *as they are published*, and does not try to correct these problems. Thus, data in here can cause problems if not treated with care!

We can try to modify the model very slightly to see if we can make this kernel non-singular. We will try to set a parallel minimum for the `s` function value in that model, which will hopefully pull our $det(I - P)$ into positive territory. We need to work with the `proto_ipm` object for this, because we want to propagate the function value changes to the sub-kernels (*i.e.* the P kernel).

We can alter the `s` function with `vital_rate_exprs<-` and `pdb_new_fun_form()`. We will update it to take the parallel minimum of the published `s` function, and the maximum value we'd want the function to ideally have (in this case, 0.98).

```
# First, peak at the current functional form
vital_rate_exprs(simple_det_list)$aaa341
```

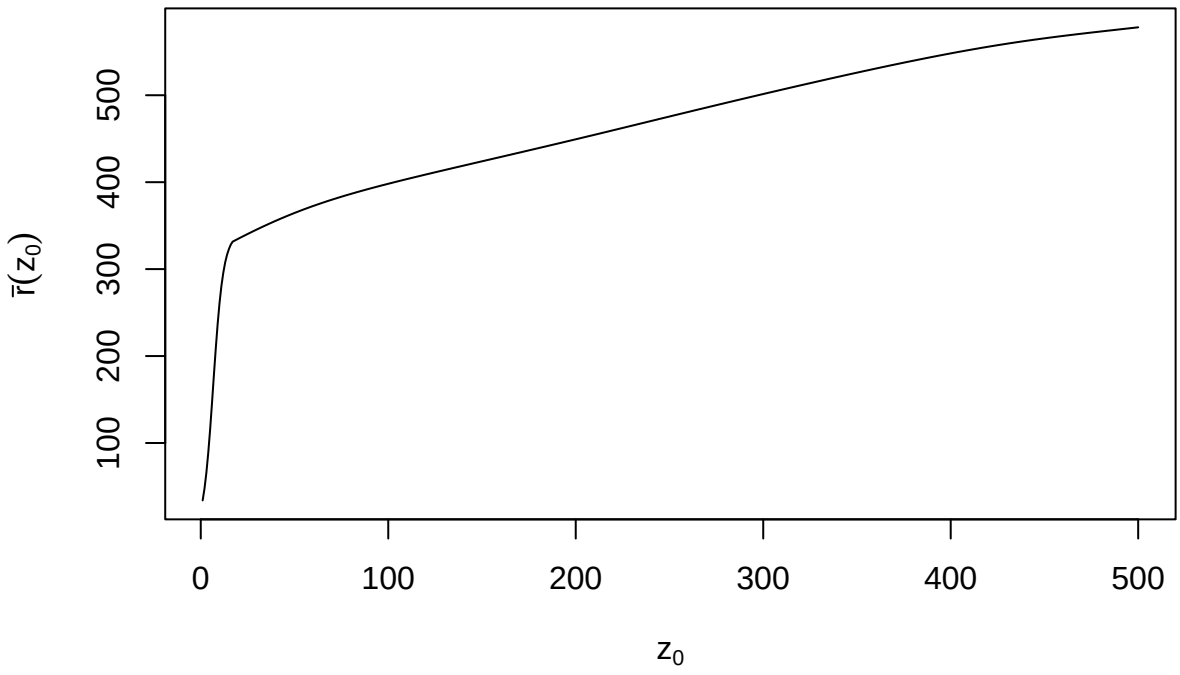
```
## s: 1/(1 + exp(-(si + ss1 * size_1 + ss2 * size_1^2)))
## g_mean: gi + gs * size_1
## g: dnorm(size_2, g_mean, g_sd)
## Fp: 1/(1 + exp(-(fpi + fps * size_1)))
## Fs: exp(fi + fs * size_1)
## Fd: dnorm(size_2, fd_mean, fd_sd)
```

```
# Now, update it with our desired maximum value
vital_rate_exprs(simple_det_list) <- pdb_new_fun_form(
  list(
    aaa341 = list(
      s = pmin(0.98, 1/(1 + exp(-(si + ss1 * size_1 + ss2 * size_1^2))))
    )
  )
)

# we will skip rebuilding the whole list - we just want to make sure we have fixed
# this particular model. The species in this model is Lonicera maackii.

new_lonicera <- pdb_make_ipm(simple_det_list["aaa341"])
```

```
P <- new_lonicera$aaa341$sub_kernels$P
I <- make_i(P)

cat("New det(I - P) is: ", det(I - P))
```

```
## New det(I - P) is:  1.22853e-05
```

```
N <- solve(I - P)
F <- new_lonicera$aaa341$sub_kernels$F

r_bar_lonicera <- as.vector(colSums(F) %*% N)

plot(r_bar_lonicera,
     type = 'l',
     ylab = expression(bar(r)(z[0])),
     xlab = expression(z[0]))
```




```
# Since we are now happier with how this IPM is behaving, we will insert it into
# our list for subsequent analyses
new_ipms$aaa341 <- new_lonicera$aaa341
```

That looks a bit closer to reality!

# Vital rate and parameter level analyses

We can also run analyses at the parameter and the vital rate level for PADRINO. These require more care - it is strongly recommended to check the original publications to for the meaning of each parameter and vital rate. There is simply too much variability in the way vital rates and parameters are estimated in the literature to provide systematic descriptions of them in PADRINO. With this caveat in mind, we will proceed to a couple examples of using vital rate functions and parameters in further analyses.

The ability to perturb function values and parameter estimates is one of the great strengths of IPMs. Furthermore, computing many life history traits requires the values of vital rate functions. Therefore, we took great pains when designing the database to ensure these analyses were still possible. As noted above, they require some additional effort, but are usually worth it. We will step through the code pieces required to extract these below. We will start with an example computing the sensitivity of $\lambda$ to vital rate function values. After that, we will show how to compute the mean size at death conditional on initial state $z_0$ and the size at death kernel $\Omega(z', z_0)$, both of which rely on extracting the survival functions.

## Vital rate function value perturbations

These require modifying the general sensitivity formula to compute the partial derivative of $\lambda$ with respect to change in $f(z)$. These expressions depend on the form of the kernel, and so no general formula exists for function value perturbations. However, we can use the chain rule and the general formula for sensitivity to a given perturbation to work it out. The latter formula is:

1. $\left.\frac{\partial\lambda}{\partial\epsilon}\right|_{\epsilon=0} = \frac{\langle v, Cw\rangle}{\langle v, w\rangle}$.

Here, $v$ and $w$ are the left and right eigenvectors of the iteration kernel (provided by `left_ev` and `right_ev`), and $C$ is the perturbation kernel, which we will need to identify. We will take the following steps:

1. Identify models we want to use.
2. Inspect the kernel formulae and vital rate functions using `print` methods.
3. Write down the perturbation kernels for each model.
4. Construct the IPM objects and extract $v$ and $w$.
5. Implement the perturbations in $R$.

### Identifying models (1)

In order to keep things simple, we will work with models where survival only occurs in 1 kernel. There are numerous examples of how to extend these analyses elsewhere (*e.g.* Ellner, Childs & Rees 2016). We can look into this using the `pdb$VitalRateExpr$kernel_id` column in conjunction with the `pdb$VitalRateExpr$demographic_paratemer` column.

```
# Find all rows in VitalRateExpr corresponding to survival
init_ind <-
  my_pdb$VitalRateExpr[
    my_pdb$VitalRateExpr$demographic_parameter == "Survival", ]

# Next, select ipm_id's that have survival functions that only show up
# in "P"

keep_ind <- init_ind$ipm_id[init_ind$kernel_id == "P"] %>%
  unique()
```

```
# To keep things quick, we will just use the first 3 in this index

keep_ind <- keep_ind[1:3]

vr_sens_pdb <- pdb_subset(my_pdb, keep_ind)
```

### Inspect the kernels and vital rate expressions (2)

Next, we will construct `proto_ipm` objects for each model and check to see how the kernels are constructed:

```
proto_list <- pdb_make_proto_ipm(vr_sens_pdb)

proto_list[[1]]
```

```
## A simple, density independent, deterministic proto_ipm with 2 kernels defined:
##  P, F
##
## Kernel formulae:
##
## P: s * g
## F: r * fn * pE * d
##
## Vital rates:
##
## s: 1/(1 + exp(-(s_b + s_m * lnsize_1)))
## g_mean: g_b + g_m * lnsize_1
## g_var: sqrt(gv_b + gv_m * lnsize_1)
## g: dnorm(lnsize_2, g_mean, g_var)
## r: 1/(1 + exp(-(r_b + r_m * lnsize_1)))
## fn: exp(fn_b + fn_m * lnsize_1)
## d: dexp(lnsize_2, 1/d_mean)
##
## Parameter names:
##
##  [1] "s_b"    "s_m"    "g_b"    "g_m"    "gv_b"   "gv_m"   "r_b"    "r_m"
##  [9] "fn_b"   "fn_m"   "t_r"    "d_mean" "pE"
##
## All parameters in vital rate expressions found in 'data_list':  TRUE
##
## Domains for state variables:
##
## lnsize: lower_bound = 0, upper_bound = 5, n_meshpoints = 500
##
## Population states defined:
##
## n_lnsize: Pre-defined population state.
##
## Internally generated model iteration procedure:
##
## n_lnsize_t_1: right_mult(kernel = P, vectr = n_lnsize_t) + right_mult(kernel = F,
##     vectr = n_lnsize_t)
```

```
proto_list[[2]]
```

```
## A simple, density independent, deterministic proto_ipm with 2 kernels defined:
##  P, F
##
## Kernel formulae:
##
## P: s * g
## F: r * fn * pE * d
##
## Vital rates:
##
## s: 1/(1 + exp(-(s_b + s_m * lnsize_1)))
## g_mean: g_b + g_m * lnsize_1
## g_var: sqrt(gv_b + gv_m * lnsize_1)
## g: dnorm(lnsize_2, g_mean, g_var)
## r: 1/(1 + exp(-(r_b + r_m * lnsize_1)))
## fn: exp(fn_b + fn_m * lnsize_1)
## d: dexp(lnsize_2, 1/d_mean)
##
## Parameter names:
##
##  [1] "s_b"    "s_m"    "g_b"    "g_m"    "gv_b"   "gv_m"   "r_b"    "r_m"
##  [9] "fn_b"   "fn_m"   "t_r"    "d_mean" "pE"
##
## All parameters in vital rate expressions found in 'data_list':  TRUE
##
## Domains for state variables:
##
## lnsize: lower_bound = 0, upper_bound = 5, n_meshpoints = 500
##
## Population states defined:
##
## n_lnsize: Pre-defined population state.
##
## Internally generated model iteration procedure:
##
## n_lnsize_t_1: right_mult(kernel = P, vectr = n_lnsize_t) + right_mult(kernel = F,
##     vectr = n_lnsize_t)
```

```
proto_list[[3]]
```

```
## A simple, density independent, deterministic proto_ipm with 2 kernels defined:
##  P, F
##
## Kernel formulae:
##
## P: s * g
## F: rep_p * es_p * sdl_s * n_infl * n_fl * n_seed
##
## Vital rates:
##
## s: ssurv * wsurv
## ssurv: exp(ssurv_i + ssurv_s * ((log(size_1^3) - smlv)/sslv) + ssurv_el *
##     elev_s)/(1 + exp(ssurv_i + ssurv_s * ((log(size_1^3) - smlv)/sslv) +
##     ssurv_el * elev_s))
## wsurv: exp(wsurv_i + wsurv_s * ((log(size_1^3) - wmlv)/wslv) + wsurv_f *
```

```
##     cumfrost)/(1 + exp(wsurv_i + wsurv_s * ((log(size_1^3) -
##     wmlv)/wslv) + wsurv_f * cumfrost))
## g_mean: sign(growth) * abs(growth)^(1/3)
## growth: g_i + g_el * elev_g + g_frost * annfrost + g_s * (size_1^3)
## g: dnorm(size_2, g_mean, g_sd)
## rep_p: fl_p * germ_p
## fl_p: exp(fl_i + fl_s * ((log(size_1^3) - fmlv)/fslv))/(1 + exp(fl_i +
##     fl_s * ((log(size_1^3) - fmlv)/fslv)))
## germ_p: exp(germ_i + germ_el * elev_germ)/(1 + exp(germ_i + germ_el *
##     elev_germ))
## n_infl: exp(infl_n)
## n_fl: exp(fl_n)
## n_seed: exp(seed_i)
## sdl_s: dnorm(size_2, sdl_mean, sdl_sd)
##
## Parameter names:
##
##  [1] "ssurv_i"   "ssurv_el"  "ssurv_s"   "wsurv_i"   "wsurv_s"   "wsurv_f"
##  [7] "g_i"       "g_el"      "g_frost"   "g_s"       "g_sd"      "fl_i"
## [13] "fl_s"      "germ_i"    "germ_el"   "es_p"      "sdl_mean"  "sdl_sd"
## [19] "infl_n"    "fl_n"      "seed_i"    "smlv"      "sslv"      "wmlv"
## [25] "wslv"      "fmlv"      "fslv"      "annfrost"  "cumfrost"  "elev_germ"
## [31] "elev_g"    "elev_s"
##
## All parameters in vital rate expressions found in 'data_list':  TRUE
##
## Domains for state variables:
##
## size: lower_bound = 2, upper_bound = 55, n_meshpoints = 1000
##
## Population states defined:
##
## n_size: Pre-defined population state.
##
## Internally generated model iteration procedure:
##
## n_size_t_1: right_mult(kernel = P, vectr = n_size_t) + right_mult(kernel = F,
##     vectr = n_size_t)
```

We can see that for each IPM, $P = s(z) * G(z', z)$. In the third models, $s(z)$ is comprised of two additional functions, which we can perturb individually. We will show a quick example of that after applying our perturbation kernels to all 3.

### Write out the perturbation kernels (3)

Our perturbation kernel for all 3 models will take the following form: $C(z', z) = \delta_{z_0}(z)G(z', z)$. With a little rearranging (see Ellner, Childs, & Rees 2016 Chapter 4), we find the following:

$$\frac{\partial \lambda}{\partial s(z_0)} = \frac{\int v(z')G(z', z_0)w(z_0)dz'}{\int v(z)w(z)dz} = \frac{(vG) \circ w}{\langle v, w \rangle}.$$

The second portion of the equations above is the first part re-written to use operator notation (which drops the $z$s and $z'$s for brevity). The $\circ$ denotes point-wise multiplication.

### Implement the models (4)

Now that we have written down our perturbation formulae, we need to rebuild the models, and make use of some non-standard arguments to `pdb_make_ipm`. By default, *Rpadrino* does not return the vital rate functions values. To get those, we need to specify `return_all_envs = TRUE` in the `addl_args` list.

```
arg_list <- lapply(keep_ind, function(x) list(return_all_envs = TRUE)) %>%
  setNames(keep_ind)

ipm_list <- pdb_make_ipm(proto_list, addl_args = arg_list)

r_evs    <- right_ev(ipm_list)
l_evs    <- left_ev(ipm_list)
```

### Implement the perturbations (5)

Next, we need to implement the formula above. This is fairly straightforward, and we will make use of another function in *Rpadrino*: `vital_rate_funs()` (not to be confused with `vital_rate_exprs()`!). This extracts the vital rate function values from each model and returns them in a named list (`vital_rate_exprs()` extracts the expressions that create these values). Let's see what this looks like:

```
vr_funs <- vital_rate_funs(ipm_list)

vr_funs$aaaa34
```

```
## $P
## s (not yet discretized): A 500 x 500 kernel with minimum value: 0.3638 and maximum value: 0.852
## g_mean (not yet discretized): A 500 x 500 kernel with minimum value: 1.1114 and maximum value: 3.680
## g_var (not yet discretized): A 500 x 500 kernel with minimum value: 0.9749 and maximum value: 0.9749
## g (not yet discretized): A 500 x 500 kernel with minimum value: 0 and maximum value: 0.1293
##
## $F
## r (not yet discretized): A 500 x 500 kernel with minimum value: 0.0062 and maximum value: 0.9969
## fn (not yet discretized): A 500 x 500 kernel with minimum value: 43.05 and maximum value: 1280.9888
## d (not yet discretized): A 500 x 500 kernel with minimum value: 0 and maximum value: 0.1481
```

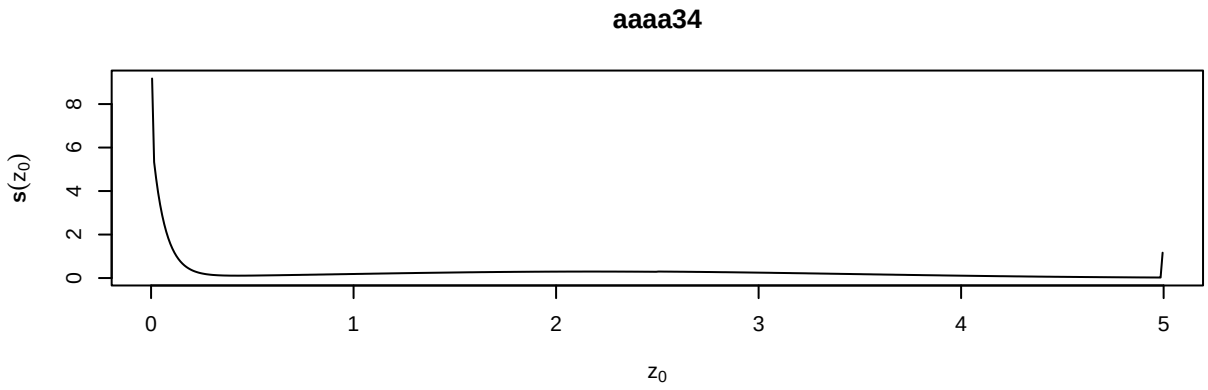
```
vr_funs$aaa144
```

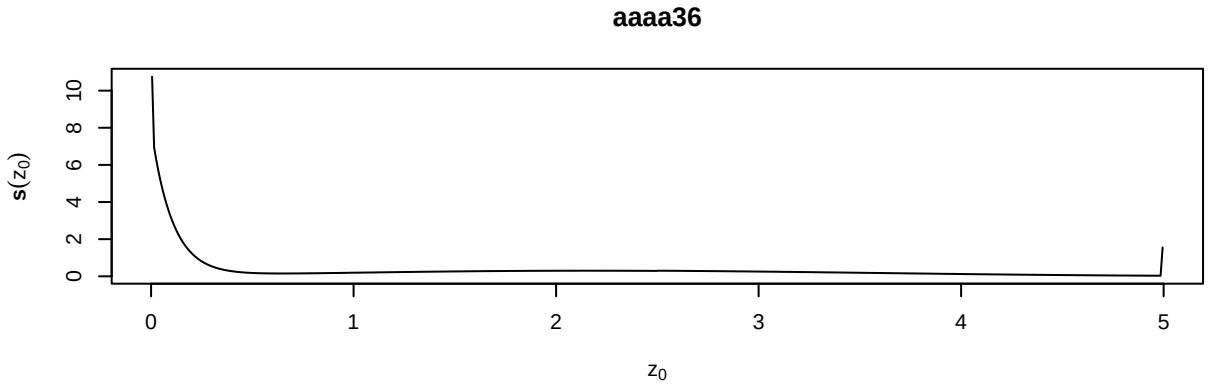
```
## $P
## s (not yet discretized): A 1000 x 1000 kernel with minimum value: 0.9712 and maximum value: 0.9997
## ssurv (not yet discretized): A 1000 x 1000 kernel with minimum value: 0.9729 and maximum value: 0.999
## wsurv (not yet discretized): A 1000 x 1000 kernel with minimum value: 0.9983 and maximum value: 1
## g_mean (not yet discretized): A 1000 x 1000 kernel with minimum value: 18.6951 and maximum value: 42
## growth (not yet discretized): A 1000 x 1000 kernel with minimum value: 6534.029 and maximum value: 75
## g (not yet discretized): A 1000 x 1000 kernel with minimum value: 0 and maximum value: 0.0798
##
## $F
## rep_p (not yet discretized): A 1000 x 1000 kernel with minimum value: 0 and maximum value: 0.0049
## fl_p (not yet discretized): A 1000 x 1000 kernel with minimum value: 0 and maximum value: 0.3948
## germ_p (not yet discretized): A 1000 x 1000 kernel with minimum value: 0.0124 and maximum value: 0.0
## n_infl (not yet discretized): A 1000 x 1000 kernel with minimum value: 2.4843 and maximum value: 2.48
## n_fl (not yet discretized): A 1000 x 1000 kernel with minimum value: 131.6307 and maximum value: 131
## n_seed (not yet discretized): A 1000 x 1000 kernel with minimum value: 13.1971 and maximum value: 13
## sdl_s (not yet discretized): A 1000 x 1000 kernel with minimum value: 0 and maximum value: 0.3988
```

