squidSim: a flexible R package for structured and reproducible simulations in Ecology and Evolutionary Biology

Joel L. Pick[1,2,3,*], Hassen Allegue[3,4,5], Yimen G. Araya-Ajoy[2,3], Barbara Class[3,6], Ned A. Dochtermann[3,7], Edward R. Ivimey-Cook[8], Kate L. Laskowski[3,9], Denis Réale[3,10], Jonathan Wright[2,3] Niels J. Dingemanse[3,11], David F. Westneat[3,12],

1 Institute of Ecology and Evolution, University of Edinburgh, Charlotte Auerbach Road, Edinburgh, EH9 3FL, UK

2 Department of Biology, Norwegian University of Science and Technology (NTNU), N-7491 Trondheim, Norway.

3 Members of the SQuID working group

4 Department of Mathematics & Statistics, Dalhousie University, Halifax, Nova Scotia, Canada

5 Institut Maurice-Lamontagne, Fisheries and Oceans Canada, Mont-Joli, Quebec, Canada

6 Direction pour la Science Ouverte (DipSO), INRAE, France

7 Department of Biological Sciences; North Dakota State University, USA

8 School of Biodiversity, One Health and Veterinary Medicine, University of Glasgow, UK

9 Department of Evolution and Ecology, University of California Davis, Davis CA 95616 USA

10 Département des Sciences Biologiques, Université du Québec à Montréal, Montreal, QC, Canada

11 Behavioural Ecology, Faculty of Biology, LMU Munich, Planegg-Martinsried 82152, Germany

12 Department of Biology, University of Kentucky, Lexington, Kentucky, US

***** Corresponding author, email address: joel.pick@ed.ac.uk

Running title: Structured simulations for ecology and evolution

Abstract

1. Complex statistical methodology now allows a growing array of questions to be addressed in ecology and evolutionary biology. However, the particular question being addressed or the complex nature of the data collected often raise issues with how statistical models perform and potentially limit inference. Simulations provide a powerful approach to help empiricists understand the assumptions, limitations, and output of generalised linear mixed models (GLMMs), advance teaching of statistical modelling and design more informed studies around their usage.

2. Datasets in ecology and evolutionary biology often have complex hierarchical structures, which create challenges in creating simulations. This problem is exacerbated by the current lack of flexible and reproducible tools that facilitate simulating complex data from a wide range of data structures.

3. Here we present the `squidSim` R package, a flexible and logical program designed to accommodate many of the common data structures in ecology and evolutionary biology. The program can simulate from a wide diversity of models in a generalised linear mixed model (GLMM) framework, including data from Gaussian and non-Gaussian models, multi-response models, as well as spatial, temporal, genetic and phylogenetic effects.

4. In addition to facilitating simulations for a wide range of models and data structures, `squidSim` R package provides a fully reproducible workflow and has established utility for teaching. We also provide a graphical user interface via the `shinySim` R package.

Keywords: simulation, linear models, hierarchical models, random effects, genetic variation, pedigree, phylogeny, autocorrelation, multivariate, reproducibility

1 Introduction

As computational power grows, the fields of ecology and evolutionary biology (E&E) are increasingly dominated by sophisticated statistical methods that can deal with the complex data structures that researchers frequently encounter. These complex data can be structured over space or time, include genetic or phylogenetic relatedness of individuals and species, or be hierarchically structured, with data varying at different hierarchical levels, such as having repeated measurements of individuals. Typically, researchers analyse these data within a (generalised) linear mixed model (GLMM) framework, and extensions thereof, which offer flexibility in dealing with missing data and unbalanced designs, non-independence, and the complex data structures that are common in E&E datasets (Kruuk, 2004; **?**; O'Hara, 2009; **?**; **?**; Nakagawa & Schielzeth, 2013; Dingemanse & Dochtermann, 2013). However, as statistical methods become more complex, we are faced with the challenge of understanding their assumptions and limitations and designing studies around their use.

Monte Carlo simulations (hereafter simulations) provide a powerful approach to help empiricists do this. Simulations provide a way of creating artificial datasets through randomly generating data from known underlying deterministic models and probability distributions that are structured to imitate real or hypothetical situations. These simulations have many applications in E&E. First, simulations are a fundamental tool for statistical research, helping researchers understand and test new statistical methods (Morris *et al.*, 2019; Lotterhos *et al.*, 2022; DiRenzo *et al.*, 2023). By simulating data under a known data-generating process, researchers can test whether a statistical tool is doing what they think, explore how models perform under different scenarios, assess what limitations they may have, and compare different statistical models to determine which predicts the focal process best (van Benthem *et al.*, 2017; Westneat *et al.*, 2020; Schielzeth *et al.*, 2020). Importantly, because the true underlying values are known, there is an absolute benchmark with which to compare the results. Second, simulations are a fantastic and underused tool for teaching statistics (Allegue *et al.*, 2017; Kéry & Schaub, 2012; Kéry & Royle, 2020). Simulating and then analysing datasets enables students to see what assumptions are being made and better understand the output of statistical models. They also allow educators to create datasets with known characteristics that can be used to demonstrate target principles to students. Third, simulations have an important and underappreciated role in study design and prospective design analysis. The sampling distributions of many parameters in complex statistical models are unknown and consequently simulations are required to assess metrics such as power, accuracy and bias (Martin *et al.*, 2011; Dingemanse & Dochtermann, 2013; Kain *et al.*, 2015; Pick *et al.*, 2023a). Finally, simulations can be used to aid statistical inference, including posterior predictive checks (Gelman & Hill, 2007), parametric bootstrapping (Stoffel *et al.*, 2017), and creation of null models and distributions (e.g. **?**Ihle *et al.*, 2019; Pick *et al.*, 2023b,a). Simulations are therefore a key tool by which researchers can better assess statistical methodology, increase analytical robustness, and guide study design. However, in our experience simulations are a massively underused tool in most fields, including E&E, and are even uncommon in methodological studies (DiRenzo *et al.*, 2023).

In theory, simulating data is simple. All commonly used programming languages in E&E (e.g. R, Python, Julia) provide built-in functions that allow for (pseudo-)random data simulation (e.g. the `rnorm()`, `rpois()` and `rbinom()` functions in `R`). This functionality allows you to iteratively build simulations of varying complexity. So why are simulations not more widespread in E&E? There are several barriers to their uptake. First, our own experience shows that many researchers find the idea of simulations intimidating and overly time-consuming. Although there are many basic functions for simple simulations, data in E&E often have a level of complexity that can be challenging to implement in simulations. As discussed above, researchers typically have structured data (hierarchical, genetic, phylogenetic, temporal, spatial), and it may be unclear how to practically incorporate this complexity into a simulation. Consequently, considerable coding experience may be required to simulate such data. Given the number of existing simulation studies across fields, we might presume that there exists a lot of code on which to base new simulations. This, however, highlights our second problem: simulations are often coded from scratch by experienced coders, likely in an idiosyncratic way according to their particular coding style. The code is typically not produced with transferability or education in mind, and is often written to optimise simulations for a particular question. Furthermore, code is rarely provided with the corresponding publication (Culina *et al.*, 2020; Kimmel *et al.*, 2023; Kellner *et al.*, 2025) and, when it is, it often either does not run or is badly documented (Kellner *et al.*, 2025). This general lack of clarity and standardisation means that code is often difficult for others to understand and adapt. These two problems are compounded by the lack of easily accessible tools. As we discuss below, there are few software packages available to simulate complex data, and those that are available are either specialised for a particular task (e.g. power analysis for a particular statistical model), and/or do not have the flexibility to incorporate the different kinds of data structures commonly found in E&E. Furthermore, these tools are often not utilised in simulation studies, which makes linking available code on simulations to these tools difficult. Whilst the focus of the researcher should be on the parameterisation of the simulations, much of the struggle of creating simulations rests on an individual's ability to create their simulations from scratch or decipher inaccessible code. We believe that a simple, flexible framework for implementing simulations, that emphasises the statistical model and parameters of the simulation rather than coding ability, will help remove this barrier to this important methodology.

Here we present `squidSim`, an R package that can flexibly produce a extensive array of simulations based on the structure of a GLMM. As starting with simulations can seem like a daunting task, `squidSim` is designed to facilitate that transition and focus attention on the data structure and parameters needed for simulation, rather than programming knowledge. Inputting a large number of parameters for a complex simulation inevitably leads to convoluted code, often requiring troubleshooting for misplaced or missing brackets and commas. To aid with this, we also provide a GUI interface (contained in the `shinySim` R package `https://github.com/squidgroup/shinySim`), which provides a user-friendly way to generate the R code required to simulate with `squidSim`, again taking the focus off coding and placing it on the parameters of the simulation. The `squidSim` R package also comes with a large amount of documentation and extensive worked examples (https://squidgroup.org/squidSim_vignette/index.html). Although the focus of these tools is largely on those starting with simulation, we believe that `squidSim` also provides a useful tool for experienced programmers. A major motivation for the `squidSim` package is to provide a consistent framework for simulations, which can be interpreted by many people rather than having to decipher someone's personal code. It has therefore been constructed with reproducibility in mind. Not only does it use a consistent syntax, but also the `squidSim` object produced from a simulation contains all the information used to parameterise the simulation and therefore to reproduce its results.

2 The `squidSim` R package

The `squidSim` R package is designed to simulate data from any GLMM structure (i.e. if you can analyse data in this framework, in theory you can simulate data from it). As well as accommodating a hierarchical data structure, `squidSim` allows the simulation of uni- and multi-response data with genetic and phylogenetic effects, temporal and spatial variation, and from Gaussian, Poisson and binomial distributions (including several link functions). Following from the ethos of the original `squid` R package (Allegue *et al.*, 2017), `squidSim` allows the simulation of an idealised population and then provides functions to sample from this population. As well as being able to simulate data, the package can therefore be used to create realistic sampling schemes, and so explore the potential biases created by this sampling and design appropriately powered sampling schemes.

squidSim joins several other existing R packages that perform simulations. These packages are generally limited in their scope (i.e. the range of model structures they accommodate) or focused on a particular usage (Table 1), particularly power analysis. Although this is useful, simulations can be used for many additional tasks (e.g. estimating sampling distributions, bias, precision, etc.). `squidSim` is specifically designed to flexibly simulate ecologically realistic datasets, and although the package itself does not conduct power analyses, these (and other such analyses) can be easily coded using the output (examples of which are provided in the vignette; https://squidgroup.org/squidSim_vignette/9-power.html).

The squidSim package can be installed from github using the devtool package, and loaded in R

```
devtools::install_github("squidgroup/squidSim")
library(squidSim)
```

In the following sections, we discuss in more detail the process of creating or inputting a data structure, specifying parameters and simulating data, and sampling from those data.