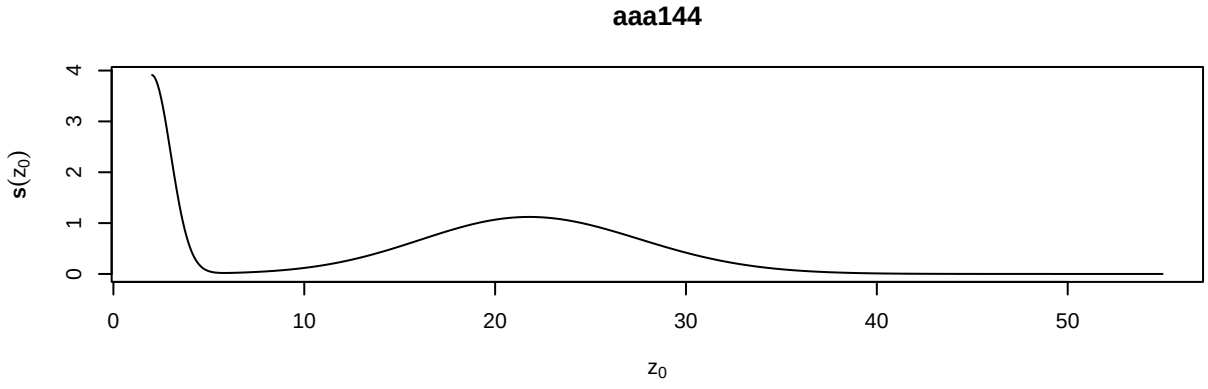
We see from the printed values that each vital rate function contains the complete $n \times n$ set of values for each combination of meshpoints. Additionally, it warns us that these are not yet integrated. This is actually a good thing - we want the continuous function values for the sensitivity, not the discretized values. We know from our formula above that we need to extract $G(z', z)$, and that these are named `g` in each model. This will not always be true for PADRINO, so care must be taken at this step to make sure you extract the correct values!

Recall that we are about to implement: $\frac{(vG) \circ w}{\langle v,w \rangle}$. The $\langle ... \rangle$ is the inner product of $v, w$, and so we also need the value of $dz$ to implement the denominator. We will get those using `int_mesh()` again.

```
mesh <- lapply(ipm_list, function(x) int_mesh(x, full_mesh = FALSE))
d_zs <- lapply(mesh, function(x) x[[1]])

sens_list <- lapply(seq_along(vr_funs),
                    function(idx, r_evs, vr_funs, l_evs, d_zs) {

                      # Extract objects to temporary values so the formula is more
                      # readable

                      G <- vr_funs[[idx]]$P$g
                      v <- unlist(l_evs[[idx]])
                      w <- unlist(r_evs[[idx]])
                      d_z <- d_zs[[idx]]

                      numerator <- as.vector((v %*% G) * w)
                      denominator <- sum(v * w * d_z)

                      numerator / denominator

                    },
                    r_evs = r_evs,
                    l_evs = l_evs,
                    vr_funs = vr_funs,
                    d_zs = d_zs)

par(mfrow = c(3, 1))

for(i in seq_along(sens_list)) {

  plot(sens_list[[i]], x = mesh[[i]][[2]],
       type = "l",
       main = names(mesh)[i],
       ylab = expression(bold(s)(z[0])),
       xlab = expression(z[0]))

}
```



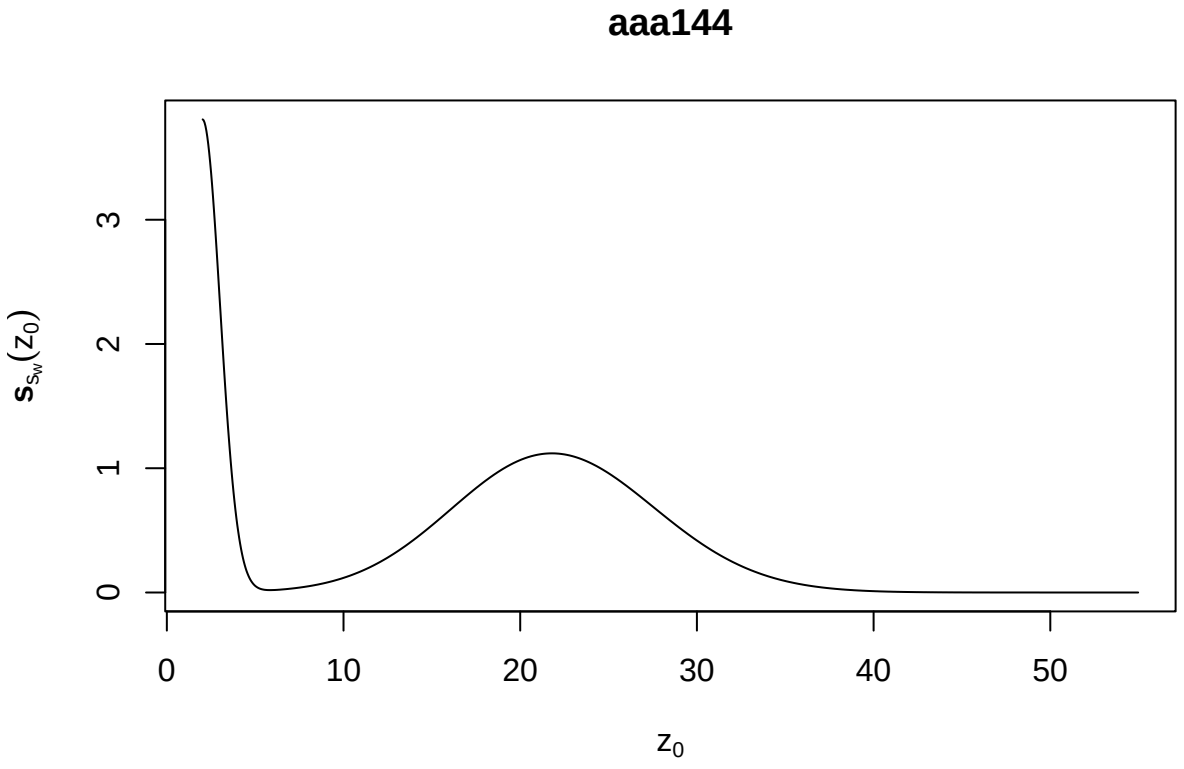




**Other perturbation kernels**

As mentioned above, the survival function in the last IPM is comprised of two additional functions: `ssurv` and `wsurv`. Thus, we could re-write the $P$ kernel as $P(z', z) = s_s(z) * s_w(z) * G(z', z)$. Thus, if we wanted to know the effect of perturbing only $s_w(z)$, we would re-write our perturbation kernel as $C(z', z) = \delta(z_0) * s_s(z) * G(z', z)$, and our perturbation formula (in operator notation) becomes $\mathbf{s}_s(z_0) = \frac{(v s_s G) \circ w}{\langle v, w \rangle}$. We will drop the first model from our lists because this analysis does not apply to it. We can implement this by slightly modifying the code above:

```
s_s <- vr_funs[[3]]$P$ssurv
G <- vr_funs[[3]]$P$g
v <- unlist(l_evs[[3]])
w <- unlist(r_evs[[3]])
d_z <- d_zs[[3]]

numerator   <- as.vector((v %*% (s_s * G)) * w)
denominator <- sum(v * w * d_z)

sens_s_w <- numerator / denominator

par(mfrow = c(1, 1))


mesh_ps <- mesh[[3]][[2]]
nm      <- names(mesh)[3]

plot(y = sens_s_w, x = mesh_ps,
     type = "l",
     main = nm,
     ylab = expression(bold(s)[s[w]](z[0])),
     xlab = expression(z[0]))
```



### Mean size at death and size at death kernels

We can also use vital rate function values in conjunction with sub-kernels to implement calculations of life history traits. For this example, we will examine mean size at death and the size at death kernel. These are given by the following equations:

Mean size at death $= \bar{\omega} = (\mathbf{i} \circ (1 - s))N$ and size at death kernel $= \Omega(z', z_0) = (1 - s(z')) * N(z', z_0)$.

In these equations, $s$ and $s(z')$ are survival functions, and $N$ is the fundamental operator, which is defined as $N = (I - P)^{-1}$, where $P$ is the survival/growth kernel from the IPM and $I$ is an identity kernel (analogous to an identity matrix). Fortunately, we can reuse our `N_kerns` code from above. However, it is important to remember that the $(1 - s)$ term represents all mortality pathways, and species may have more than one way to die (*e.g.* monocarpic perennials die through natural mortality as well as the flowering process). Therefore, we also want to check our kernel formulae and see if those include additional terms that may represent alternative mortality pathways. Additionally, we need to get the meshpoints which correspond to $\mathbf{i}$.

This time, we will use considerably more IPMs - we will get those from the `new_ipms` object we created earlier. However, we need to rebuild them with `return_all_envs = TRUE` so that we can access the vital rate function values.

```
arg_list <- lapply(names(new_ipms), function(x) list(return_all_envs = TRUE,
                                                     iterations      = 250)) %>%
  setNames(names(new_ipms))


new_ipms <- pdb_make_ipm(simple_det_list, addl_args = arg_list)

# Removing our these problem IPMs we identified before

keep_ind <- setdiff(names(new_ipms), c("dddd24", "dddd26",
                                       "dddd40", "dddd41"))

new_ipms <- new_ipms[keep_ind]

kernel_formulae(new_ipms)
```

```
## $aaaa34
## P: s * g
## F: r * fn * pE * d
## $aaaa36
## P: s * g
## F: r * fn * pE * d
## $aaa144
## P: s * g
## F: rep_p * es_p * sdl_s * n_infl * n_fl * n_seed
## $aaa227
## P: s * g
## F: rep_p * es_p * sdl_s * n_infl * n_fl * n_seed
## $aaa310
## P: s * g
## F: f_n * f_d
## $aaa323
## P: s * g
## F: p_es * sdl_size * n_seeds
## $aaa326
## P: (1 - p_fl) * s * g
```

```
## F: p_fl * fec1 * sdl_size * est_p
## $aaa341
## P: s * g
## F: Ep * Fp * Fs * Fd
## $aaa351
## P: s * g
## F: Pf * Nfruit * Nseeds * Pe * Fd
## $aaa385
## P: s * g
## F: f * fd
## $ccccc1
## P: s * g
## F: p_r * r_s * r_d * r_r
## $ddddd3
## P: s * g
## F: d * r
## $ddddd4
## P: s * g
## F: d * r
## $ddddd5
## P: s * g
## F: d * r
## $ddddd7
## P: s * g
## F: d * r
## $dddd10
## P: s * g * d_len
## F: d * r
## $dddd29
## P: s * g
## F: (s * r * pHS * d)/2
## $dddd30
## P: s * g
## F: s * r * pg * 0.5 * d
## $dddd33
## P: s * g
## F: rp * f * EP * d
## $dddd35
## P: s * g
## F: p_fertile * polyps * fec * p_est * d
## $dddd36
## P: s * g
## F: p_fertile * polyps * fec * p_est * d
## $dddd37
## P: s * g
## F: d * r
## $dddd39
## P: s * g
## F: f1 * f2 * f3 * f4 * f5 * d * 0.5
```

Upon further inspection, model ID `aaa326` contains a `(1 p_fl)` term, which represents mortality due to flowering. We need to include this in the calculations above. We do this like so:

$\bar{\omega} = (\mathbf{i} * (p_{fl} + (1 - p_{fl}) * (1 - s))N,$

and

$\Omega(z', z_0) = (p_{fl}(z') + (1 - p_{fl}(z')) * (1 - s(z')))N(z', z_0)$.

This can be summarized by saying "a plant dies with flowering probability $p_{fl}$, and if the plant does not flower $(1 - p_{fl})$, it dies with probability $1 - s$." We are now ready to proceed with our calculations!

```
N_kerns <- lapply(new_ipms, function(x) {

  P <- x$sub_kernels$P
  I <- make_i(P)

  solve(I - P)

})


surv_funs <- lapply(new_ipms, function(x) {
  vital_rate_funs(x)$P$s
})

p_flower <- vital_rate_funs(new_ipms$aaa326)$P$p_fl
```

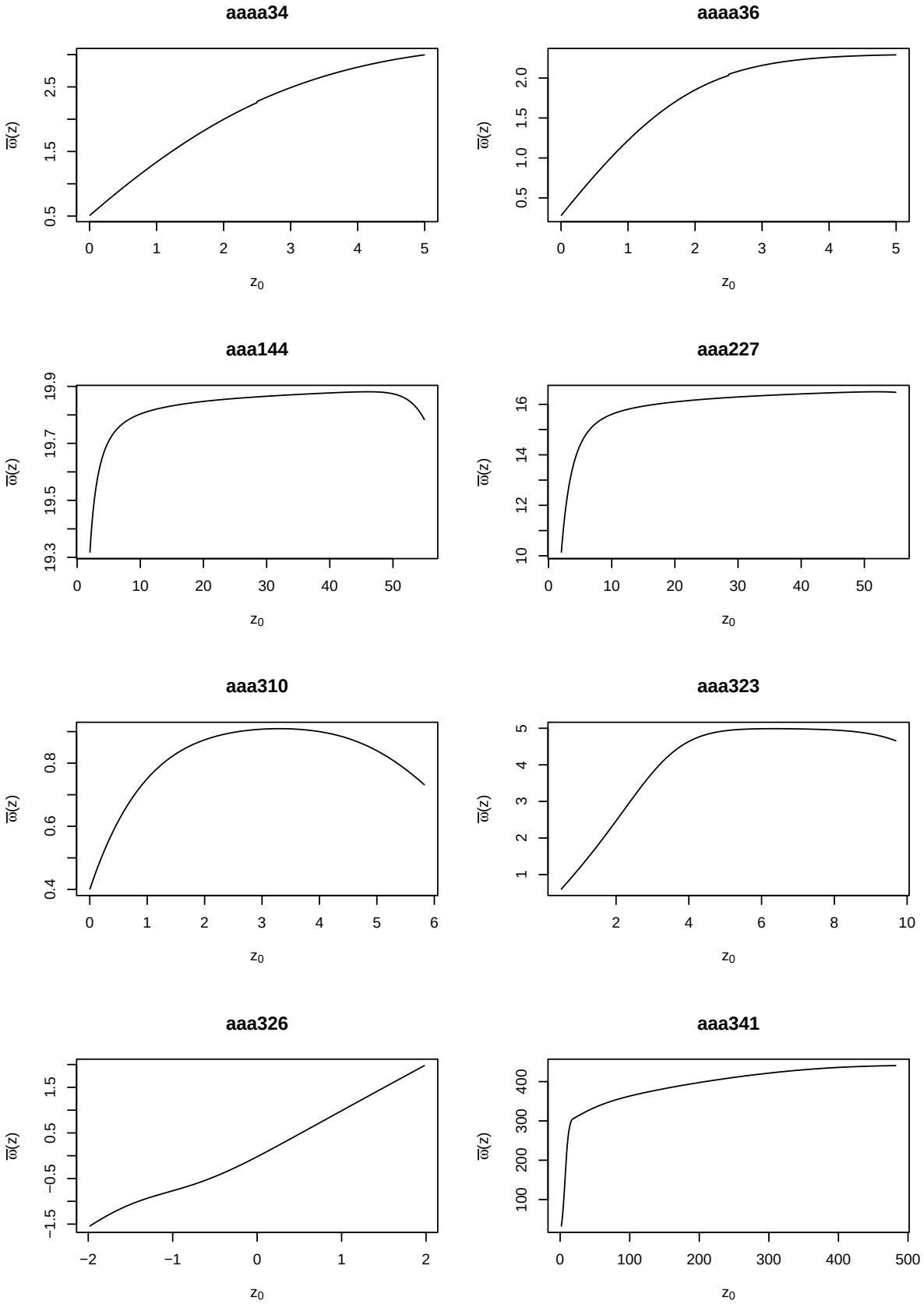
The **i** corresponds to the sizes in each IPM. Thus, we want to get the meshpoints, which we do with `int_mesh` like before. The `lapply(x[[2]])` extracts the value of `z_1`, as this is always the second entry in the list of 3 returned by `int_mesh` (and we do not need the `d_z` or `z_2` values).

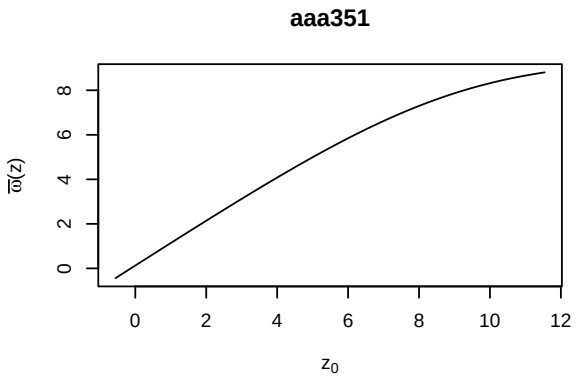
```
i_vals <- int_mesh(new_ipms, full_mesh = FALSE) %>%
  lapply(function(x) x[[2]])
```

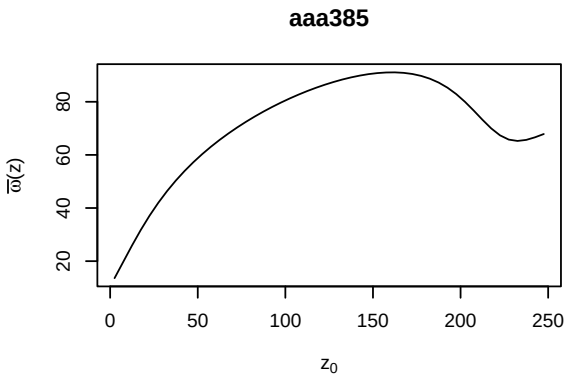
Now, we just have to implement the calculations:

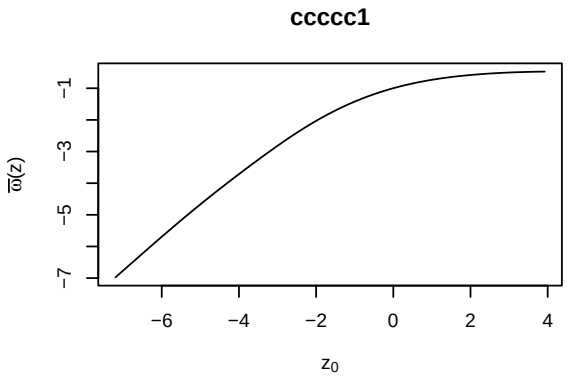
```
omega_bar_z <- lapply(seq_along(new_ipms),
                      function(index, i_vals, surv_funs, N_kerns, p_flower) {

                        id <- names(i_vals)[index]
                        i <- i_vals[[index]]

                        # The survival function is represented as a bivariate
                        # function in ipmr even though it is actually a univariate
                        # function of z. Thus, we need to pull out the
                        # univariate form of it to ensure we get the correct
                        # result from (1 - s) (_i.e._ a vector, not an array!).
                        # The correct univariate form is given by the rows,
                        # as every column contains the same values.

                        s <- surv_funs[[index]][1, ]
                        N <- N_kerns[[index]]

                        if(id == "aaa326") {
                          # Same indexing as the survival function
                          p_fl <- p_flower[1, ]

                          out <- (i * (p_fl + (1-p_fl) * (1 - s))) %*% N
                        } else {

                          out <- (i * (1 - s)) %*% N
```

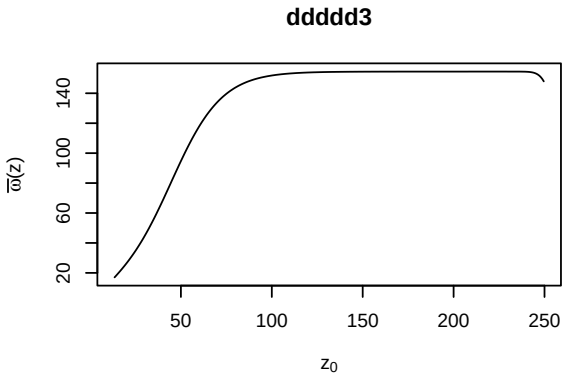
```
                        }

                        return(out)
                      },
                      i_vals    = i_vals,
                      surv_funs = surv_funs,
                      N_kerns   = N_kerns,
                      p_flower  = p_flower) %>%
  setNames(names(i_vals))



Omega_z0_z <- lapply(seq_along(new_ipms),
                     function(index, surv_funs, N_kerns, p_flower) {
                       id <- names(surv_funs)[index]

                       s <- surv_funs[[index]][1, ]
                       N <- N_kerns[[index]]

                       if(id == "aaa326") {
                         p_fl <- p_flower
                         out <- (p_fl + (1-p_fl) * (1 - s)) %*% N
                       } else {

                         out <- (1 - s) * N
                       }

                       return(out)
                     },
                     surv_funs = surv_funs,
                     N_kerns   = N_kerns,
                     p_flower  = p_flower) %>%
  setNames(names(i_vals))



par(mfrow = c(4, 2))

for(i in seq_along(omega_bar_z)) {

  plot(x    = i_vals[[i]],
       y    = as.vector(omega_bar_z[[i]]),
       type = "l",
       main = names(i_vals)[i],
       xlab = expression(paste(z[0])),
       ylab = expression(paste(bar(omega),"(z)")))
}
```

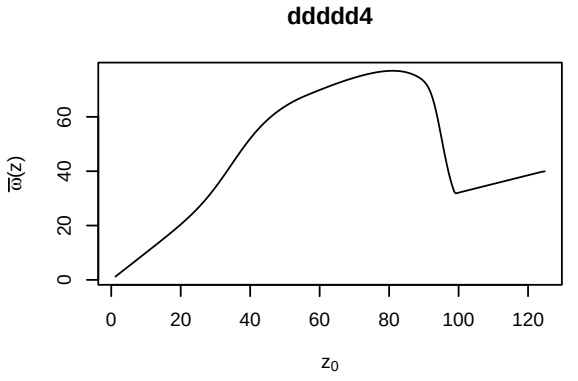

aaaa34
aaaa36
aaa144
aaa227
aaa310
aaa323
aaa326
aaa341
$\overline{\omega}(z)$
$z_0$

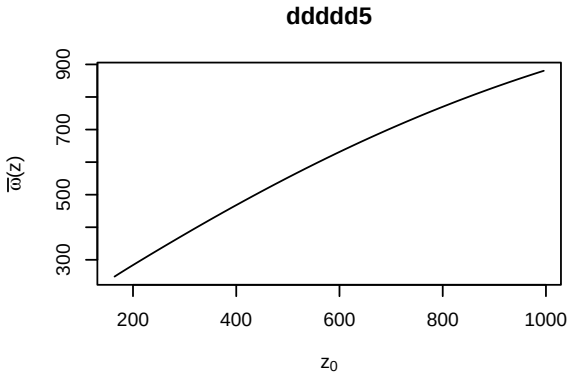

aaa351
$\overline{\omega}(z)$
0 2 4 6 8
0 2 4 6 8 10 12
$z_0$

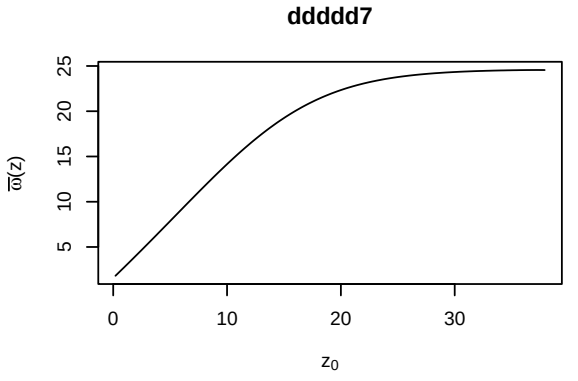

aaa385
$\overline{\omega}(z)$
20 40 60 80
0 50 100 150 200 250
$z_0$

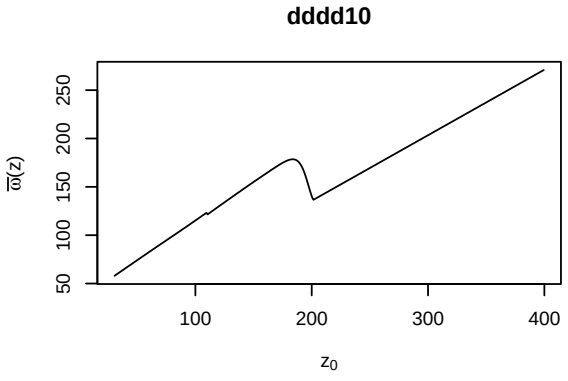

ccccc1
$\overline{\omega}(z)$
−7 −5 −3 −1
−6 −4 −2 0 2 4
$z_0$

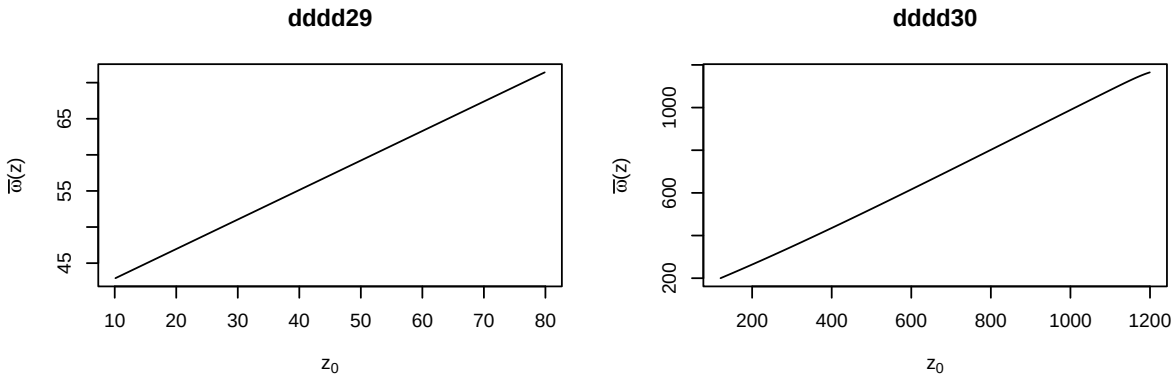

ddddd3
$\overline{\omega}(z)$
20 60 100 140
50 100 150 200 250
$z_0$


ddddd4
$\overline{\omega}(z)$
0 20 40 60
0 20 40 60 80 100 120
$z_0$

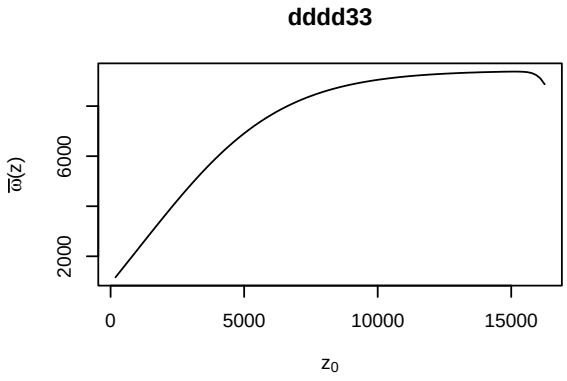

ddddd5
$\overline{\omega}(z)$
300 500 700 900
200 400 600 800 1000
$z_0$

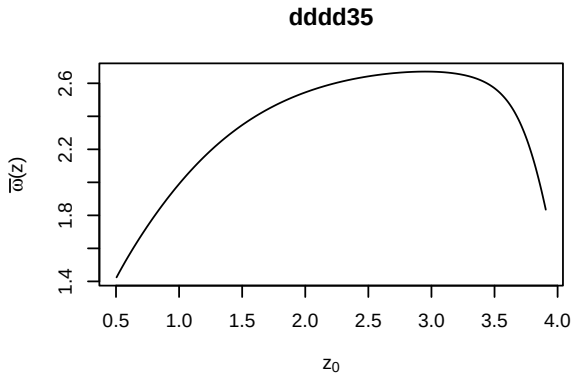

ddddd7
$\overline{\omega}(z)$
5 10 15 20 25
0 10 20 30
$z_0$

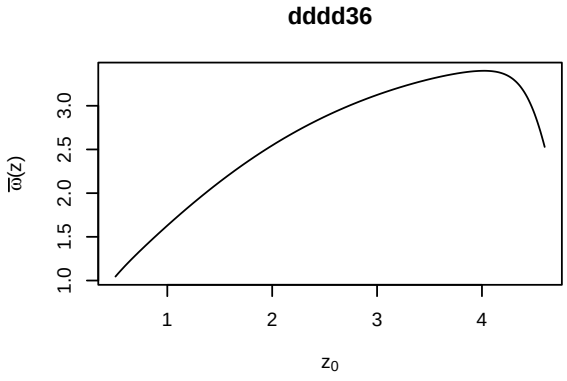

dddd10
$\overline{\omega}(z)$
50 100 150 200 250
100 200 300 400
$z_0$

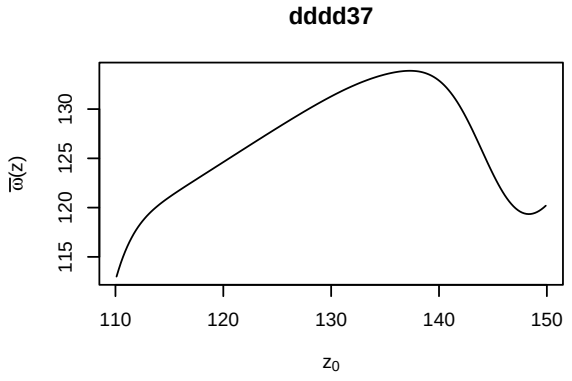

dddd29
$\overline{\omega}(z)$
45
55
65
10
20
30
40
50
60
70
80
$z_0$

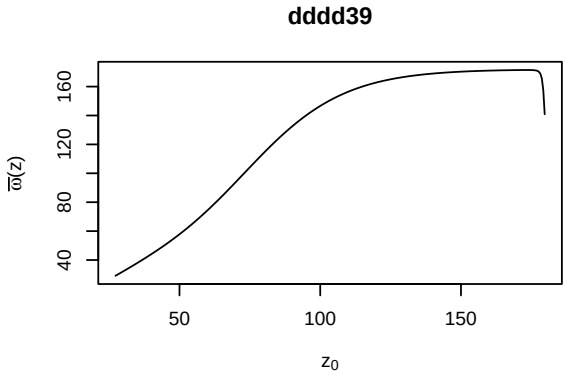

dddd30
$\overline{\omega}(z)$
200
600
1000
200
400
600
800
1000
1200
$z_0$


dddd33
$\overline{\omega}(z)$
2000
6000
0
5000
10000
15000
$z_0$


dddd35
$\overline{\omega}(z)$
1.4
1.8
2.2
2.6
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
$z_0$


dddd36
$\overline{\omega}(z)$
1.0
1.5
2.0
2.5
3.0
1
2
3
4
$z_0$


dddd37
$\overline{\omega}(z)$
115
120
125
130
110
120
130
140
150
$z_0$


dddd39
$\overline{\omega}(z)$
40
80
120
160
50
100
150
$z_0$

This is an interesting variety of relationships! Sometimes, it increases linearly, whereas other times we get a parabolic relationships. This is a good first pass on an analysis, though subsequent digging would likely reveal quirks in some of these models that we'd need to address more thoroughly. After these exercises, you should have the tools to do just that!

## Recap

We first showed how to subset PADRINO using the Metadata table, as well as a few others. Next, we demonstrated how to rebuild kernels from the database, as well as basic analyses such as deterministic population growth rates and perturbations. Next we moved into some examples of life cycle properties, such as recruit production and size at death. Along the way, we encountered some issues with data that required us to manipulate the underlying `proto_ipm`s. This is far from an exhaustive display of potential analyses. However, this case study should serve as a guide for posing questions and solving issues with PADRINO.

## Citations


1. Caswell, H. (2001) Matrix population models: construction, analysis, and interpretation, 2nd edn. Sunderland, MA: Sinauer Associates Inc
2. Wickham, H., François, R., Henry, L., & Müller, K. (2021). dplyr: A Grammar of Data Manipulation. R package version 1.0.6. https://CRAN.R-project.org/package=dplyr
3. Shelley, M. (1818) Frankenstein; or, the Modern Prometheus. London, Lackington, Hughes, Harding, Mayor, & Jones.
4. Ellner, S.P., Childs, D.Z., Rees, M. (2016) Data-driven modelling of structured populations: a practical guide to the integral projection model. Basel, Switzerland: Springer International Publishing AG

# Appendix 5: PADRINO Case Study 2

# Combined analyses with Rpadrino, ipmr, and other databases

## Overview

In this case study, we demonstrate how PADRINO can be used in conjunction with (1) your own unpublished data, and (2) external data repositories.

In many cases, we may wish to combine data that we've collected and not yet published with data from PADRINO. For example, we may wish to compare the vital rates or sensitivities of our (as-yet-unpublished) study system with those from published IPMs on e.g. similar species. We describe how to do this in Section 1 of this case study.

Using PADRINO in conjunction with external data sources presents unique opportunities to address synthetic questions in ecology, evolutionary biology, and conservation. External data sources could (though are not limited to) include climate data, species range distributions, phylogenies, or life tables. We describe how to use PADRINO in conjunction with BIEN in Section 2 of this case study.

## Combining your own data with PADRINO

In this section, we demonstrate how to combine data from PADRINO with a users own unpublished data. The first part of the code generates two IPMs - these are our "unpublished IPMs". The next section shows how to combine our unpublished IPMs with those stored in PADRINO, and how to perform simple analyses (i.e. calculate population growth rate) across the combined set of IPMs.

### Creating our own IPMs

The goal of this case study is to show how to combine data, and not necessarily how to use `ipmr`. `ipmr` is extensively documented on the project's website and in the publication describing the package. Therefore, the next few chunks of code assume you have already consulted these resources and will have a reasonable understanding of what's going on. If you have not already consulted these, please do so now.

Our first "homemade" IPM will be a general IPM. For now, we are only going to construct `proto_ipm` objects for each one of these homemade IPMs. Once we have our PADRINO IPMs selected, we will splice everything together and generate actual IPM objects.

```
# Loading Rpadrino automatically loads ipmr, so we do not need to load both.
library(Rpadrino)
```

```
## Warning: package 'ipmr' was built under R version 4.2.3
```

```
# Set up the initial population conditions and parameters.
# These are hypothetical values and do not correspond to any particular
# species.

data_list <- list(
  g_int     = 5.781,
  g_slope   = 0.988,
  g_sd      = 20.55699,
```

```
  s_int     = -0.352,
  s_slope   = 0.122,
  s_slope_2 = -0.000213,
  r_r_int   = -11.46,
  r_r_slope = 0.0835,
  r_s_int   = 2.6204,
  r_s_slope = 0.01256,
  r_d_mu    = 5.6655,
  r_d_sd    = 2.0734,
  e_p       = 0.15,
  g_i       = 0.5067,
  sb_surv   = 0.2
)

# Lower bound, upper bound, and number of meshpoints.
L <- 1.02
U <- 624
n <- 500

# Initialize a population vector. The continuous state will have 500 meshpoints,
# and we will pretend there's a seedbank.

init_pop_vec   <- runif(500)
init_seed_bank <- 20

my_general_ipm <- init_ipm(sim_gen = "general", di_dd = "di", det_stoch = "det") %>%
  define_kernel(
    name          = "P",
    formula       = s * g * d_ht,
    family        = "CC",
    g             = dnorm(ht_2, g_mu, g_sd),
    g_mu          = g_int + g_slope * ht_1,
    s             = plogis(s_int + s_slope * ht_1 + s_slope_2 * ht_1^2),
    data_list     = data_list,
    states        = list(c('ht')),
    uses_par_sets = FALSE,
    evict_cor     = TRUE,
    evict_fun     = truncated_distributions('norm',
                                            'g')
  ) %>%
  define_kernel(
    name          = "go_discrete",
    formula       = r_r * r_s * d_ht,
    family        = 'CD',
    r_r           = plogis(r_r_int + r_r_slope * ht_1),
    r_s           = exp(r_s_int + r_s_slope * ht_1),
    data_list     = data_list,
    states        = list(c('ht', "b")),
    uses_par_sets = FALSE
  ) %>%
  define_kernel(
    name          = "stay_discrete",
    family        = "DD",
```

```
    formula       = sb_surv * (1 - g_i),
    data_list     = data_list,
    states        = list(c("b")),
    uses_par_sets = FALSE
  ) %>%
  define_kernel(
    name          = 'leave_discrete',
    formula       = e_p * g_i * r_d * d_ht,
    r_d           = dnorm(ht_2, r_d_mu, r_d_sd),
    family        = 'DC',
    data_list     = data_list,
    states        = list(c('ht', "b")),
    uses_par_sets = FALSE,
    evict_cor     = TRUE,
    evict_fun     = truncated_distributions('norm',
                                            'r_d')
  ) %>%
  define_impl(
    list(
      P              = list(int_rule    = "midpoint",
                            state_start = "ht",
                            state_end   = "ht"),
      go_discrete    = list(int_rule    = "midpoint",
                            state_start = "ht",
                            state_end   = "b"),
      leave_discrete = list(int_rule    = "midpoint",
                            state_start = "b",
                            state_end   = "ht"),
      stay_discrete  = list(int_rule    = "midpoint",
                            state_start = "b",
                            state_end   = "b")
    )
  ) %>%
  define_domains(
    ht = c(L, U, n)
  ) %>%
  define_pop_state(
    pop_vectors = list(
      n_ht = init_pop_vec,
      n_b  = init_seed_bank
    )
  )
```