3 Data Structure

To simulate complex, structured data, you need to have a structure that describes the organisation of the data. In simple models, this data structure is just represented by the sample size, for example, in a simple linear model with predictors that vary at the level of the observations. More complex data structures in squidSim are expressed as a data.frame (or matrix), with all the grouping factors and their levels/IDs, as we would see in a typical dataset, for example the IDs of different individuals, locations, or sexes. IDs from this data structure data.frame can also be used to link to more complex data structure information, such as pedigrees, phylogenies, spatial correlation matrices or other covariance matrices, which can also be input (see section 4.3).

With the squidSim package, the user can either make use of any existing data structure they have access to, or create data structures themselves. For example, the make_structure() function creates simple nested and crossed hierarchical data structures (Figure 1). Importantly, make_structure() only produces balanced data structures which are often not realistic for real world datasets, but sampling functions can be used to make them unbalanced, as outlined in Section 5.

Table 1: Comparison of different simulation R packages.

Function	squidSim	squid[1]	faux[2]	SIMR[3]	PAMM[4]
Data Structure					
Generate balanced structure	✓	✓	✓	✗	✓
Generate unbalanced structure	✓	✗	✓	✗	✗
Generate hierarchical structure	✓	✓	✓	✗	✓
Import existing data structure	✓	✗	✓	✓	✓
Accommodate multiple hierarchical levels	✓	✓[a]	✓	✓	✗
Simulations					
Imports known predictors	✓	✗	✓	✓	✓
Simulates predictors	✓	✓[b]	✓	✗	✓[c]
Predictors at multiple hierarchical levels	✓	✗	✗	✗	✗
Hierarchical (Random intercepts and slopes)	✓	✓	✓	✓	✓
Non-Gaussian	✓	✗	✓	✓	✗
Multivariate	✓	✓[d]	✗	✗	✗
Phylogenetic effects	✓	✗	✗	✗	✗
Genetic effects	✓	✗	✗	✗	✗
Temporal and spatial autocorrelation	✓	✗	✗	✗	✗
Sampling					
Missing data	✓	✗	✓	✗	✗
Survival	✓	✗	✗	✗	✗
Nested	✓	✓	✗	✓	✓
Temporal	✓	✓	✗	✗	✗
Additional Functionality					
Exports data	✓	✓	✓	✓	✗
In built power analysis	✗	✗	✗	✓[e]	✓[e]
Simulate from an existing analysis model	✗	✗	✗	✓[e]	✓[e]
Specify custom model equation	✓	✗	✗	✗	✗

[1] Allegue *et al.* 2017; [2] DeBruine 2023; [3] Green & MacLeod 2016; [4] Martin *et al.* 2011

[a] max two levels; [b] max two predictors; [c] max one predictors; [d] max two responses; [e] only from lme4

a) Crossed

```r
make_structure(
  structure = "year(3) + individual(3)",
  repeat_obs = 2
)
```

	year	individual
1	1	1
2	1	1
3	1	2
4	1	2
5	1	3
6	1	3
7	2	1
8	2	1
9	2	2
10	2	2
11	2	3
12	2	3
13	3	1
14	3	1
15	3	2
16	3	2
17	3	3
18	3	3

b) Nested

```r
make_structure(
  structure = "year(3)/individual(3)",
  repeat_obs = 2
)
```

	year	individual
1	1	1
2	1	1
3	1	2
4	1	2
5	1	3
6	1	3
7	2	4
8	2	4
9	2	5
10	2	5
11	2	6
12	2	6
13	3	7
14	3	7
15	3	8
16	3	8
17	3	9
18	3	9

Figure 1: Examples of balanced data structure generation in `squidSim` using the `make_structure()` function. a) shows a crossed data structure, in which each of the 3 individuals is present in each of the 3 years. b) shows a nested data structure, in which three different individuals are present in each of three years (i.e. individual nested within years). In both examples, there are two observations for each combination.

4 Simulating Data

The heart of the squidSim R package is the `simulate_population()` function, which simulates data from a given data structure. Importantly, all underlying data are simulated from multivariate normal distributions, which aligns with the assumptions of typical GLMMs (note this is on the latent scale for non-Gaussian GLMMs, see section 4.3). To simulate data, users provide the `simulate_population()` function with a data structure (either a sample size, given to the `n` argument, or a `data.frame` containing a hierarchical data structure given to the `data_structure` argument), a list of parameters (`parameters` argument), and various other optional arguments that facilitate more complex simulations. In many scenarios, researchers will want to simulate many datasets under the same parameter sets. This can be easily achieved by specifying the number of datasets in the `n_pop` argument. The `simulate_population()` function generates a `squidSim` object, which stores the simulated datasets, as well as all the information about the simulation (see Reproducibility section below). The `get_population_data()` can then return the simulated data from the `squidSim` object, alongside the data structure.

4.1 Basic functionality

The key to using `simulate_population()` is matching the `parameters` list with a model equation. To demonstrate this, we will take the example of a linear mixed model:

$$
\begin{aligned}
y_{ijk} &= \beta_0 + \boldsymbol{x}_i\boldsymbol{\beta}_{\boldsymbol{x}} + \boldsymbol{w}_j\boldsymbol{\beta}_{\boldsymbol{w}} + u_k + \beta_3 x_{1,i}x_{2,i} + \epsilon_{ijk} \\
\boldsymbol{x}_i &\sim \mathcal{N}(\boldsymbol{\mu}_x, \Sigma_x) \\
\boldsymbol{w}_j &\sim \mathcal{N}(\boldsymbol{\mu}_w, \Sigma_w) \\
u_k &\sim \mathcal{N}(0, \sigma^2_u) \\
\epsilon_{ij} &\sim \mathcal{N}(0, \sigma^2_\epsilon)\,.
\end{aligned}
$$

Here, the response variable (y) is a function of the intercept (β_0), some observation-level predictors ($\boldsymbol{x}_i$; note that bold symbols are used to represent vectors) scaled by a vector of regression slopes $\boldsymbol{\beta_x}$, some predictors ($\boldsymbol{w}$) that vary at level j (e.g. among individuals) scaled by a vector of regression slopes $\boldsymbol{\beta_w}$, some 'random' effects (u) varying at level k (e.g. among years), an interaction between two observation level effects ($x_1 x_2$), and some residual variation (ϵ). $\boldsymbol{x}_i$, $\boldsymbol{w}_j$, u_k and ϵ_i are all drawn from multivariate normal distributions, with means $\boldsymbol{\mu}_x$, $\boldsymbol{\mu}_w$, and 0 and variance covariance matrices Σ_x, Σ_w, σ^2_u and σ^2_ϵ, respectively.

The parameters are specified as a set of nested lists, with a component for each of these parts of the equation as we show in Figure 2. The intercept (β_0) is provided as a single number (red area of Figure 2), or a vector of intercepts for a multi-response model (see Multivariate section below). The residual variance (σ^2_ϵ) parameter (`vcov`; the yellow area at the bottom of Figure 2) must always be specified; this parameter will also be a single number (unless there are multiple response variables). Observation-level predictors (x) can be simulated by adding an `observation` component to the parameters list (dark blue area in Figure 2). These predictors are simulated from a multivariate normal distribution using inputted `mean` and `vcov` parameters, the latter providing the variance-covariance matrix of the predictors (Σ_x). To generate the response, these predictors are scaled by the `beta` parameters (i.e. the regression slopes), and added together to create the response. The `mean`, `vcov` and `beta` parameters do not have to be specified, and have sensible default values (`mean=0`, `vcov=`I and `beta=1`, where I is an identity matrix). In Figure 2, we have specified `vcov` as a vector rather than a matrix; `simulate_population()` interprets this to be the variances (i.e. the diagonal of the variance-covariance matrix), and assumes the respective covariances are 0. If we have no complex data structure (i.e. everything varied at the level of the observation, with no w_j or u_k in the above equation), we could specify a single sample size in the argument `n`, rather than inputting a `data.frame` to the `data_structure` argument. We have also added the `names` argument to the individual and observation lists, resulting in the simulated variables having those names.

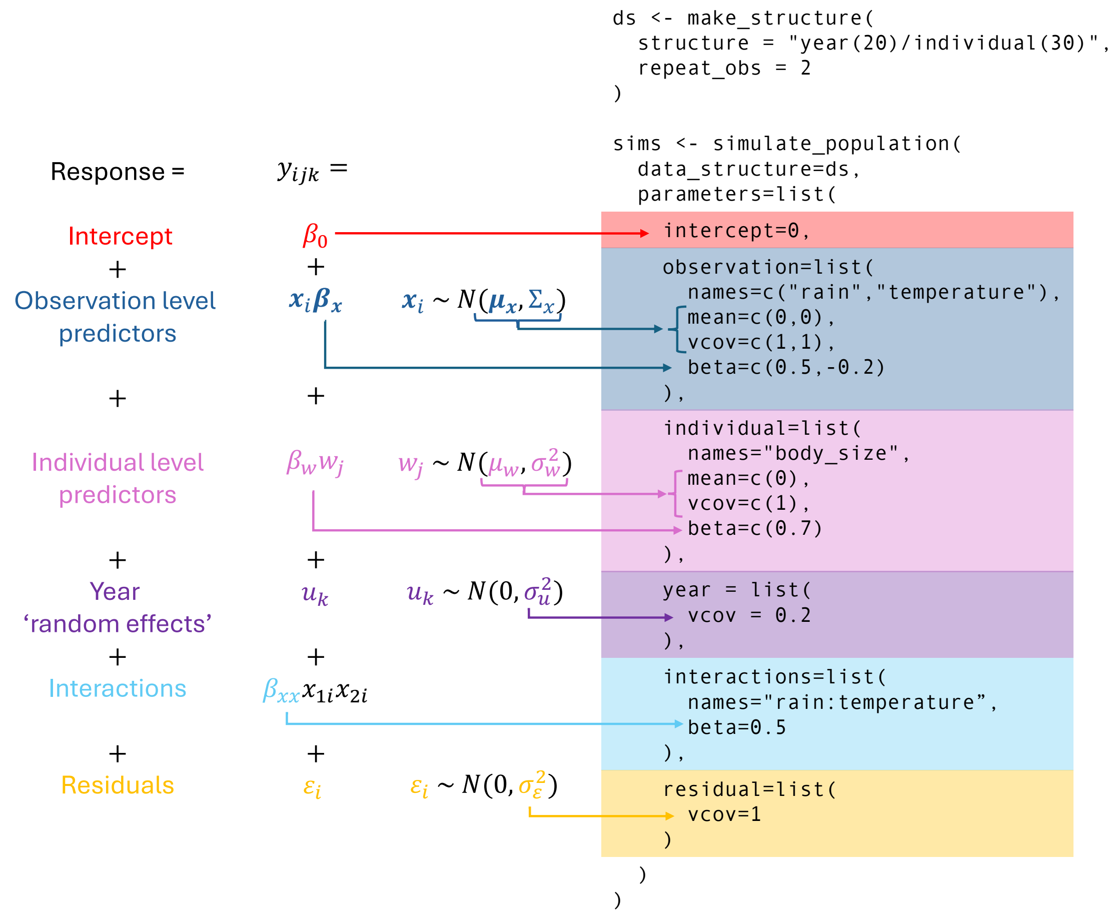


Figure 2: Demonstration of the modular structure of the `simulate_population()` function in `squidSim`. The figure shows the link between the verbal model, the model equation and the `squidSim` code, with the different colours showing how the different components map onto each other.