Our next IPM will be a simple one:

```
# Another hypothetical model. These parameters also do not correspond to any
# species.

my_data_list = list(s_int     = -2.2,
                    s_slope   = 0.25,
                    g_int     = 0.2,
                    g_slope   = 0.99,
                    sd_g      = 0.7,
                    r_r_int   = 0.003,
```

```
                    r_r_slope = 0.015,
                    r_s_int   = 0.45,
                    r_s_slope = 0.075,
                    mu_fd     = 2,
                    sd_fd     = 0.3)

my_simple_ipm <- init_ipm(sim_gen = "simple",
                          di_dd   = "di",
                          det_stoch = "det") %>%
  define_kernel(
    name      = "P_simple",
    family    = "CC",
    formula   = s * G,
    s         = plogis(s_int + s_slope * dbh_1),
    G         = dnorm(dbh_2, mu_g, sd_g),
    mu_g      = g_int + g_slope * dbh_1,
    data_list = my_data_list,
    states    = list(c('dbh')),
    evict_cor = TRUE,
    evict_fun = truncated_distributions(fun    = 'norm',
                                        target = 'G')
  ) %>%
  define_kernel(
    name      = 'F_simple',
    formula   = r_r * r_s * r_d,
    family    = 'CC',
    r_r       = plogis(r_r_int + r_r_slope * dbh_1),
    r_s       = exp(r_s_int + r_s_slope * dbh_1),
    r_d       = dnorm(dbh_2, mu_fd, sd_fd),
    data_list = my_data_list,
    states    = list(c('dbh')),
    evict_cor = TRUE,
    evict_fun = truncated_distributions(fun    = 'norm',
                                        target = 'r_d')
  ) %>%
  define_impl(
    make_impl_args_list(
      kernel_names = c("P_simple", "F_simple"),
      int_rule     = rep("midpoint", 2),
      state_start  = rep("dbh", 2),
      state_end    = rep("dbh", 2)
    )
  ) %>%
  define_domains(
    dbh = c(0,
            50,
            100
    )
  ) %>%
  define_pop_state(
    n_dbh = runif(100)
  )
```

```
my_ipm_list = list(ipm_1 = my_general_ipm, ipm_2 = my_simple_ipm)
```

### Combining user-defined and PADRINO-defined IPMs

Next, we will create a list of `proto_ipm` objects from PADRINO, and then put everything together. For simplicity, we will select a small number of plant species. The `pdb` object is contained within the `Rpadrino` package. It is not a complete version of PADRINO. We will use the complete data set in the next section, accessed with `pdb_download()`.

```
data(pdb)

id_index <- c(
  paste0( "aaaa", c(34, 55)),
  paste0("aaa", c(310, 312, 339, 341, 353, 388))
)

small_db <- pdb_subset(pdb, id_index)
```

Next, we need to create a list that holds both the PADRINO IPMs and the ones we created above. After that, we can call `pdb_make_ipm()` on the combined data set, and voila! We have our database IPMs and our own homemade ones.

```
proto_list <- c(
  pdb_make_proto_ipm(small_db),
  my_ipm_list
)
## 'ipm_id' aaa310 has the following notes that require your attention:
## aaa310: 'Geo and time info retrieved from COMPADRE (v.X.X.X.4)'
## 'ipm_id' aaa388 has the following notes that require your attention:
## aaa388: 'Same data as AAA388. State variable Height (Cm)'
```

Great! In that single step, we combined PADRINO IPMs with our own IPMs. Because these are all in the `proto_ipm` format, we do not need to think about technical differences between each type - we can use the exact same toolbox for analyzing both! Let's build the IPM objects and calculate deterministic per-capita growth rates!

```
ipm_list   <- pdb_make_ipm(proto_list)
lambdas    <- lambda(ipm_list)
```

We could now proceed with any further analyses just as we did in the case study 1. Since those types of analyses are already covered by the previous case study, we will move on to combining PADRINO data with information from other databases.

## Extending analyses with other databases

Here, we show how to combine data from PADRINO with data from other external sources. Specifically, we show how to combine with data from COMPADRE MPM database and BIEN , a database containing the spatial distribution and phylogenies of many plant species. We then demonstrate how we can use these combined datasets to address the question: "How do population growth rates vary by the distance of the studied population from the known range edge of that species?"

Given the question we posed above, we need to get range maps for each species and the per-capita growth rate for some populations. For the former, we will use range maps from BIEN.For the latter, we will augment

PADRINO with data from COMPADRE. This analysis will not be the most complete - it is intended to demonstrate the steps for combining data, not to make a scientific point. With that in mind, let's dive in!

## Required packages

BIEN allows users to download range maps programmatically from their database using the BIEN R package. You can install that from CRAN using the chunk below. we will also use `Rcompadre`, `mgcv`, `ggplot2`, `sf`, and `dplyr` to work with the data, so you'll need to install those as well. Finally, we need to install `rgeos` so that we can slightly modify the function that downloads range maps from BIEN (the current version of this function does not work with the newest version of `sf` and `sp`).

```
install.packages(c("BIEN", "ggplot2", "sf", "rgeos" ,
                   "dplyr", "Rcompadre", "mgcv"))
```

After that, we have to load them:

```
library(BIEN)
library(rgeos)
library(methods)
library(ggplot2)
library(sf)
library(dplyr)
library(Rcompadre)
library(Rpadrino)
library(mgcv)
```

## Data identification

BIEN allows us to programmatically query the database and retrieve all species names for which there is a range map. we will load that, then load COMPADRE and PADRINO, and see how much overlap there is.

```
bien_rng_spps <- BIEN_ranges_list()
pdb           <- pdb_download(save = FALSE)
cdb           <- cdb_fetch("compadre")
```

```
## This is COMPADRE version 6.23.5.0 (release date May_06_2023)
## See user agreement at https://compadre-db.org/Help/UserAgreement
## See how to cite at https://compadre-db.org/Help/HowToCite
```

```
# Insert an underscore to make sure name format matches between COMPADRE,
# PADRINO, and BIEN

cdb_spp <- gsub(" ", "_", cdb$SpeciesAccepted)

pdb_spp <- pdb_species_accepted(pdb)
```

Nice! We have 509 overlapping species between COMPADRE/PADRINO and BIEN's range maps. This next chunk determines which species from PADRINO and COMPADRE have range maps available in BIEN:

```
all_spp <- unique(c(cdb_spp, pdb_spp))
pos_spp <- all_spp[all_spp %in% bien_rng_spps$species]

pdb_rng_spp <- unique(pdb_spp[pdb_spp %in% pos_spp])
cdb_rng_spp <- unique(cdb_spp[cdb_spp %in% pos_spp])
```

## Subsetting

We probably should not use all of these, as those calculations would take quite some time for a tutorial, so we will select a subset. we will take the species for which the demographic data are from North America. For PADRINO, we need to find their `ipm_ids`, and then pass those into `pdb_subset()`. For COMPADRE, we can just use `dplyr` verbs as if we were working with a `data.frame`.

```
# First, we will create a vector of ipm_id's which meet the following requirements:
# 1. They have range maps in BIEN (species_accepted %in% pdb_rng_spp)
# 2. The model is from data collected in North America (continent == "n_america")
# 3. The data are from unmanipulated populations (treatment == "Unmanipulated")

pdb_ids     <- pdb$Metadata$ipm_id[pdb$Metadata$species_accepted %in% pdb_rng_spp &
                                   pdb$Metadata$continent == "n_america" &
                                   pdb$Metadata$treatment == "Unmanipulated"]

use_pdb     <- pdb_subset(pdb, pdb_ids)

# For COMPADRE, we have to first replace "_" in the species names with a space.
# Then we can use filter() syntax to subset COMPADRE to the species we want.

cdb_rng_spp_f <- gsub("_", " ", cdb_rng_spp)

use_cdb <- filter(cdb,
                  SpeciesAccepted %in% cdb_rng_spp_f &
                  Continent == "N America" &
                  MatrixTreatment == "Unmanipulated")
```

## Check data quality

PADRINO data is validated before it is uploaded to ensure the IPM behaves as the published version behaves. There are additional checks you might want to perform on your own, and those depend on the subsequent anaylsis. Case study 1 shows an example of a singular kernel creating some biologically impossible results. However, there are not built-in functions in `Rpadrino` yet to assist with this. Therefore, it is usually a good idea to check the original publications just to be sure there are not caveats to the model that the authors have raised. We can find the citations using `pdb_citation()` and `pdb_report()`. `pdb_citation()` returns a character vector of citations in APA style, whereas `pdb_report()` generates an RMarkdown report based on the information in the database.

```
cites <- pdb_citations(use_pdb)

pdb_report(use_pdb)
```

we will also want to check COMPADRE for some common data issues using the `cdb_flag()` function. This is documented much more thoroughly in the `Rcompadre` package website. For simplicity, we will just use ones which do not raise any flags, as fixing issues with COMPADRE data is beyond the scope of this case study. Furthermore, we will subset out the mean matrices, as we want to work with individual transitions.

```
cdb_f <- cdb_flag(use_cdb)

use_cdb <- filter(cdb_f, !check_NA_A & !check_NA_U & !check_NA_F & !check_NA_C&
                         !check_zero_U & !check_singular_U & check_component_sum &
                         check_ergodic & check_irreducible & check_primitive &
                         check_surv_gte_1 & MatrixComposite == "Individual")
```

### Data transformation

Next, we need to do a bit of data wrangling. From PADRINO, we only need the `ipm_id` and species names for plotting and analyzing, so we will just grab those from the metadata table. We're going to create an `sf` object for this data using the coordinates stored in the `"lat"` and `"lon"` columns of the metadata. `sf` provides a standardized interface for dealing with multiple types of spatial data, and also plays nicely with `dplyr`, which makes managing data much easier. The `st_as_sf()` function handles the conversion for us.

```
# Create a standard data.frame with ipm_id, species, lat+lon data from PADRINO

temp_coords <- use_pdb$Metadata %>%
  select(ipm_id, species_accepted, lat, lon)

# This next bit does the following:
# 1. Creates a data.frame from COMPADRE data with the same columns as PADRINO.
# 2. Changes the names so they match the PADRINO version.
# 3. Combines the COMPADRE and PADRINO versions.
# 4. Eliminates studies that do not have complete latitude/longitude information.

temp_db <- use_cdb@data %>%
  select(MatrixID, SpeciesAccepted, Lat, Lon) %>%
  setNames(names(temp_coords)) %>%
  rbind(temp_coords) %>%
  .[complete.cases(.), ]


# Finally, create an 'sf' object with the combined coordinates from COMPADRE and
# PADRINO

study_coords <- st_as_sf(temp_db,
                         coords = c("lon", "lat"),
                         crs = "WGS84")
```

### Querying BIEN

Now that we have our final species list, we're going to download the range maps for each species using the `BIEN_ranges_load_species()` function, and then convert that into an `sf` object which will make subsequent analysis and plotting easier. Below, we define a modified version of the `BIEN_ranges_load_species()` function because the current package's version fails with the newest version of `sf` and `sp` installed.

```
BIEN_ranges_load_species <- function(species,
                                     ...) {

  species <- gsub(" ", "_", species)

  # set the query
  query <- paste(
    "SELECT ST_AsText(geom),species,gid FROM ranges WHERE species in (",
    paste(shQuote(species, type = "sh"), collapse = ', '),
    ") ORDER BY species ;")

  # create query to retrieve
  df <- BIEN:::.BIEN_sql(query, ...)
```

```
  if(length(df) == 0){

    message("No species matched")

  }else{

    poly <- list()
    for(l in 1:length(df$species)){
      Species<-df$species[l]
      #sp_range<-readWKT(df$st_astext[l])
      poly[[l]]<-readWKT(df$st_astext[l], p4s = st_crs(4326)[[2]])

      #assigns a unique ID to each species' polygon
      slot(object = poly[[l]]@polygons[[1]],
                    name = "ID") <- as.character(df$gid[l])

    } #for species in df loop


  }

  poly <- SpatialPolygons(unlist(lapply(poly, function(x) x@polygons)))
  poly <- SpatialPolygonsDataFrame(Sr = poly,data = df['species'],match.ID = FALSE)
  poly@proj4string <- CRS(projargs = st_crs(4326)[[2]])
  return(poly)

}
```

After converting the range maps to an `sf` object, we also need to create a different version of the polygons that are a set of lines representing the edges. This will allow us to quickly calculate the distance between our study points and the edge of the range. `st_cast()` handles this conversion for us.

```
# The next piece:
# 1. Downloads range maps
# 2. Converts each range map into an 'sf' object
# 3. Resolves issues that arise in the conversion process (e.g. self intersections)
# 4. Sorts the resulting 'sf' object alphabetically on species

rng_maps <- BIEN_ranges_load_species(study_coords$species_accepted) %>%
  st_as_sf() %>%
  st_make_valid() %>%
  arrange(species)

# We need to create a copy of each range map that, rather than a polygon format,
# is a line format. This enables us to use functions to compute distance to edge
# much more easily.

line_maps <- st_cast(rng_maps, "MULTILINESTRING")

# Put the "_"'s back into study_coords so that we can match all names later on.

study_coords$species_accepted <- gsub(" ", "_", study_coords$species_accepted)

# Finally, sort study_coords alphabetically as well. Now, the species_accepted
```

```
# column should be identical to the species column in rng_maps. This will be
# important in the next step.

study_coords <- arrange(study_coords, species_accepted)  %>%
  .[!duplicated(.$species_accepted), ]
```

## Compute distance from edges

Ok, we're finally ready to compute the distance from each study site to the range edge. We're going to use the `st_distance()` function for this. This finds the minimum distance between the first and second arguments and computes a matrix for all possible combinations. It will ignore the fact that sometimes the closest edge is an ocean (which our species cannot grow in). However, working out how to improve that calculation is a problem for another day!

We start by extracting a distance matrix and taking the diagonal. The diagonal represents the shortest distance between our species study site and the edge of the polygon of its range map (NB: This only works because we sorted each object alphabetically ahead of time!). Next, we add in the species name information and set the data frame's names to something useful. Finally, we will convert the distances to kilometers.

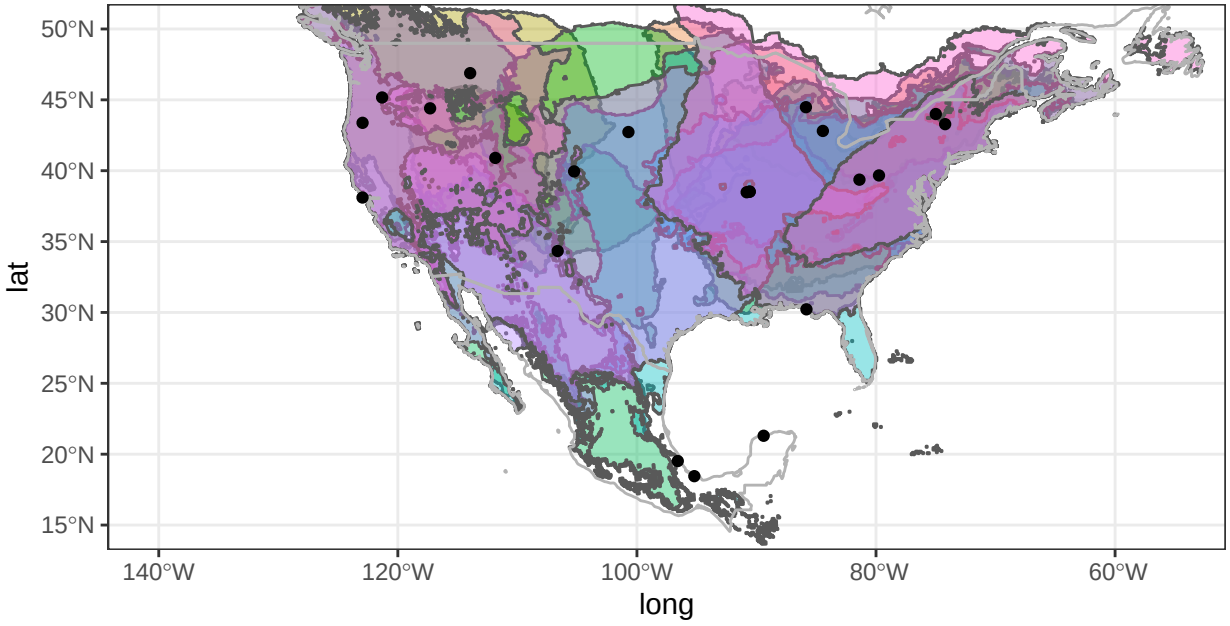
```
# Quickly check to make sure all of our species line up positionally.
# If not, we'd need to make sure they do, otherwise it will be difficult
# to extract the distances from the distance matrix we are about to compute!

stopifnot(all(line_maps$species == study_coords$species_accepted))

# This next piece does:
# 1. Computes distances between all pairs of points and line objects
# 2. Extracts the diagonal of the distance matrix. This represents the distance
# from a species' study site to the edge of its range (again, only because
# we sorted study_coords and line_maps alphabetically).
# 3. converts this information to a data.frame
# 4. Adds the species names to that data.frame
# 5. Sets column names for that data.frame
# 6. Converts the distance to km

dist_from_edge <- st_distance(study_coords, line_maps) %>%
  diag() %>%
  data.frame() %>%
  cbind(study_coords$species_accepted, .) %>%
  setNames(c("species", "distance_in_meters")) %>%
  mutate(
    distance_in_km = round(as.numeric(distance_in_meters) / 1e3, 2)
  )
```

## Visualize our dataset

we will plot our range maps with the study sites overlaid on them using `ggplot2`. `ggplot2` has built in `geoms` designed to handle `sf` objects, which will make our lives much easier!

```
world     <- map_data("world")
n_america <- filter(world, region %in% c("USA", "Canada", "Mexico"))

ggplot(rng_maps) +
```

```
  geom_sf(aes(fill = species), alpha = 0.4) +
  geom_polygon(data = n_america, aes(x = long, y = lat, group = group),
               inherit.aes = FALSE,
               color = "grey70",
               fill = NA) +
  geom_sf(data  = study_coords) +
  coord_sf(xlim = c(-140, -55),
           ylim = c(15, 50)) +
  theme_bw() +
  theme(
    legend.position = "none"
  )
```




Already, we can see that our range maps do not perfectly align with the COMPADRE and PADRINO population coordinates. We can check and see which study populations are actually contained by their range map like so:

```
# Notice that now we are using the POLYGONS object (rng_maps) as opposed to the
# to the LINESTRING version (line_maps).

covered_ind <- st_covered_by(study_coords,
                             st_make_valid(rng_maps),
                             sparse = FALSE) %>%
  diag()

# Print studies not covered by BIEN range map
```

```
study_coords[!covered_ind , ]
```

```
## Simple feature collection with 6 features and 2 fields
## Geometry type: POINT
## Dimension:     XY
## Bounding box:  xmin: -122.9567 ymin: 18.45 xmax: -85.82 ymax: 45.16667
## Geodetic CRS:  WGS 84
## # A tibble: 6 x 3
##   ipm_id species_accepted                  geometry
##   <chr>  <chr>                          <POINT [°]>
## 1 241875 Astragalus_tyghensis   (-121.3167 45.16667)
## 2 242052 Calathea_ovandensis          (-95.2 18.45)
## 3 239708 Euphorbia_telephioides    (-85.82 30.21972)
## 4 244332 Lupinus_tidestromii    (-122.9567 38.10861)
## 5 247007 Mammillaria_gaumeri           (-89.4 21.3)
## 6 241182 Tillandsia_deppeana    (-96.58333 19.51667)
```