If there is a data structure, predictors can then be simulated at each hierarchical level that exists in the data structure. For example, a researcher might collect multiple measures per individual, and so some predictors (w) vary at the level of the individual (j), for example, body size, whereas other predictors (x) might vary at the level of the observation (i), for example the weather during a focal measurement. For each hierarchical level, an additional list in the parameter list code is specified, with the name exactly matching the corresponding column name in the data structure (e.g. variation in body size among individuals in the pink area in Figure 2). If a data structure is specified, then `n` is no longer needed, and is taken to be the number of rows in the data structure.

Random effects are simulated in a similar way. From the perspective of simulating data, there is no distinction between simulating random effects and simulating a predictor varying at a particular hierarchical level, as random effects are essentially just unobserved predictors, which your analysis model is estimating. Thus, they have the same general format (purple area in Figure 2). These random effects (u) can be simulated simply by specifying only the `vcov` parameter; the `beta` and `mean` parameters will default to 1 and 0, respectively. This is consistent with how random effects are typically described in E&E.

All the components of the parameters list (intercept, observation, and those linked to the data structure; intercept and year in Figure 2) are additive. Multiplicative elements can be specified as interactions between predictors, by adding an `interactions` list to the `parameters` list (light blue in Figure 2). Quadratic effects can be added in a similar way (a quadratic is just an interaction between a trait and itself).

244

Box 1: Worked Example - Random Regression

In evolutionary ecology, we are often interested in how a relationship varies across some hierarchical level, for example, when studying whether phenotypic plasticity varies among individuals ('IxE') or among genotypes ('GxE'). Typically, such questions are modelled using a random regression (i.e. random slopes) model. Random slopes represent an interaction between variables at different hierarchical levels. In a statistical model, one of these variables (the random slopes) is an unobserved variable (e.g. some property of the individual), which the model estimates (for which there is no 'main effect', which is why beta=0 for the slope variable in Figure B1). When simulating data, there is no distinction between observed and unobserved variables, and so we code both variables in a similar way. Here, we take the example of among-individual variation in the aggressiveness of female Ural owls (*Strix uralensis*) in response to the change in prey density (Δ prey) between subsequent years, shown in Kontiainen *et al.* 2009. In this study, the authors found an overall positive effect of Δ prey (0.13; when the predictor was scaled to have zero mean and unit variance), variation among individual intercepts (σ^2_{u1}=1.3) and also among individual slopes (σ^2_{u2}=0.17), with a correlation between intercepts and slopes of 0.45. In Figure B1, we use these parameters as the basis for our simulation. This simulated data could be used for many purposes, including a power analysis for future studies or an assessment of model performance.

a)
$$y_{ijk} = \beta_0 + \beta_x x_i + u_{1j} + \beta_x x_{1i} u_{2j} + \varepsilon_i$$
$$x_i \sim N(\mu_x, \sigma^2_x)$$
$$\boldsymbol{u_j} \sim N(\boldsymbol{0}, \Sigma_u)$$
$$\varepsilon_i \sim N(0, \sigma^2_\varepsilon)$$

b)
```
sims <- simulate_population(
  seed= 1015,
  data_structure=ds,
  response_name="aggression",
  parameters=list(
    intercept=0,
    observation=list(
      names="delta_prey",
      beta=0.13
    ),
    individual=list(
      names=c("ind_intercepts","ind_slopes"),
      vcov=matrix(c(1.3,0.45,0.45,0.17),nrow=2),
      beta=c(1, 0)
    ),
    interactions=list(
      names="delta_prey:ind_slopes",
      beta=1
    ),
    residual=list(
      vcov=2
    )
  )
)
sim_data <- get_population_data(sims)
head(sim_data)
```

c)
```
##   aggression ind_intercepts   ind_slopes  delta_prey    residual delta_prey:ind_slopes individual squid_pop
## 1 -1.4290288      -0.155527 -0.004236808  0.19393800 -1.29789202          -0.000821678          1         1
## 2 -0.2505148      -0.155527 -0.004236808 -1.30425660  0.06903969           0.005525885          1         1
## 3 -0.7156069      -0.155527 -0.004236808  0.63493096 -0.63993079          -0.002690080          1         1
## 4 -2.1064867      -0.155527 -0.004236808  0.03152845 -1.95492475          -0.000133580          1         1
## 5 -1.5971451      -0.155527 -0.004236808 -1.24192082 -1.28543013           0.005261780          1         1
## 6 -0.1796857      -0.155527 -0.004236808  1.02422065 -0.15296790          -0.004339426          1         1
```

d)