These are COMPADRE matrices. Rather than try to figure out what's going on, we will just drop those out of our analysis.

```
study_coords <- study_coords[covered_ind, ]
```

## Compute lambdas for each type of model

Great! The next step is to generate and then join our lambda values with the distance information. This is a two-step process. First, we will build our PADRINO IPMs:

```
# Extract PADRINO IDs - we do not want to give COMPADRE ones to PADRINO machinery!

pdb_ids <- study_coords$ipm_id[study_coords$ipm_id %in% pdb$Metadata$ipm_id]

# Construct the proto_ipm list
proto_list <- pdb_make_proto_ipm(
  use_pdb,
  pdb_ids
)
```

```
## 'ipm_id' aaa310 has the following notes that require your attention:
## aaa310: 'Geo and time info retrieved from COMPADRE (v.X.X.X.4)'
```

```
## 'ipm_id' aaa329 has the following notes that require your attention:
## aaa329: 'Based on IPM from Rose Ecology 2005; The GPS coordinates were approximated
## to the closest geographic location described in the reference'
```

```
## 'ipm_id' aaa385 has the following notes that require your attention:
## aaa385: 'Same data as AAA385. State variable Height (Cm)'
```

```
# Construct the IPMs

ipm_list <- pdb_make_ipm(proto_list)

# Some IPMs may have many values for lambda, because they were constructed from
# vital rate models that have time varying parameters (e.g. random effects for
# year). we will need to account for this. We need to convert those from a list to
# a data.frame for modeling, and need to keep track of which lambda belongs to
# which ID. The loop below will correctly format this.
```

```
lambdas <- lambda(ipm_list, type_lambda = "last")

temp <- data.frame(ipm_id = NA,
                   lambda = NA)

for(i in seq_along(lambdas)) {

  # Create a temporary object to store lambda values and ipm_id's. Each lambda
  # value will have its own row, with the corresponding ipm_id next to it. This
  # will help us track which value belongs to which model.

  temp_2 <- data.frame(ipm_id = names(lambdas)[i],
                       lambda = lambdas[[i]])


  # I don't normally recommend using rbind in a 'for' loop, but there aren't many
  # iterations here, so we will not worry about the memory footprint
  temp <- rbind(temp, temp_2)

}

# Remove the dummy row of NAs

temp <- temp[-1, ]
```

Next, we will get our COMPADRE lambdas, and stick them back in with the PADRINO lambdas.

```
use_cdb <- filter(use_cdb, MatrixID %in% study_coords$ipm_id)

matAs   <- matA(use_cdb)

use_cdb@data$lambda <- vapply(matAs,
                              function(x) Re(eigen(x)$values[1]),
                              numeric(1L))

cdb_lambda <- use_cdb@data %>%
  select(MatrixID, lambda) %>%
  setNames(c("ipm_id", "lambda"))

all_lambdas <- rbind(temp, cdb_lambda)
```

Finally, we need to join lambda values with coordinate data set to recover the species names, and then use those to join with the distance from edge object. Once that's done, we can plot everything!

```
all_lambdas <- left_join(all_lambdas, study_coords, by = "ipm_id") %>%
  select(-geometry)

all_data <- left_join(all_lambdas, dist_from_edge,
                      by = c("species_accepted" = "species"))
```

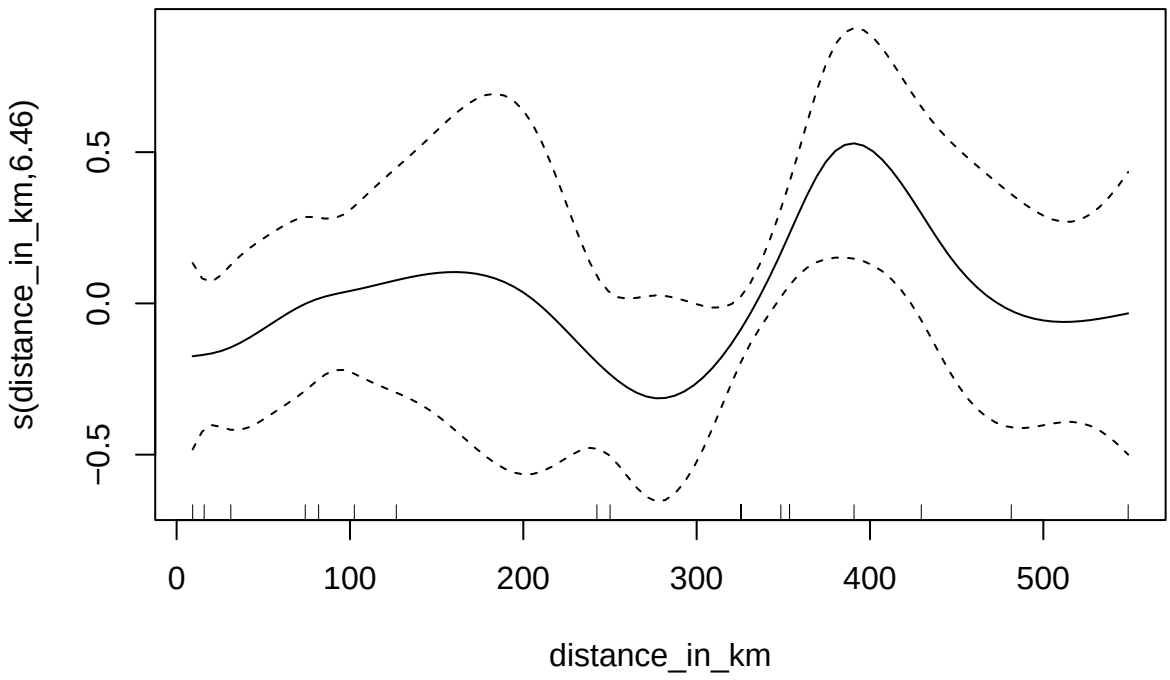
## Regression modelling

We're ready to plot and analyze the data. GAMs (Wood 2011) are a great way to spot general trends in data, so we will use those.

```
lambda_by_dist <- gam(lambda ~ s(distance_in_km, bs = "cs"),
                      data = all_data,
                      family = Gamma(link = "identity"))

summary(lambda_by_dist)
```

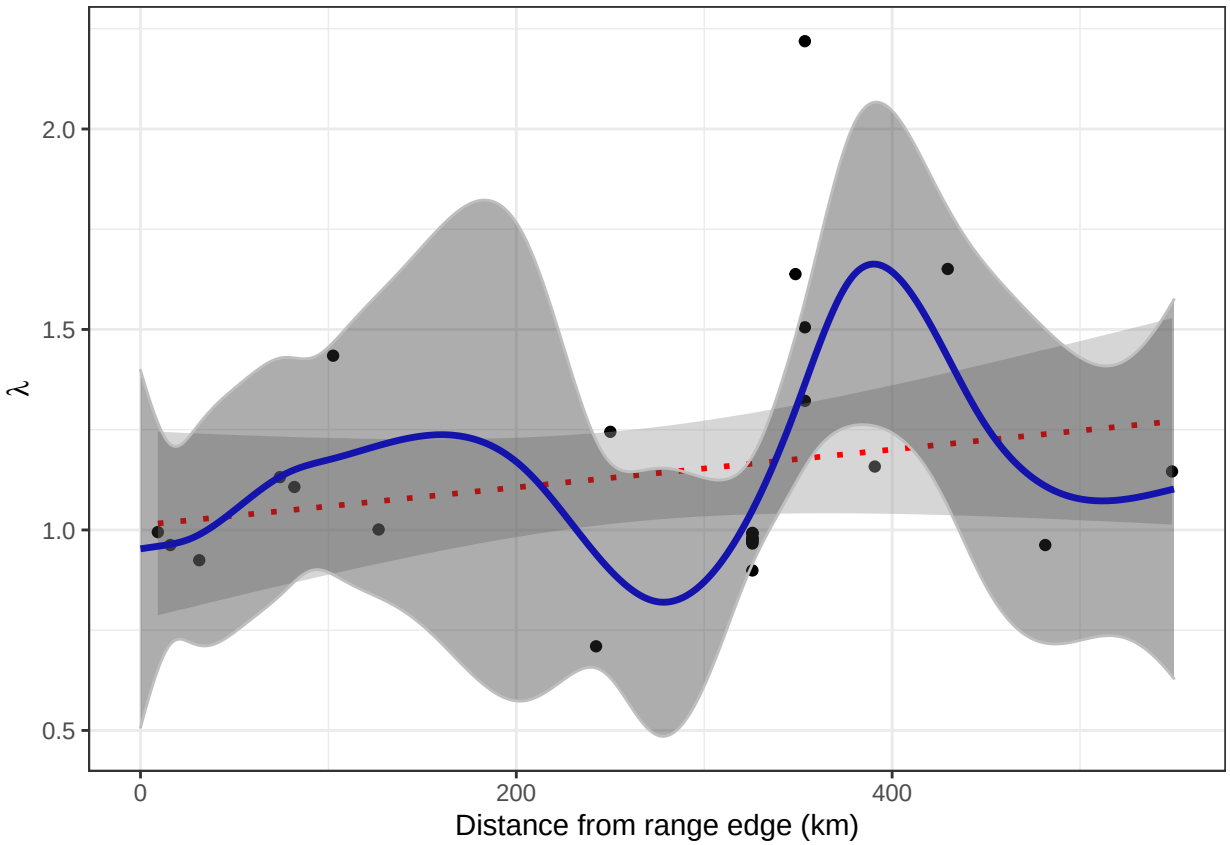
```
##
## Family: Gamma
## Link function: identity
##
## Formula:
## lambda ~ s(distance_in_km, bs = "cs")
##
## Parametric coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  1.13381    0.04843   23.41 9.22e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##                       edf Ref.df     F p-value
## s(distance_in_km) 6.463      9 1.305   0.135
##
## R-sq.(adj) =  0.197   Deviance explained = 47.5%
## GCV = 0.059745  Scale est. = 0.048724  n = 27
```

```
plot(lambda_by_dist)
```




```
preds <- cbind(data.frame(predict(lambda_by_dist,
                                  data.frame(distance_in_km = seq(0, 550, 1)),
                                  type = "response",
                                  se.fit = TRUE)),
               x = seq(0, 550, 1)) %>%
  mutate(upper = fit + se.fit * 1.96,
         lower = fit - se.fit * 1.96)



ggplot(all_data, aes(x = distance_in_km, y = lambda)) +
  geom_point() +
  geom_smooth(method = "glm",
              formula = y ~ x,
              method.args = list(family = Gamma("identity")),
              color = "red",
              linetype = "dotted") +
  geom_line(data = preds,
            aes(x = x, y = fit),
            inherit.aes = FALSE,
            color = "blue",
            size = 1.2) +
  geom_ribbon(data = preds,
              aes(x = x, ymin = lower, ymax = upper),
              color = "grey",
              alpha = 0.4,
```

```
            inherit.aes = FALSE) +
theme_bw() +
xlab("Distance from range edge (km)") +
ylab(expression(lambda))
```




There is a positive trend in range centrality and species performance (red line), and the GAM is likely overfit (blue line). There is a lot of residual variance, and we can certainly find better ways to model this phenomenon, but this is a good start for an exploratory analysis. We will leave the further analyses as an exercise to you!

## Recap

We have shown how to combine PADRINO data with user-defined IPMs as well as join it with information from other databases. This is hardly a comprehensive overview of PADRINO's applications - there are many other uses and databases one could combine PADRINO with. It is our hope that this and the previous case study provide a general guide to the considerations and steps one needs to take when using this data!

## Citations

1. Wood, S.N. (2011) Fast stable restricted maximum likelihood and marginal likelihood estimation of semiparametric generalized linear models. Journal of the Royal Statistical Society (B) 73(1):3-36

2. Maitner, B., Boyle B., Casler N., Condit R., Donoghue J., Duran S.M., *et al.* (2017) The bien r package: A tool to access the Botanical Information and Ecology Network (BIEN) database. Methods in Ecology and Evolution 9(2): 373-379. https://doi.org/10.1111/2041-210X.12861

# Appendix 6: Supplementary Information for Chapter 3

# Introduction to PADRINO

PADRINO v0.0.1 consists of 10 tables (Table 1, Figure S1). In this first version, PADRINO currently contains 280 IPMs from 40 peer-reviewed publications that consider 14 animal and 26 species (Table 1 main text). However, we highlight that PADRINO is under active development, and we continue to digitize studies for release in future versions. These tables form a database, with tables linked using a common column across all tables: `ipm_id`. The scope of each `ipm_id` is determined by the way that an IPM is parameterized. IPMs that characterize the same species across, for example, many years or sites, with the same functional form, are included under a single `ipm_id`. For instance, a growth model that includes a random intercept for different years could be used to generate many unique projection kernels. These are stored under a single `ipm_id` because the functional form of the IPM is identical for each year, and only the parameter values change. One exception to this grouping rule is when the sites (*i.e.* where the raw data are reported to have come from) are far enough apart that separate sets of GPS coordinates are used to describe them. These IPMs are split into separate `ipm_id`s so that the spatial distinctions are preserved, which facilitates matching PADRINO data with, for example, gridded environmental data (*e.g.* Compagnoni et al. 2021b, Case Study 2).

Finally, there are two important details potential users should be aware of. The first detail is that PADRINO provides IPMs *as they are published following peer review*. We do not alter these IPMs when digitizing them, except to correct typographical errors that may have found their way into the peer-reviewed publication. The second detail is that PADRINO does not store any raw data used to create the IPMs. Users should be aware of these, and we encourage all users to consult and cite the original publications of each IPM before including it in an analysis.

## The Digitization Process

The IPM digitization process begins when a peer-reviewed paper containing an IPM is published. We have set alerts for the following keyword searches: "Integral Projection Model OR IPM OR sensitivit* OR elasticit* OR Vital rate OR LTRE". This automatic weekly search is run on Google Scholar and Scopus, and resulting hits are examined manually to find publications that contain an IPM. Once a paper containing an IPM is identified, we extract five types of metadata: taxonomic information (*e.g.* species names, functional groups), publication information (*e.g.* authors, complete citation, year of publication), temporal metadata (*e.g.* study duration, data collection beginning and ending months and years), spatial metadata (*e.g.* latitude/longitude, ecoregion), and model specific metadata (*e.g.* experimental treatments applied, density-(in)dependent). Table S1 contains a complete description of the metadata table in PADRINO.

Following the metadata digitization, we extract functional forms of each sub-kernel, vital rate function, and how the environment varies (if applicable). The functional forms of each component of the model are expressed in the syntax introduced in the main text. Finally, we extract all of the parameter values, as well as information on the range of values each trait can take on and how they are numerically approximated (*i.e.* integration rules). The parameter values and integration information are then substituted for symbol names when the user requests a built model. For example, in *Rpadrino*, the `Norm(mu_g, sd_g)` from the main text would be translated to `dnorm(z_2, mu_g, sd_g)`.

Often times, not all of the required information is present in the publication or its supplementary materials. Therefore, we often contact authors to request the required information and/or ask for clarification. We also extract a target value for the data validation step (see next section), so that we can ensure that released data really does replicate the published IPM. A complete guide to our digitization process and documentation of the database syntax is publicly available on PADRINO's webpage (https://padrinoDB.github.io/Padrino/).

## Data Validation and Reproducibility

The PADRINO IPM Database has automated testing built into the data release process. All IPMs are checked to ensure they recover the behavior of the published version prior to release. In most cases, validation consists

of reproducing the kernel-specific asymptotic population growth rate ($\lambda$) to within $\pm$ 0.03 of the published $\lambda$ value in the source publication. It is worth noting that this margin of error is considerably lower than the uncertainty that arises from fitting statistical models to the raw data used in the IPM (*e.g.* Clark 2003), and so it should be acceptable for almost any application. For stochastic models with continuously varying environments, it is often not computationally feasible to re-run the IPM for 10-50,000 iterations since they are time consuming to run and there are many in PADRINO. Thus, we manually check for shorter term behavior that is similar to published dynamics (*e.g.* stochastic population growth rate ($\lambda_s$) after 1000 iterations). For publications where population growth rates are not available, we manually examine the publication and check the model digitized in PADRINO against some reported behavior (*e.g.* generation time). A given IPM can only enter a scheduled database release if it is explicitly flagged by a digitizer as validated, or if it passes its automated test. The manual testing functionality is contained in the open source *R* package *pdbDigitUtils* (available on GitHub (https://github.com/padrinoDB/pdbDigitUtils)), and PADRINO's build scripts are in the project's GitHub repository (https://github.com/padrinoDB/Padrino/tree/main/R).

### Challenges

Digitizing IPMs into the PADRINO IPM Database is not without issues. First, it is often the case that the complete form of the IPM is not reported: approximately 80% of papers we have examined thus far fall into this category. Many studies may report the general form of the model (*e.g.* $n(z', t+1) = \int_L^U K(z', z)n(z, t)dz$), but do not then report the functional forms of the sub-kernels or vital rates. Without the functional form of all vital rates and sub-kernels, it is impossible to reproduce the IPM. Second, some parameter values may be missing from the main text or supplementary materials - common culprits are terms for the variation of the growth/fecundity kernels, number of meshpoints, and integration bounds (i.e. $L, U$ in Eq 1). The authors of this paper have been guilty of this, as well as other sins of omission, in their own IPM publications. The intent here is not to alienate other authors, but offer a gentle reminder that reporting all parameter values and functional forms can go a long way towards making their science reusable and extensible. Reproducible science can often bring great benefit to the original authors as well as the broader community (Kousta et al. 2019).

### Adding your own IPMs to PADRINO

The easiest way to ensure your own IPMs can get added to PADRINO is to use `ipmr` when constructing them. Since both `Rpadrino` and `ipmr` use the `proto_ipm` object to generate IPM objects, almost all the information the PADRINO digitization team needs to make the model available is already contained in the `proto_ipm`. We provide the `make_ipm_report()` in `ipmr` to help users generate a reproducible document containing equations, parameter values, and implementation details of their IPMs, so that things like notation don't become stumbling blocks in reporting.

Of course, as acknowledged in the main text, there are IPMs that `ipmr` cannot handle currently. We therefore advocate that as a general rule for reproducibility, writing down the IPM and vital rate equations in either the main text or supplementary information of the publication, and all associated parameter values implementation details (e.g. integration rule, range of the trait values used).

There are certainly columns in the `Metadata` table which cannot be inferred from the `proto_ipm` or the equations and parameter values described above (e.g. latitude/longitude of populations, starting/ending year of data collection, species names). We therefore advocate for authors provided comprehensive descriptions for their study species and sites, regardless of what software they use to construct IPMs.

## Technical overview of PADRINO

PADRINO is structured such that each model gets one row for the Metadata table, and an arbitrary number of rows for every table after that. Some models may have 0, 1, or many rows for some of these tables.

Information for each model is linked across tables by the `ipm_id` column. Complete descriptions of each column are provided here.


When a user calls `pdb_make_proto_ipm()` and specifies `ipm_id`s, the function loops over the specified IDs subsetting the database to each single one. It then calls `.make_proto()`, which first translates each IPM component from PADRINO syntax into `ipmr` syntax, then calls `define_*` functions from `ipmr` to generate a `proto_ipm`. If there is more than one ID requested, then `pdb_make_proto_ipm()` repeats the process as many times as requested to generate a list of `proto_ipm`s. This list can be passed to `pdb_make_ipm()`, `pdb_make_ipm()` is a thin wrapper around `ipmr`'s `make_ipm()`, and allows for different sets of additional arguments to be passed to each individual IPM build process.

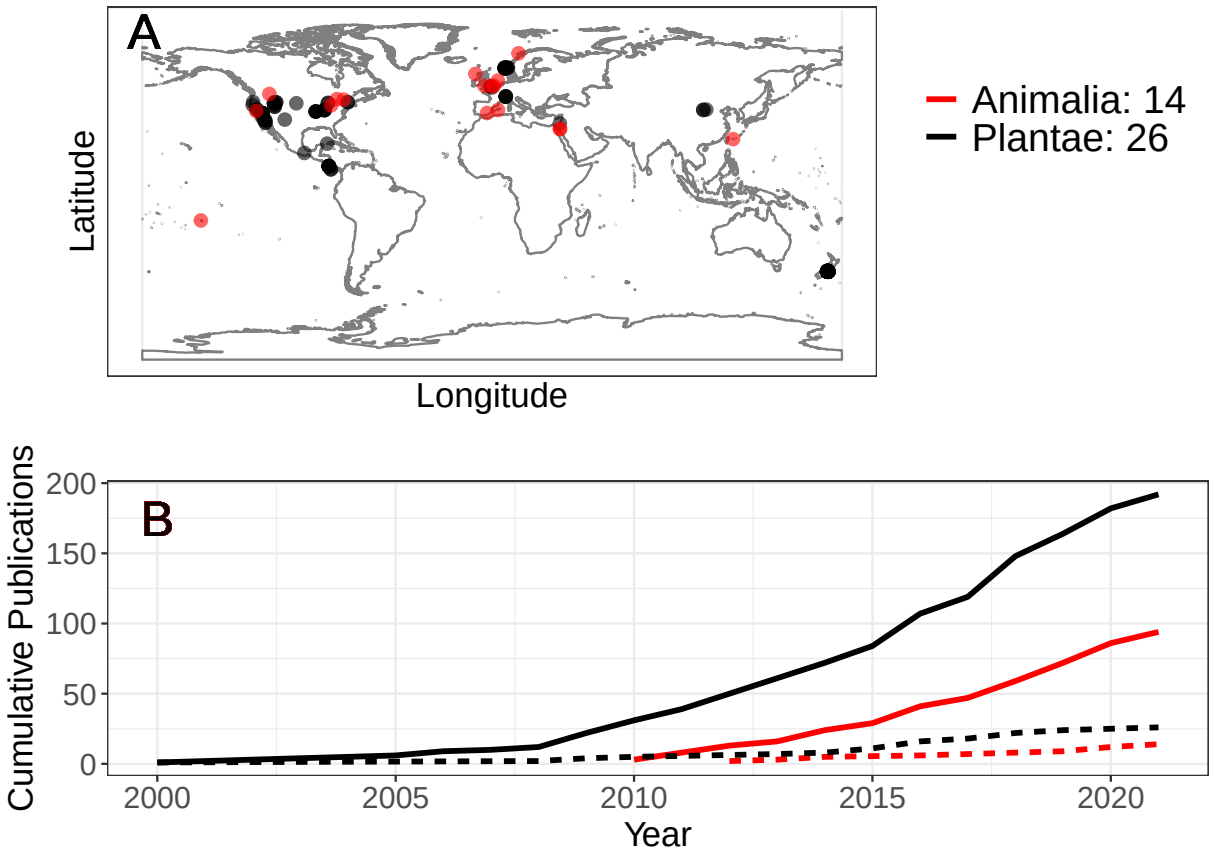



**Figure S1**: The geographic and temporal coverage of studies in the PADRINO IPM Database. (A) Geographic distribution of publications currently contained in PADRINO (i.e. studies from Table 1). (B) Cumulative number of publications found by our search criteria by year (solid lines), and the number that are in the released version of PADRINO (dashed lines). Future releases will include those that we have found, but are not yet completely digitized (i.e. those represented by solid lines, but not yet included in the dashed lines). See the Supplementary Data for a complete list of IPM publications.

Table S1: Summaries of the information contained in each table of the PADRINO database. A complete guide to each column in each table is available on the project's webpage in the form of the guide provided to digitizers (there are too many columns to provide the information here).

| Table | Description |
|---|---|
| Metadata | This table contains metadata for each IPM. This is organized into taxonomic information (full taxonomy plus functional group information), publication information (citation, authorship, source), data collection information (study period/duration, GPS coordinates, ecoregion), and model specific information (studied sexes, eviction corrections, treatments applied, and model implementation details). See Table S2 for more information on these columns. |
| State Variables | This table contains the names of the state variables used in the model and whether or not they are discrete or continuously distributed. |
| Continuous States | This table contains names and ranges for each continuously distributed state variable in the model, as well as which kernels they apply to (kernels are the P(z',z), F(z',z), and C(z',z) in Main Text's Eq 1). |
| Integration Rules | This table contains information on how each continuous state variable is numerically approximated in the model (i.e. number of meshpoints, which integration rule was used). |
| Population Trait Distributions | This table contains the names of the population trait distributions used in the model (n(z,t) and n(z',t+1) in Main Text's Eq. 1). |
| IPM Sub-kernels | This table contains the functional forms of each sub-kernel in the IPM (_e.g._ P(z',z) in Main Text's Eq 1 becomes 'P = s * G'), and information on which traits it acts on and creates. This table makes use of ipmr's [parameter set index notation](https://levisc8.github.io/ipmr/articles/index-notation.html) to concisely represent models which may produce many kernels. |
| Vital Rate Functions | This table contains the functional forms of each vital rate in the IPM (e.g. 'mu_g = int_g + slope_g * z_1'). This table makes use of ipmr's [parameter set index notation](https://levisc8.github.io/ipmr/articles/index-notation.html) to concisely represent models which may produce many kernels. |
| Parameter Values | This table contains the names and values of each parameter in the model, with the exception of parameters that are associated with continuous environmental variation. |
| Continuous Environmental Variation | This table contains parameter values and functional forms of any continuously varying environmental conditions (e.g. yearly variation in precipitation and/or temperature). Any model that contains information in this table is considered stochastic by default, as these variables must be sampled at least once to construct a model with Rpadrino. |
| Parameter Set Indices | This table contains the parameter set indices. These are substituted into the IPM kernels and vital rate expressions when a model is built, so that a single symbolic expression can represent an arbitrary number of realized expression. For example, the vital rate expression 'mu_g_yr = g_int_yr + g_slope * z_1' can be used to represent a range of years for a model with year-specific intercepts. This table contains values substituted in for '_yr' across the model. See the [ipmr vignette on Index Notation](https://levisc8.github.io/ipmr/articles/index-notation.html) for more details. |

Table S2: All columns contained in the Metadata table.

| Concept | Column Name | Description |
|---|---|---|
| | ipm_id | Unique ID for each model. |
| Taxonomy | species_author | The Latin species name used by the authors of the paper. |
| | species_accepted | The Latin species name accepted by Catalogue of Life. |
| | tax_genus | The genus name accepted by Catalogue of Life. |
| | tax_family | The family name accepted by Catalogue of Life. |
| | tax_order | The order name accepted by Catalogue of Life. |
| | tax_class | The class name accepted by Catalogue of Life. |
| | tax_phylum | The phylum name accepted by Catalogue of Life. |
| | kingdom | The kingdom name accepted by Catalogue of Life. |
| | organism_type | General functional type of the species (_e.g._ annual, fern, mammal, reptile). |
| | dicot_monocot | If a plant species, whether the species is a dicot or a monocot. |
| | angio_gymno | If a plant species, whether the species is an angiosperm, gymnosperm, or neither. |
| Source | authors | All of a study authors' last names, separated by ';'. |
| | journal | Abbreviated journal name (www.abbreviations.com/jas.php), or 'PhD', 'MSc' if a thesis. |
| | pub_year | The year of publication. |
| | doi | Digital object identifier and/or ISBN (if available). |
| | corresponding_author | The name of the corresponding author on the paper. |
| | email_year | The email address of the corresponding author and the year it was extracted (some email addresses may be defunct now). |
| | remark | Additional remarks from the digitizer regarding the publication, if any. |
| | apa_citation | The full APA citation for the source. |
| | demog_appendix_link | The URL for the Supplementary information containing additional model details, if available. |
| Temporal Metadata | duration | The duration of the study, defined 'study_end - study_start + 1'. Does not consider skipped years. |
| | start_year | The year demographic data collection began. |
| | start_month | The month demographic data collection began. |
| | end_year | The year demographic data collection ended. |
| | end_month | The month demographic data collection ended. |
| | periodicity | Frequency of the model (1: annual transition, 2: semi-annual transition, 0.2: 5 year transition). |
| Spatial Metadata | population_name | The name of the population given in the data source. |
| | number_populations | The number of populations that a given model describes. |
| | lat | The decimal latitude of the population. |
| | lon | The decimal longitude of the population. |
| | altitude | The altitude of the population above sea level, obtained either from the publication or Google Earth. |
| | country | The ISO3 code for the country or countries in which the data were collected. |
| | continent | The continent or continents on which the data were collected. |

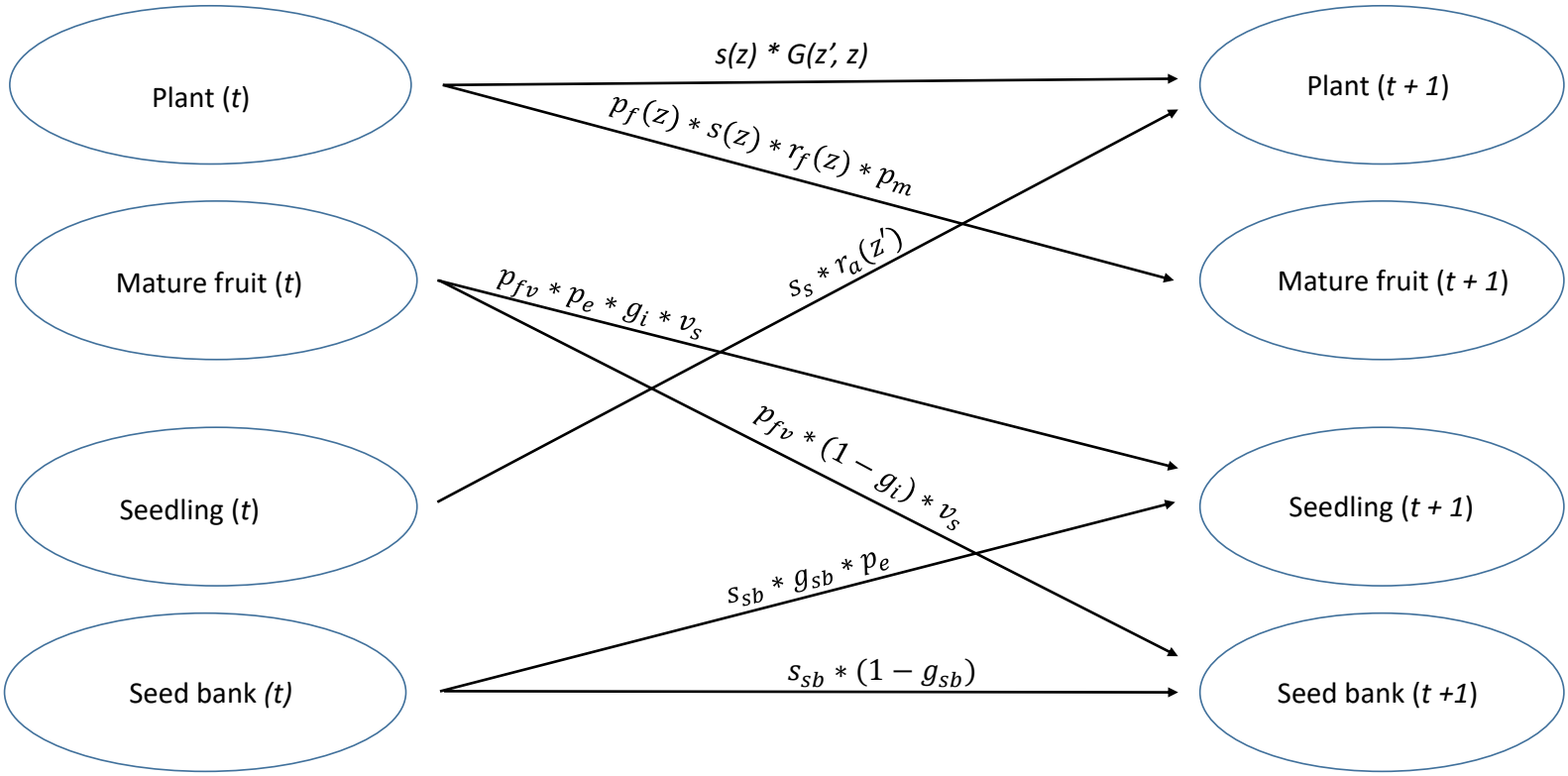
Table S2: All columns contained in the Metadata table. *(continued)*

| Concept | Column Name | Description |
|---|---|---|
| | ecoregion | The terrestrial or aquatic ecoregion corresponding to the [World Wildlife Fund](https://www.worldwildlife.org/biomes) classification. If data are from a controlled setting (greenhouse, lab), denoted with 'LAB'. |
| Model-specific metadata | studied_sex | Sexes used to construct the model. |
| | eviction_used | Whether or not the authors explicitly state that they corrected for eviction (see Williams et al. 2012). |
| | evict_type | If the authors did correct for eviction, then the type of correction that was applied. Current options are 'stretched_domain', 'truncated_distributions', and 'disctrete_extrema'. |
| | treatment | A description of any experimental treatment applied to the population. |
| | has_time_lag | Whether or not the model contains a time lagged vital rate/kernel. |
| | has_age | Whether or not the model has age structure in addition to other continuous state variables. |
| | has_dd | Whether or not the model is density dependent. |
| | is_periodic | Whether or not the model is periodic. |

# Appendix 7: Supplementary Information for Chapter 4

## Model diagram

The following diagram describes a single iteration of the IPM.


Plant (t)
Mature fruit (t)
Seedling (t)
Seed bank (t)
Plant (t + 1)
Mature fruit (t + 1)
Seedling (t + 1)
Seed bank (t +1)
s(z) * G(z', z)
$p_f(z) * s(z) * r_f(z) * p_m$
$s_s * r_a(z')$
$p_{fv} * p_e * g_i * v_s$
$p_{fv} * (1 - g_i) * v_s$
$s_{sb} * g_{sb} * p_e$
$s_{sb} * (1 - g_{sb})$

See the next section for a complete description of each vital rate parameter.

### IPM Equations

IPMs describe how the abundance and distribution of a continuously distributed trait changes in a population through discrete time. Vital rates are combined in projection kernels that describe state-dependent per-capita contributions of existing individuals to the population trait distribution in the following time step via survival and development (denoted $(P(z', z))$) and sexual and asexual reproduction (denoted $F(z', z)$ and $C(z', z)$ respectively).

$$n(z', t+1) = \int_L^U [G(z'|z, \sigma, \theta) * s_a(z, \theta)]n(z, t)dz + s_s * r_a(z')sdl(t), \tag{4.1.1}$$

$$mf(t+1) = \int_L^U [p_f(z, \theta) * s_a(z, \theta) * r_f(z, \theta) * p_m * p_{fv}]n(z, t)dz, \tag{4.1.2}$$

$$sdl(t+1) = p_e * g_i * v_s * mf(t) + s_{sb} * g_{sb} * p_e * sb(t), \tag{4.1.3}$$

and

$$sb(t+1) = s_{sb} * (1 - g_{sb}) * sb(t) + (1 - g_i) * v_s * mf(t). \tag{4.1.4}$$

The survival probability of non-seedlings function, $s_a(z, \theta)$, is given by:

$$\begin{aligned} Logit(s_a(z, \theta)) = \beta_{0,s,i} + \beta_{s,z} * z + \\ \beta_{s,\theta_t,dry} * \theta_{t,dry,i} + \beta_{s,\theta_t,wet} * \theta_{t,wet,i} + \\ \beta_{s,\theta_p,dry} * \theta_{p,dry,i} + \beta_{s,\theta_p,wet} * \theta_{p,wet,i} + \\ \beta_{s,\theta_{s3},dry} * \theta_{s3,dry,i} + \beta_{s,\theta_{s3},wet} * \theta_{s3,wet,i} + \\ \beta_{s,\theta_t \times z,dry} * \theta_{t,dry,i} * z + \beta_{s,\theta_t \times z,wet} * \theta_{t,wet,i} * z + \\ \beta_{s,\theta_p \times z,dry} * \theta_{p,dry,i} * z + \beta_{s,\theta_p \times z,wet} * \theta_{p,wet,i} * z + \\ \beta_{s,\theta_{s3} \times z,dry} * \theta_{s3,dry,i} * z + \beta_{s,\theta_{s3} \times z,wet} * \theta_{s3,wet,i} * z + \\ \beta_{s,native} * g(i) + \beta_{s,i}, \end{aligned} \tag{4.1.5}$$

where *wet* and *dry* denote wet season and dry seasons covariation values, $i$ indexs each site in Table 4.1, and the function $g()$ takes a site $i$ and returns 0 for sites in the invaded range and 1 for sites in the native range. The development function, $G(z'|z, \sigma, \theta)$ is given by:

$$G(z'|z, \sigma, \theta) = f_G(z'|\mu_G(z, \theta), \sigma_G(z, i)), \tag{4.1.6}$$

where $f_G$ denotes a normal probability density function, $\mu_G(z, \theta)$ is given by:

$$\begin{aligned} \mu_G(z, \theta) = \beta_{0,G,i} + \\ \beta_{G,\theta_t,dry} * \theta_{t,dry,i} + \beta_{G,\theta_t,wet} * \theta_{t,wet,i} + \\ \beta_{G,\theta_p,dry} * \theta_{p,dry,i} + \beta_{G,\theta_p,wet} * \theta_{p,wet,i} + \\ \beta_{G,\theta_{s3},dry} * \theta_{s3,dry,i} + \beta_{G,\theta_{s3},wet} * \theta_{s3,wet,i} + \end{aligned} \tag{4.1.7}$$

$$\beta_{G,\theta_t \times z,dry} * \theta_{t,dry,i} * z + \beta_{G,\theta_t \times z,wet} * \theta_{t,wet,i} * z +$$
$$\beta_{G,\theta_p \times z,dry} * \theta_{p,dry,i} * z + \beta_{G,\theta_p \times z,wet} * \theta_{p,wet,i} * z +$$
$$\beta_{G,\theta_{s3} \times z,dry} * \theta_{s3,dry,i} * z + \beta_{G,\theta_{s3} \times z,wet} * \theta_{s3,wet,i} * z,$$
$$\beta_{G,native} * g(i) + \beta_{G,i},$$