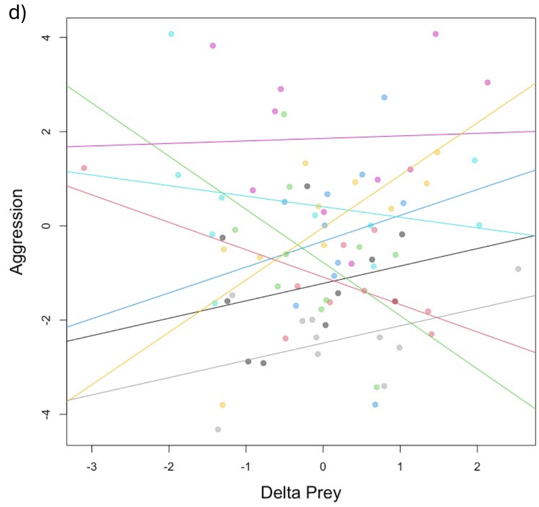


Figure B1: Simulating random slopes data using `squidSim`. We start with the model equation a) which we translate into `squidSim` code b) and input the parameter values (see text in Box 1). We then view the output c) and plot simulated data d).

4.2 Simulating multiple responses

Researchers may want to generate structured data with multiple response variables. This kind of data is common in quantitative genetics when investigating genetic correlations between multiple traits (Kruuk, 2004), and in behavioural ecology when considering covariance in behavioural traits among and within individuals (Dingemanse & Dochtermann, 2013). In such cases, we can simulate from a multi-response (or multivariate) model:

$$
\begin{aligned}
\boldsymbol{y}_{ij} &= \boldsymbol{\beta}_0 + \boldsymbol{x}_i B_x + \boldsymbol{u}_j + \boldsymbol{\epsilon}_{ij} \\
\boldsymbol{x}_i &\sim \mathcal{N}(\boldsymbol{\mu}_x, \Sigma_x) \\
\boldsymbol{u}_j &\sim \mathcal{N}(\mathbf{0}, \Sigma_u) \\
\boldsymbol{\epsilon}_{ij} &\sim \mathcal{N}(\mathbf{0}, \Sigma_\epsilon)\,,
\end{aligned}
$$

where $\boldsymbol{y}_{ij}$ is a vector of responses of length q for observation ij, $\boldsymbol{\beta}_0$ is a vector of intercepts of length q (number of responses), B_x is a $p*q$ matrix of βs (where p is number of predictors) relating each predictor to each response, and Σ_u and Σ_ϵ are $q*q$ variance-covariance matrices for the among-group (e.g. individual) effects and residuals across the different responses. We show how this relates to `squidSim` code in Figure 2.

4.3 Additional Functionality

Many more complex data structures in E&E are characterised by correlations between observations, such as genetic and phylogenetic effects, and spatial and temporal autocorrelations. These data structures can often be captured by a correlation matrix at a specific hierarchical level. Generally, data can be simulated from any such correlated data structure using `squidSim`, by passing a covariance matrix to the `cov_str` argument of `simulate_population()`. We discuss a few specific examples here.

Following on from the original `squid` R package (Allegue *et al.*, 2017), `squidSim` allows different temporal structures to be simulated, such as linear and cyclical environmental effects (out-

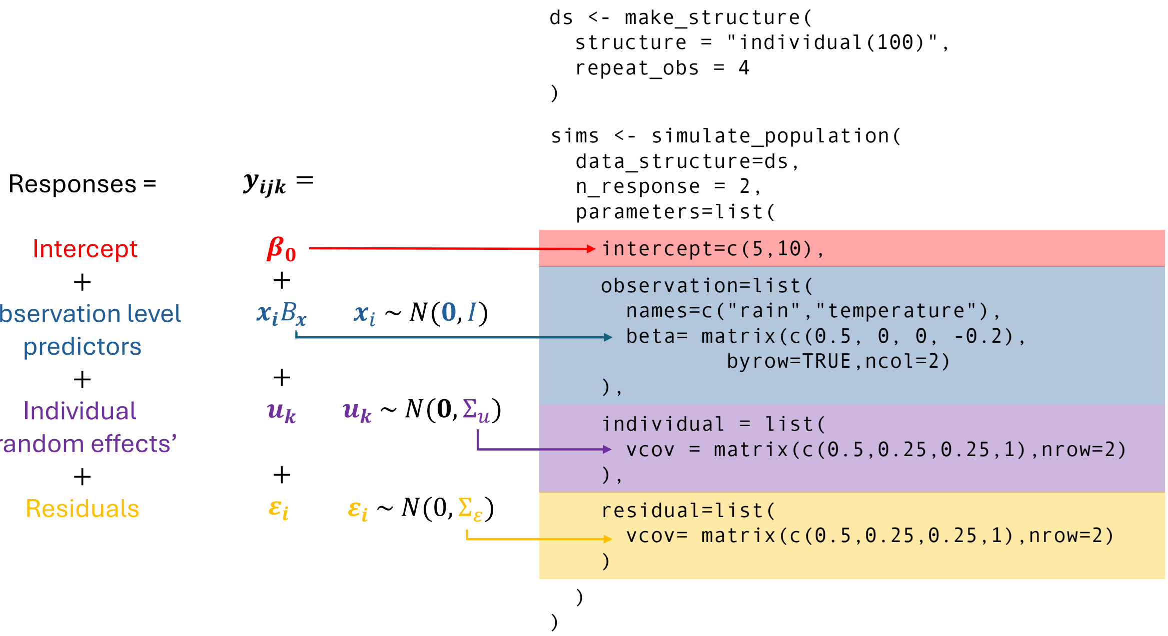


Figure 2: Demonstration of simulating multi-response data with the `simulate_population()` function in `squidSim`. The figure shows the link between the verbal model, the model equation and the `squidSim` code, with the different colours showing how the different components map onto each other. Here we simulated two response variables, with two observation level predictors, each with an effect on one response variable.

lined at https://squidgroup.org/squidSim_vignette/6-temporal-and-spatial-effects.html). Temporal and spatial auto-correlation can be simulated by inputting spatial/temporal correlation matrices to the `cov_str` argument of `simulate_population()`. These correlation matrices can be generated from existing temporal or spatial data using, for example, the `corClasses` functions in the `nlme` R package (Pinheiro & Bates, 2025).