$\sigma_G(z,i)$ is given by:

$$\sigma_G(z,\theta) = \beta_{0,\sigma_G,i} + \tag{4.1.8}$$
$$\beta_{\sigma_G,\theta_t,dry} * \theta_{t,dry,i} + \beta_{\sigma_G,\theta_t,wet} * \theta_{t,wet,i} +$$
$$\beta_{\sigma_G,\theta_p,dry} * \theta_{p,dry,i} + \beta_{\sigma_G,\theta_p,wet} * \theta_{p,wet,i} +$$
$$\beta_{\sigma_G,\theta_{s3},dry} * \theta_{s3,dry,i} + \beta_{\sigma_G,\theta_{s3},wet} * \theta_{s3,wet,i} +$$
$$\beta_{\sigma_G,native} * g(i) + \beta_{\sigma_G,i},$$

The probability of flowering function, $p_f(z,\theta)$, is given by:

$$Logit(p_f(z,\theta)) = \beta_{0,p_f,i} + \beta_{z,p_f} * z + \tag{4.1.9}$$
$$\beta_{p_f,\theta_t,dry} * \theta_{t,dry,i} + \beta_{p_f,\theta_t,wet} * \theta_{t,wet,i} +$$
$$\beta_{p_f,\theta_p,dry} * \theta_{p,dry,i} + \beta_{p_f,\theta_p,wet} * \theta_{p,wet,i} +$$
$$\beta_{p_f,\theta_{s1},dry} * \theta_{s1,dry,i} + \beta_{p_f,\theta_{s1},wet} * \theta_{s1,wet,i} +$$
$$\beta_{p_f,\theta_t \times z,dry} * \theta_{t,dry,i} * z + \beta_{p_f,\theta_t \times z,wet} * \theta_{t,wet,i} * z +$$
$$\beta_{p_f,\theta_p \times z,dry} * \theta_{p,dry,i} * z + \beta_{p_f,\theta_p \times z,wet} * \theta_{p,wet,i} * z +$$
$$\beta_{p_f,\theta_{s1} \times z,dry} * \theta_{s1,dry,i} * z + \beta_{p_f,\theta_{s1} \times z,wet} * \theta_{s1,wet,i} * z,$$
$$\beta_{p_f,native} * g(i) + \beta_{p_f,native \times z} * g(i) * z + \beta_{p_f,i},$$

The number of flowers produced conditional on flowering function, $r_f(z,\theta)$, is given by:

$$Log(r_f(z,\theta)) = \beta_{0,r_f,i} + \beta_{z,r_f} * z + \tag{4.1.10}$$
$$\beta_{r_f,\theta_t,mean} * \theta_{t,mean,i} + \beta_{r_f,\theta_t,seas} * \theta_{t,seas,i} +$$
$$\beta_{r_f,\theta_p,total} * \theta_{p,total,i} + \beta_{r_f,\theta_p,seas} * \theta_{p,seas,i} +$$
$$\beta_{r_f,\theta_{s3},mean} * \theta_{s3,mean,i} + \beta_{r_f,\theta_{s3},seas} * \theta_{s3,seas,i} +$$
$$\beta_{r_f,\theta_t \times z,mean} + \theta_{t,mean,i} * z + \beta_{r_f,\theta_t \times z,seas} * \theta_{t,seas,i} * z +$$
$$\beta_{r_f,\theta_p \times z,total} + \theta_{p,total,i} * z + \beta_{r_f,\theta_p \times z,seas} * \theta_{p,seas,i} * z +$$
$$\beta_{r_f,\theta_{s3} \times z,mean} + \theta_{s3,mean,i} * z + \beta_{r_f,\theta_s 3 \times z,seas} * \theta_{s3,seas,i} * z +$$
$$\beta_{r_f,native} * g(i) + \beta_{r_f,native \times z} * g(i) * z.$$

$g(i)$ is a function that returns 1 if site $i$ in the native range (South Africa) and 0 when site $i$ is located elsewhere. Finally, the size distribution of newly observed non-seedling plants, $r_a(z')$, is given by:

$$r_a(z') = f_{r_a}(z'|\mu_{r_a}, \sigma_{r_a}), \tag{4.1.11}$$

where $f_{r_a}$ is a Gaussian probability density function.

# Vital rate model summaries

### Survival model

```
summary(surv_mod)
```

```
##  Family: bernoulli
##   Links: mu = logit
## Formula: alive ~ log_size + temp_dry_t * log_size + temp_wet_t * log_size + prec_dry_t * log_size + prec_wet_t * log_size + sw3_dry_t *
##    Data: data (Number of observations: 5958)
##   Draws: 4 chains, each with iter = 2000; warmup = 1000; thin = 1;
##          total post-warmup draws = 4000
##
## Group-Level Effects:
## ~site (Number of levels: 13)
##               Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## sd(Intercept)     0.23      0.18     0.01     0.69 1.00     1157     1632
##
## Population-Level Effects:
##                     Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## Intercept               2.45      0.60     1.31     3.71 1.00     1224     1661
## log_size                0.23      0.13    -0.02     0.49 1.00     1292     1963
## temp_dry_t             -0.84      0.38    -1.60    -0.05 1.00     1383     1440
## temp_wet_t             -0.43      0.52    -1.49     0.57 1.00     1700     1880
## prec_dry_t             -2.92      0.50    -3.92    -1.90 1.00     1384     1678
## prec_wet_t              1.64      0.62     0.36     2.88 1.00     1193     1547
## sw3_dry_t               1.95      0.78     0.35     3.44 1.00     1167     1591
## sw3_wet_t              -2.00      1.05    -4.01     0.18 1.00     1251     1706
## native                  0.13      0.57    -1.05     1.30 1.00     2530     2167
## log_size:temp_dry_t    -0.38      0.08    -0.52    -0.22 1.00     2060     2461
## log_size:temp_wet_t     0.11      0.10    -0.08     0.30 1.00     2575     2853
## log_size:prec_dry_t    -1.04      0.12    -1.28    -0.81 1.00     1495     2215
## log_size:prec_wet_t     0.84      0.14     0.57     1.12 1.00     1256     1832
## log_size:sw3_dry_t      1.05      0.17     0.71     1.39 1.00     1265     1735
## log_size:sw3_wet_t     -1.11      0.23    -1.57    -0.66 1.00     1428     1971
##
## Draws were sampled using sample(hmc). For each parameter, Bulk_ESS
## and Tail_ESS are effective sample size measures, and Rhat is the potential
## scale reduction factor on split chains (at convergence, Rhat = 1).
```

### Growth model

```
summary(grow_mod)
```

```
##  Family: gaussian
##   Links: mu = identity; sigma = log
## Formula: log_size_next ~ log_size + temp_dry_t * log_size + temp_wet_t * log_size + prec_dry_t * log_size + prec_wet_t * log_size + sw3
##          sigma ~ log_size + temp_dry_t + temp_wet_t + prec_dry_t + prec_wet_t + sw3_dry_t + sw3_wet_t + native + (1 | site)
##    Data: data (Number of observations: 4280)
##   Draws: 4 chains, each with iter = 2000; warmup = 1000; thin = 1;
##          total post-warmup draws = 4000
##
## Group-Level Effects:
## ~site (Number of levels: 13)
##                     Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## sd(Intercept)           0.46      0.21     0.21     1.02 1.01     1210     1625
## sd(sigma_Intercept)     0.21      0.12     0.09     0.50 1.00      967      932
##
## Population-Level Effects:
##                     Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## Intercept               0.05      0.68    -1.30     1.41 1.00     1806     1599
## sigma_Intercept        -0.50      0.31    -1.16     0.10 1.00     1844     1386
## log_size                0.85      0.03     0.79     0.92 1.00     1898     2330
## temp_dry_t              0.06      0.51    -0.95     1.02 1.00     2139     1514
## temp_wet_t             -0.24      0.68    -1.56     1.16 1.00     2313     1630
## prec_dry_t             -0.40      0.55    -1.54     0.68 1.00     1813     1343
## prec_wet_t              0.15      0.74    -1.34     1.65 1.00     1554     1560
## sw3_dry_t              -0.10      0.91    -1.93     1.73 1.00     1604     1501
## sw3_wet_t               0.24      1.25    -2.28     2.77 1.00     1682     1524
## native                  0.25      0.83    -1.44     1.98 1.00     2605     1933
## log_size:temp_dry_t    -0.08      0.02    -0.11    -0.04 1.00     2477     2803
## log_size:temp_wet_t     0.05      0.02     0.01     0.10 1.00     3051     2866
## log_size:prec_dry_t    -0.14      0.03    -0.20    -0.09 1.00     1838     2660
## log_size:prec_wet_t     0.12      0.04     0.04     0.21 1.00     1876     2457
## log_size:sw3_dry_t      0.20      0.05     0.11     0.29 1.00     1938     2312
## log_size:sw3_wet_t     -0.19      0.06    -0.30    -0.07 1.00     2116     2460
## sigma_log_size         -0.12      0.01    -0.13    -0.10 1.00     7536     3177
## sigma_temp_dry_t        0.22      0.23    -0.24     0.68 1.00     2154     1600
## sigma_temp_wet_t       -0.34      0.32    -0.98     0.29 1.00     2396     1582
```

```
## sigma_prec_dry_t        0.05      0.25    -0.49     0.54 1.00     1752     1395
## sigma_prec_wet_t        0.23      0.34    -0.44     0.94 1.00     1693     1388
## sigma_sw3_dry_t         0.07      0.42    -0.76     0.94 1.00     1682     1234
## sigma_sw3_wet_t        -0.26      0.59    -1.49     0.87 1.00     1743     1297
## sigma_native            0.49      0.41    -0.32     1.31 1.00     2558     1790
##
## Draws were sampled using sample(hmc). For each parameter, Bulk_ESS
## and Tail_ESS are effective sample size measures, and Rhat is the potential
## scale reduction factor on split chains (at convergence, Rhat = 1).
```

### Pr(Flowering) model

```
summary(repr_mod)
```

```
##  Family: bernoulli
##   Links: mu = logit
## Formula: repro ~ log_size + temp_dry_t_1 * log_size + temp_wet_t_1 * log_size + prec_dry_t_1 * log_size + prec_wet_t_1 * log_size + sw1
##    Data: data (Number of observations: 6581)
##   Draws: 4 chains, each with iter = 2000; warmup = 1000; thin = 1;
##          total post-warmup draws = 4000
##
## Group-Level Effects:
## ~site (Number of levels: 13)
##               Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## sd(Intercept)     2.58      0.99     1.35     5.06 1.00     1536     2058
##
## Population-Level Effects:
##                       Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS
## Intercept                -0.97      1.72    -4.26     2.45 1.00     3793
## log_size                  1.59      0.14     1.33     1.87 1.00     2421
## temp_dry_t_1             -0.87      2.80    -6.57     4.55 1.00     2473
## temp_wet_t_1              1.23      3.89    -6.63     9.35 1.00     2426
## prec_dry_t_1              0.08      4.35    -8.91     8.89 1.00     2433
## prec_wet_t_1             -0.49      2.75    -6.05     4.94 1.00     2309
## sw1_dry_t_1              -0.14      6.29   -12.89    12.90 1.00     2342
## sw1_wet_t_1              -0.32      5.05   -10.74     9.89 1.00     2336
## native                   -1.92      3.88   -10.23     5.54 1.00     2644
## log_size:temp_dry_t_1     0.39      0.27    -0.14     0.93 1.00     2085
```

```
## log_size:temp_wet_t_1    -0.56      0.33    -1.23     0.10 1.00     1742
## log_size:prec_dry_t_1    -0.96      0.29    -1.55    -0.41 1.00     2016
## log_size:prec_wet_t_1     0.61      0.24     0.16     1.10 1.00     1774
## log_size:sw1_dry_t_1      1.37      0.45     0.50     2.31 1.00     1709
## log_size:sw1_wet_t_1     -0.67      0.33    -1.33    -0.03 1.00     1916
## log_size:native           0.21      0.41    -0.63     1.00 1.00     2119
##                       Tail_ESS
## Intercept                 2636
## log_size                  2590
## temp_dry_t_1              2066
## temp_wet_t_1              2340
## prec_dry_t_1              1659
## prec_wet_t_1              2071
## sw1_dry_t_1               2113
## sw1_wet_t_1               2371
## native                    1988
## log_size:temp_dry_t_1     2866
## log_size:temp_wet_t_1     2470
## log_size:prec_dry_t_1     2425
## log_size:prec_wet_t_1     2277
## log_size:sw1_dry_t_1      2005
## log_size:sw1_wet_t_1      2342
## log_size:native           2547
##
## Draws were sampled using sample(hmc). For each parameter, Bulk_ESS
## and Tail_ESS are effective sample size measures, and Rhat is the potential
## scale reduction factor on split chains (at convergence, Rhat = 1).
```

### Flower number model

```
summary(flow_mod)
```

```
##  Family: negbinomial
##   Links: mu = log; shape = identity
## Formula: flower_n ~ log_size + temp_dry_t_1 * log_size + temp_wet_t_1 * log_size + prec_dry_t_1 * log_size + prec_wet_t_1 * log_size +
##    Data: data (Number of observations: 1093)
##   Draws: 4 chains, each with iter = 2000; warmup = 1000; thin = 1;
##          total post-warmup draws = 4000
```

```
##
## Group-Level Effects:
## ~site (Number of levels: 13)
##                           Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS
## sd(Intercept)                 0.53      0.29     0.15     1.24 1.00     1322
## sd(log_size)                  0.25      0.15     0.05     0.65 1.00      982
## cor(Intercept,log_size)       0.01      0.54    -0.92     0.93 1.00     1215
##                           Tail_ESS
## sd(Intercept)                 1283
## sd(log_size)                  1769
## cor(Intercept,log_size)       1916
##
## Population-Level Effects:
##                         Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS
## Intercept                  1.25      0.41     0.37     2.03 1.00     2700
## log_size                   0.64      0.20     0.21     1.02 1.00     2174
## temp_dry_t_1              -1.28      0.88    -3.14     0.34 1.00     1407
## temp_wet_t_1               2.90      1.46     0.15     6.04 1.00     1423
## prec_dry_t_1              -0.07      0.32    -0.75     0.58 1.00     1765
## prec_wet_t_1              -1.69      0.90    -3.59     0.15 1.00     1242
## sw3_dry_t_1               -3.25      1.78    -6.94     0.28 1.00     1206
## sw3_wet_t_1                5.19      2.92    -0.78    11.30 1.00     1170
## native                    -0.43      1.19    -2.67     1.74 1.00     1409
## log_size:temp_dry_t_1     -0.46      0.34    -1.27     0.13 1.00     1100
## log_size:temp_wet_t_1      1.01      0.72    -0.22     2.63 1.00     1025
## log_size:prec_dry_t_1     -0.17      0.15    -0.50     0.11 1.00     1389
## log_size:prec_wet_t_1     -0.62      0.45    -1.63     0.21 1.00      917
## log_size:sw3_dry_t_1      -1.30      0.89    -3.34     0.23 1.00      914
## log_size:sw3_wet_t_1       2.12      1.44    -0.36     5.39 1.00      891
##                         Tail_ESS
## Intercept                  2376
## log_size                   1927
## temp_dry_t_1               1979
## temp_wet_t_1               1744
## prec_dry_t_1               2067
## prec_wet_t_1               1652
## sw3_dry_t_1                1520
## sw3_wet_t_1                1360
## native                     2485
```

```
## log_size:temp_dry_t_1     1209
## log_size:temp_wet_t_1     1060
## log_size:prec_dry_t_1     1676
## log_size:prec_wet_t_1     1269
## log_size:sw3_dry_t_1      1088
## log_size:sw3_wet_t_1      1152
##
## Family Specific Parameters:
##       Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## shape     2.36      0.14     2.10     2.65 1.00     6013     2820
##
## Draws were sampled using sample(hmc). For each parameter, Bulk_ESS
## and Tail_ESS are effective sample size measures, and Rhat is the potential
## scale reduction factor on split chains (at convergence, Rhat = 1).
```

### Recruit size model

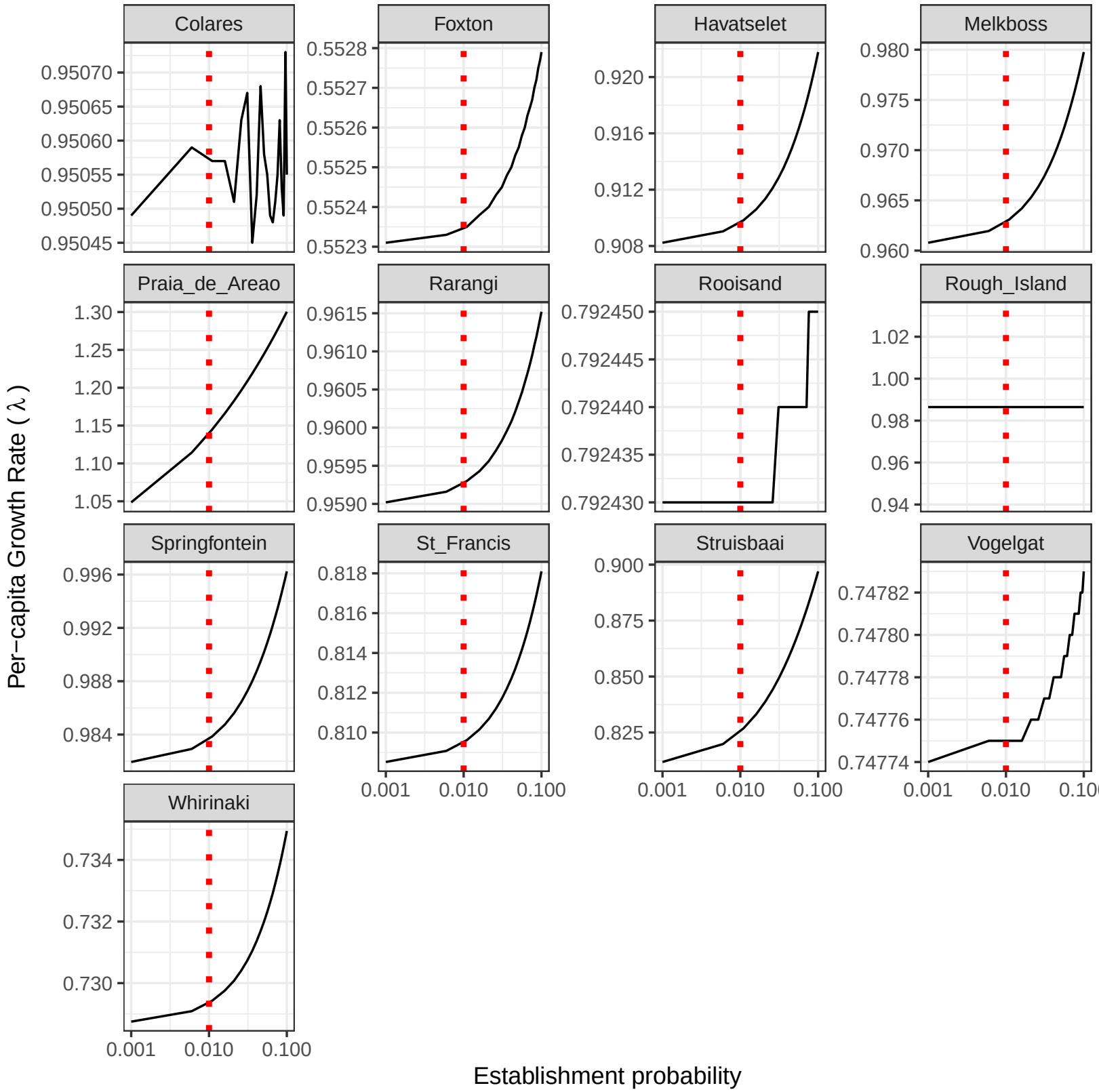
```
summary(recr_mod)
```

```
##  Family: gaussian
##   Links: mu = identity; sigma = identity
## Formula: log_size_next ~ 1
##    Data: recruits (Number of observations: 15)
##   Draws: 4 chains, each with iter = 2000; warmup = 1000; thin = 1;
##          total post-warmup draws = 4000
##
## Population-Level Effects:
##           Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## Intercept    -8.62      0.14    -8.91    -8.34 1.00     1685     1566
##
## Family Specific Parameters:
##       Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## sigma     0.54      0.12     0.37     0.84 1.00     1800     1777
##
## Draws were sampled using sample(hmc). For each parameter, Bulk_ESS
## and Tail_ESS are effective sample size measures, and Rhat is the potential
## scale reduction factor on split chains (at convergence, Rhat = 1).
```

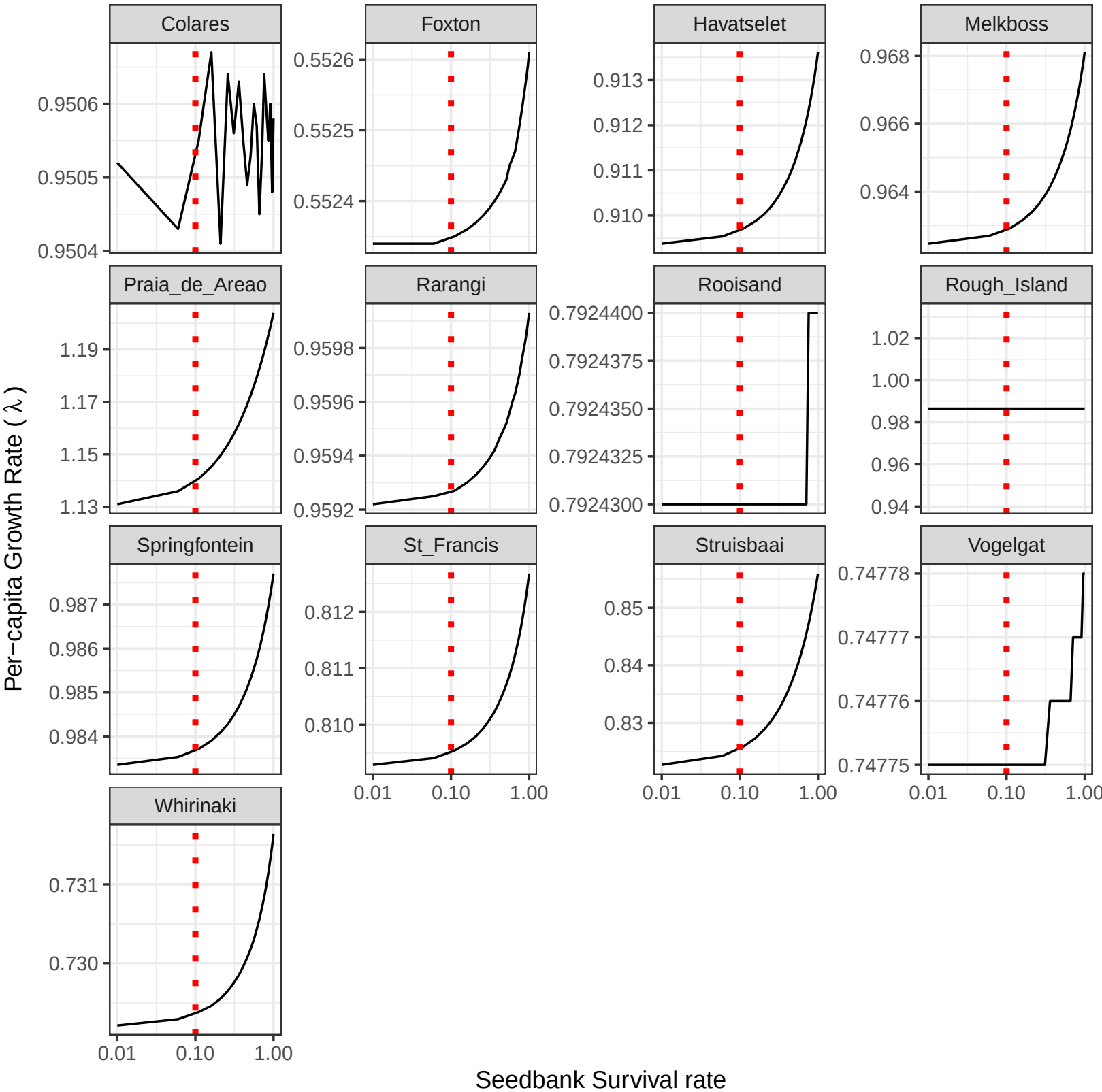
## Sensitivity of site-level $\lambda$s to simulated parameter values

We could not find parameter values for establishment probability ($p_e$) or seedbank survival rate ($s_{sb}$). Therefore, we simulated a range of values from 0-1 for each (incrementing by 0.05), re-building the model, and then computing $\lambda$. The results are reported here. Dotted, red vertical lines how the parameter value we used in the results reported in the main text.

$p_e$


Colares
Foxton
Havatselet
Melkboss
Praia_de_Areao
Rarangi
Rooisand
Rough_Island
Springfontein
St_Francis
Struisbaai
Vogelgat
Whirinaki
Per−capita Growth Rate ( λ )
Establishment probability
0.001
0.010
0.100

$s_{sb}$


Colares
Foxton
Havatselet
Melkboss
Praia_de_Areao
Rarangi
Rooisand
Rough_Island
Springfontein
St_Francis
Struisbaai
Vogelgat
Whirinaki
Per−capita Growth Rate ( λ )
Seedbank Survival rate
0.01
0.10
1.00

# Appendix 8

## Zusätzliches Information

Supplementary materials for each chapter, including published manuscripts, are included in Appendices 1-7. Additionally, for published manuscripts (Chapters 2 & 3), supplementary materials, including R code and associated data, can be digitally downloaded from the respective journal's website. Text and figures can be found on the attached CD.

## A) Curriculum Vitae

### Samuel C. Levin

### Education

*Martin Luther University Halle-Wittenberg*

PhD Biology 2018-2024

*Martin Luther University Halle-Wittenberg*

MSc Biology 2016-2017

*Wake Forest University*

BA Biology 2008-2012

### Work Experience

*Pivot Bio, Inc.*

Data Scientist II 2024-Present

Data Scientist I 2022-2024

*Martin Luther University Halle-Wittenberg*

Research Officer 2018-2022

*Helmholtz-Zentrum für Umweltforschung*

Student Research Assistant 2017

*German Centre for Integrative Biodiversity*

Student Research Assistant 2016 - 2018

### Publications

**Journal Articles** * *denotes mentee*

Novoa, A., Hirsch, H., Castillo, M.L., Canavan, S., Gonzalez, L., Richardson, D.M., *et al.* (2023) Genetic and morphological insights into the Carpobrotus hybrid complex around the world. Neobiota. DOI: 10.3897/neobiota.89.109164.

**Levin SC**, Evers S*, Pena-Guerrero M, Compagnoni AC, Childs DZ, Knight TM & Salguero-Gomez R (2022). Rpadrino: an R package to access and use PADRINO, an open access database of Integral Projection Models. Methods in Ecology and Evolution. DOI: 10.1111/2041-210X.13910

Jones OR, Barks P, Stott I, James TD, **Levin SC**, Petry WK *et al.* (2022). Rcompadre and Rage—Two R packages to facilitate the use of the COMPADRE and COMADRE databases and calculation of life-history traits from matrix population models. Methods in Ecology and Evolution. DOI: 10.1111/2041-210X.13792

**Levin SC**, Childs DZ, Compagnoni AC, Evers S*, Knight TM & Salguero-Gomez R (2021). ipmr: Flexible implementation of Integral Projection Models in R. Methods in Ecology and Evolution. DOI: 10.1111/2041-210X.13683

Keppel G, Craven D, Weigelt P, Smith SA, van der Sande MT, Sandel B, **Levin SC**, Kreft H & Knight TM (2021). Synthesizing tree biodiversity data to understand global patterns and processes of vegetation. Journal of Vegetation Science. DOI: 10.1111/jvs.13021

Bogdan A*, **Levin SC**, Salguero-Gomez R & Knight TM. (2021). Demographic analysis of an Israeli *Carpobrotus* population. Plos One. DOI: 10.1371/journal.pone.0250879.

Paniw M, James T, Archer CR, Romer G, **Levin SC**, Compagnoni AC, *et al.* (2021). Global analysis reveals complex demographic responses of mammals to climate change. Journal of Animal Ecology. DOI: 10.1111/1365-2656.13467

Compagnoni AC, **Levin SC**, Childs DZ, Harpole S, Paniw M, Romer G, *et al.* (2021). Short-lived plants have stronger demographic responses to climate. Nature Communications. Nature Communications. DOI: 10.1038/s41467-021-21977-9

**Levin SC**, Crandall RM, Pokoski TC*, Stein C & Knight TM (2020). Phylogenetic and functional distinctiveness explain alien plant population responses to competition. Proceedings of the Royal Society B. DOI: 10.1098/rspb.2020.1070

Sandel B, Weigelt P, Kreft H, Keppel G, van der Sande MT, **Levin SC**, Smith S, Craven DC & Knight TM (2019). Current climate, isolation, and history drive global patterns of tree phylogenetic endemism. Global Ecology and Biogeography. DOI: 10.1111/geb.13001

Compagnoni A, Bibian BJ, Ochocki BM, **Levin SC**, Zhu K & Miller TEX (2019). popler: an R package for extraction and synthesis of population time series from the long-term ecological research (LTER) network. Methods in Ecology and Evolution. DOI: 10.1111/2041-210X.13319

**Levin SC**, Crandall RM, Knight TM (2019) Population projection models for 14 alien plant species in the presence and absence of above-ground competition. Ecology. DOI: 10.1002/ecy.2681.

Carl G, **Levin SC**, Kühn I. (2018) spind: an R Package to Account for Spatial Autocorrelation in the Analysis of Lattice Data. Biodiversity Data Journal. DOI: 10.3897/BDJ.6.e20760.

### Workshops & Invited Talks

Schwartz A, **Levin SC** *et al.*. A high throughput, automated ARACAS platform measures nitrogen fixation rates from associative diazotrophs in undisturbed, soil-grown cereals. North American Symbiotic Nitrogen Fixation Conference. Burlington, VT, June 2024.

**Levin SC**, Childs DZ, Compagnoni AC, Evers S, Knight TM & Salguero-Gomez R. `ipmr`: An R Package for Easy and Flexible Construction and Interpretation of Integral Projection Models. Ecological Society of America, Long Beach, August 2021.

**Levin SC**. Invasive plants: research, control, and what you can do to help! Point Reyes National Seashore, May 2020. (Cancelled due to COVID-19 pandemic).

**Levin SC** & Salguero-Gomez R. Effective, efficient, and safe data collection with UAVs. Oxford University, January 2020.

Salguero-Gomez R, Jones OR, *et al.* A gentle introduction to the COMADRE & COMPADRE databases for demographic analyses. British Ecological Society, Belfast, December 2019.

### Conference Presentations

* *denotes mentee;* # *denotes poster presentations, otherwise oral*

**2020**

Salguero-Gomez R,Che-Castaldo JP, Jones O, Caswell H, Ezard T, Hernandez-Yanez H, Hodgson D, Knight TM, **Levin SC**, Stott I, Thomas C, Vaupel J. (2020) The next generation of demographic databases: Building and delivering a distributed network for user contributions and engagement. Ecological Society of America

**2018**

**Levin SC**, RM Crandall, TC Pokoski, Stein C, Knight TM. Mechanisms underlying the differential success of alien plant species. Ecological Society of America – New Orleans, USA

**2016**

**Levin SC**, Stein C, Knight TM. Phylogenetic novelty alters the strength of biotic interactions for exotic plant species. NeoBiota 2016 – Vianden, Luxembourg

**Levin SC**, Stein C, Knight TM. Phylogenetic novelty alters the strength of biotic interactions for exotic plant species. iDiv Conference – Leipzig, Germany

**2015**

Poor E*,Thompson AH*, **Levin SC**, Knight TM. Novel functional traits aid the success of the invasive biennial Carduus nutans. Washington University in St. Louis Undergraduate Research Symposium – St. Louis, MO #

Workman M*, Thompson AH*, **Levin SC**, Knight TM. Competitive release may increase the fitness of exotic plants in their novel range. Washington University in St. Louis Undergraduate Research Symposium – St. Louis, MO #

**2014**

Patterson A*, Galluppi CG, **Levin SC**, Maynard EE, Knight TM. How plant species become common: examining the success strategies of native and invasive plants. Washington University in St. Louis Undergraduate Research Symposium – St. Louis, MO #

Van Horn T*, Galluppi CG, **Levin SC**, Knight TM. Examining the enemy release hypothesis in Ozark woody species. Washington University in St. Louis Undergraduate Research Symposium – St. Louis, MO #

### Software

Maintainer (current) and developer (> v2.0.0) of *spind*. Project page and CRAN.

Maintainer and developer of *ipmr*. Project page and CRAN.

Maintainer and developer of the *PADRINO IPM Database* and *Rpadrino*. PADRINO page and RPadrino page.

Contributed to development of popler, bRacatus, taxlist, plotbiomes, popdemo, Rcompadre, and Rage.

### Languages

Fluent in English and R, proficient with Stan, Git, shell, and C++, and familiar with Python and German.

### Certifications

United States FAA Part 107 UAV Pilot License

United States NPS S212 A Faller

### Mentoring

| | |
|---|---|
| Sanne Evers | Helmholtz-Zentrum für Umweltforschung |
| Ana Bogdan | Babeş-Bolyai University, Cluj-Napoca, Romania |
| Tyler Pokoski | University of Iowa 2017 |

| | |
|---|---|
| Tom Collins | Missouri S&T 2017 |
| Amy Patterson | Washington University in St. Louis 2015 |
| Amibeth Thompson | Illinois College 2014 |
| Sami Hunkler | University of California, Berkeley 2017 |
| Thomas Van Horn | Washington University in St. Louis 2018 |
| Sarah Link | Eureka High School 2015 |
| Brenda Alvarado | Francis Howell North 2015 |
| Matilda Workman | Kirkwood High School 2017 |
| Elizabeth Poor | Clayton High School 2017 |

**Service**

Reviewer for rOpenSci, Functional Ecology, BMC Ecology, Annals of Botany, and Plant Ecology

*Philadelphia, Pennsylvania, United States of America, den* 01.10.2024 Sam C. Levin

## B) List of publications for the dissertation

**Levin SC**, Evers S, Pena-Guerrero M, Compagnoni AC, Childs DZ, Knight TM & Salguero-Gomez R (2022). Rpadrino: an R package to access and use PADRINO, an open access database of Integral Projection Models. Methods in Ecology and Evolution 13(9): 1923-1929.

**Levin SC**, Childs DZ, Compagnoni AC, Evers S, Knight TM & Salguero-Gomez R (2021). ipmr: Flexible implementation of Integral Projection Models in R. Methods in Ecology and Evolution 12(10): 1826-1834.

### Conference participation and invited talks

**Levin SC**, Childs DZ, Compagnoni AC, Evers S, Knight TM & Salguero-Gomez R. `ipmr`: An R Package for Easy and Flexible Construction and Interpretation of Integral Projection Models. Ecological Society of America, Long Beach, August 2021.

Salguero-Gomez R,Che-Castaldo JP, Jones O, Caswell H, Ezard T, Hernandez-Yanez H, Hodgson D, Knight TM, **Levin SC**, Stott I, Thomas C, Vaupel J. (2020) The next generation of demographic databases: Building and delivering a distributed network for user contributions and engagement. Ecological Society of America

**Levin SC** & Salguero-Gomez R. Effective, efficient, and safe data collection with UAVs. Oxford University, January 2020.

Salguero-Gomez R, Jones OR, *et al.* A gentle introduction to the COMADRE & COMPADRE databases for demographic analyses. British Ecological Society, Belfast, December 2019.

## C) Author contributions

Chapter 2: Levin SC, Childs DZ, Compagnoni AC, Evers S, Knight TM & Salguero-Gomez R (2021). ipmr: Flexible implementation of Integral Projection Models in R. Methods in Ecology and Evolution 12(10): 1826-1834.

- Design: Sam Levin (70%), Dylan Childs (15%), Tiffany Knight(5%), Roberto Salguero-Gomez (10%).
- Implementation: Sam Levin (100%).
- Analysis: Sam Levin (100%).
- Writing: Sam Levin (60%), Dylan Childs (10%), Aldo Compagnoni (5%), Sanne Evers (5%), Tiffany Knight (10%), Roberto Salguero-Gomez (10%).

Chapter 3: Levin SC, Evers S, Pena-Guerrero M, Compagnoni AC, Childs DZ, Knight TM & Salguero-Gomez R (2022). Rpadrino: an R package to access and use PADRINO, an open access database of Integral Projection Models. Methods in Ecology and Evolution 13(9): 1923-1929.

- Data Collection: Sam Levin (5%), Sanne Evers (70%), Mayra Pena-Guerrero (5%), Tomos Potter (20%).
- Implementation: Sam Levin (100%).
- Analysis: Sam Levin (100%).
- Writing: Sam Levin (60%), Dylan Childs (10%), Aldo Compagnoni (5%), Sanne Evers (5%), Tiffany Knight (10%), Roberto Salguero-Gomez (10%).

Chapter 4: Levin SC, Salguero-Gomez R & Knight TM (2023). Relationship between climate and fitness of a highly invasive succulent *unpublished*.

- Design: Sam Levin (70%), Tiffany Knight (20%), Roberto Salguero-Gomez (10%).
- Data collection: Sam Levin (100%).
- Analysis: Sam Levin (100%).
- Writing: Sam Levin (70%), Tiffany Knight (20%), Roberto Salguero-Gomez (10%).

*Philadelphia, Pennsylvania, United States of America, den* 01.10.2024 Sam C. Levin

## D) Eigenständigkeitserklärung

Hiermit erkläre ich, dass die vorliegende Arbeit mit dem Titel "Integral Projection Models Across Scales" bisher weder an der Naturwissenschaftlichen Fakultät I der Martin-Luther-Universität Halle-Wittenberg noch einer anderen wissenschaftlichen Einrichtung zum Zweck der Promotion vorgelegt wurde.

Ferner erkläre ich, dass ich die vorliegende Arbeit selbstständig und ohne unzulässige fremde Hilfe verfasst, sowie keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe. Die den Werken wörtlich oder inhaltlich entnommenen Stellen wurden als solche von mir kenntlich gemacht. Ich erkläre weiterhin, dass ich mich bisher noch nie um einen Doktorgrad beworben habe.

I hereby declare that the present work entitled "Integral Projection Models Across Scales" has not previously been submitted to the Faculty of Natural Sciences I of the Martin Luther University Halle-Wittenberg or to any other scientific institution for the purpose of a doctorate.

Furthermore, I declare that I have written this thesis independently and without unauthorized outside help, and that I have not used any sources or aids other than those indicated. The passages taken verbatim or in terms of content from the works were marked as such by me. I further declare that I have never applied for a doctoral degree before.

*Philadelphia, Pennsylvania, United States of America, den* 01.10.2024 Sam C. Levin