`squidSim` utilises the functionality of the `MCMCglmm` R package (**??**) to simulate additive genetic and phylogenetic effects (assuming Brownian motion). The simplest way to simulate additive genetic effects is to provide the `pedigree` argument in `simulate_population()` with a list, including a three-column pedigree (individual, dam, sire) and a vector identifying which grouping factor(s) in the data structure this links to. This generates additive genetic effects, with a covariance structure determined by the relatedness between individuals, described by the relatedness matrix. Researchers increasingly use genomic data to generate genomic relatedness matrices (GRMs). `squidSim` can also simulate additive genetic effects using these, by passing a GRM to the `cov_str` argument in `simulate_population()`. Similarly, non-additive genetic variance, such as dominance variance, can be simulated by passing the relevant matrix to the `cov_str` argument. Dominance and epistasis matrices can be generated using the `nadiv` R package (Wolak, 2012). Phylogenetic effects can be similarly simulated by passing a phylogeny (as a `phylo` object) to the phylogeny argument of `simulate_population()`.

To generate non-Gaussian data, such as survival, sex ratio, reproductive success, and counts of organisms or behaviours, `simulate_population()` can simulate from Binomial (specifically Bernoulli) and Poisson distributions, alongside providing different link functions (log and inverse for Poisson, and logit, probit and complementary log-log (cloglog) for binomial). These can be specified with the `family` and `link` arguments, respectively, to `simulate_population()`. It is important to note that the data is simulated as multivariate normal on the latent scale, and so the parameters relate to this latent scale and not the observed scale (i.e. not to the counts or proportions directly). In this way, the simulation matches the output of a GLMM. For a good guide to GLMMs and transforming across scales, see de Villemereuil *et al.* (2018); **?**. To aid interpretation, we also provide two functions that help transform distributions between normal and log scales (`lat2exp()` and `exp2lat()`), and show examples of transformation across several scales in the vignette (https://squidgroup.org/squidSim_vignette/1.6-nonGaussian.html). Simulating non-Gaussian data is demonstrated in the example in Box 2.

`squidSim` can also be used to generate data with an observation process, such as species occupancy/abundance or mark-recapture data by simulating two response variables, one for the process of interest, (e.g., whether a species is present), and one for the observation process (e.g., whether a species is observed conditional on being present). The responses can then be easily combined (typically through multiplication) to get the 'observed' data. In this way, a researcher can build a complex structure for both processes. A simple version of this can also be produced using the sampling functions (see Section 5 below, and demonstration in Box 2).

Many datasets have more complex model equations than the default structure of `simulate_population()`

allows (i.e. something more complex than a strictly additive model). We therefore have an additional `model` argument in `simulate_population()`, that allows the custom specification of a model equation. One example of this has already been used in a simulation study on maternal genetic effects (Pick *et al.*, 2024), which requires more complex indexing than `squidSim` allows by default (`https://squidgroup.org/squidSim_vignette/4.4-IGE.html`).

5 Sampling

After simulating the data, a researcher may want to derive certain observed data structures, or vary the data structure in a systematic way (e.g. to explore different study designs or to investigate the effect of different sample sizes). Sampling in `squidSim` is different from simply inputting different data structures. The output of `simulate_population()` retains the original simulated full dataset(s), as well as the sampled ones, meaning that the effects of down-sampling or missing data can be investigated, relative to the full dataset. When sampling functions are used, the sampled data can be returned using the `get_sample_data()` function.

Currently, `squidSim` allows for four different sampling designs. 'Nested' sampling allows the user to specify a range of different sample sizes across different nested hierarchical levels. 'Temporal' sampling allows the user to specify different sampling schemes through time. 'Missing' sampling allows the user to generate the 3 different missing data types: Missing Completely at Random (MCAR), Missing at Random (MAR) and Missing Not at Random (MNAR), through the specification of an equation that controls missingness. This sampling can also be used to get stochastic unbalanced data structures - i.e. to insert uncertainty into the data structure, which could mimic different types of uncertainty due to how data are collected in the field. Finally, 'Survival' sampling subsets survival data for the period an individual is alive (i.e. censors observations after an individual has died). This can used for the generation of data for survival analysis.

330

Box 2: Worked Example - Mark-Recapture data

Here, we present a more advanced example combining many features of `squidSim`. Mark-recapture data are common in conservation and population ecology, and are often characterised by two underlying Bernoulli processes; the probability that an individual survived and, conditional on that, the probability of observing them (e.g. recapture). In the `squidSim` framework, this process can be simulated using a Bernoulli multi-response model, with a response variable for survival and another for observation. This allows users to simulate predictor variables, random effects, etc., for each process. Here, we use the example in Kéry & Schaub (2012, chapter 7) of mark-recapture data in little owls (*Athene Noctua*). We assume a mean survival of 0.65 and a negative effect of winter severity on the latent scale of -0.3. The winter severity index is standardized (mean = 0, variance = 1). We simulate additional temporal variation not explained by winter severity, with a variance of 0.2, and a recapture probability of 0.4. To create a realistic mark-recapture dataset, we used survival sampling to restrict observations to when an individual was alive. The data were then subset for when individuals were observed. Such simulated data could be used for many purposes, for example assessing potential biases introduced by imperfect detection.

Box 2 continued

a)

$$p_i \sim Bern(\psi)$$
$$y_{it} \sim Bern(\phi_{it})$$
$$logit(\phi_{it}) = \beta_0 + \beta_x x_t$$
$$x_t \sim N(\mu_x, \sigma_x^2)$$

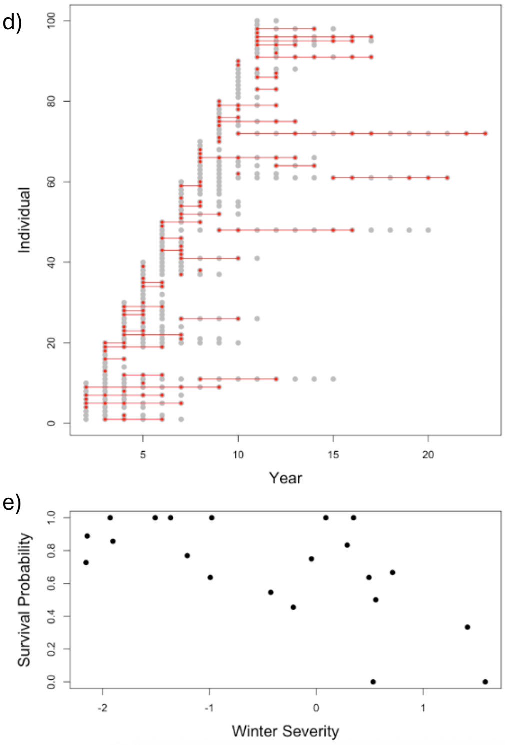
b)

```
ds <- make_structure(paste0(
"birth_year(10)/individual(10) + age(20)"))
ds$year <- ds$age + ds$birth_year

sim_dat<-simulate_population(
  seed=2056,
  data_structure = ds,
  n_response=2,
  response_names = c("survival","recapture"),
  parameters = list(
    intercept=c(qlogis(0.65),qlogis(0.4)),
    individual=list(
      names="winter",
      beta=matrix(c(-0.3,0), ncol=2))
    ),
    residual=list(
      vcov=c(0,0)
    )
  ),
  family= "binomial",
  link="logit",
  sample_type='survival',
  sample_param=list(
    y = "survival",
    ID = "individual",
    age ="age",
    death=0,
    all=TRUE)
)
dat<-get_sample_data(sim_dat)
head(dat)
```

d)

e)



c)

```
##   survival recapture      winter residual1 residual2 birth_year individual age year squid_pop
## 1        1         0  0.08825017         0         0          1          1   1    2         1
## 2        1         1 -0.04615645         0         0          1          1   2    3         1
## 3        1         1 -1.90361442         0         0          1          1   3    4         1
## 4        1         0 -1.20793229         0         0          1          1   4    5         1
## 5        1         1 -0.21630228         0         0          1          1   5    6         1
## 6        0         0  0.55559478         0         0          1          1   6    7         1
```

Figure B2: Simulating Mark-recapture data using squidSim. In this example, we started with the model equation a) which we translated into squidSim code b) and inputted the parameter values (see text in Box 1). This generated the data, part of which are shown in c) and the whole data set is plotted in d & e. In d), grey points show points where an individual was alive, red points show when an individual was captured, and red lines show the period over which an individual was known to be alive. e) shows the simulated negative relationship between winter weather and survival

6 Reproducibility

A major motivation for `squidSim` is to increase the generalisability and reproducibility of simulations. As shown above, `squidSim` can simulate many different kinds of data in standardised manner using the `simulate_population()`, where the parameterisation relates directly back to the model equation.

Running the simulate_population() generates a squidSim object. This object contains both the simulated data and all the information that was used for the simulation. This means that it is easy to retrieve the parameters used for the simulated data set. Furthermore, using an additional argument (seed), we can set a seed (given starting point) for the (pseudo)random number generators, which means that the simulation can be exactly replicated (as shown in Boxes 1 and 2). If a seed is not set, a random seed is chosen and set internally automatically, and saved with the output allowing exact duplication of the simulated dataset if so desired.

7 The shinySim R package

Another major aim for squidSim is to aid researchers in focussing on the model equations and the parameters of a simulation, rather than the intricacies of coding the simulation. To this end we have further created the shinySim R package with a graphical user interface to help users generate code for simulate_population() (https://github.com/squidgroup/shinySim). A user inputs a data structure to shinySim and then can add elements to the model based on this data structure. The app generates the model equation, shows a breakdown of the variance in the response variable explained by each hierarchical level and predictor variable, and creates the code for the parameter block of simulate_population(). Currently, the shinySim app has less functionality compared with the full range of models that squidSim can produce and covers the models outlined in the 'Basic Functionality' section, but we are constantly updating this with more functionality. Regardless, shinySim provides a good way to get started with simulations, with less focus on coding, and will provide a useful teaching tool.

8 squidSim as a teaching tool

squidSim can be used in several ways to enhance statistics teaching across a wide variety of teaching settings. The SQuID group has employed simulations as a teaching aid in 14

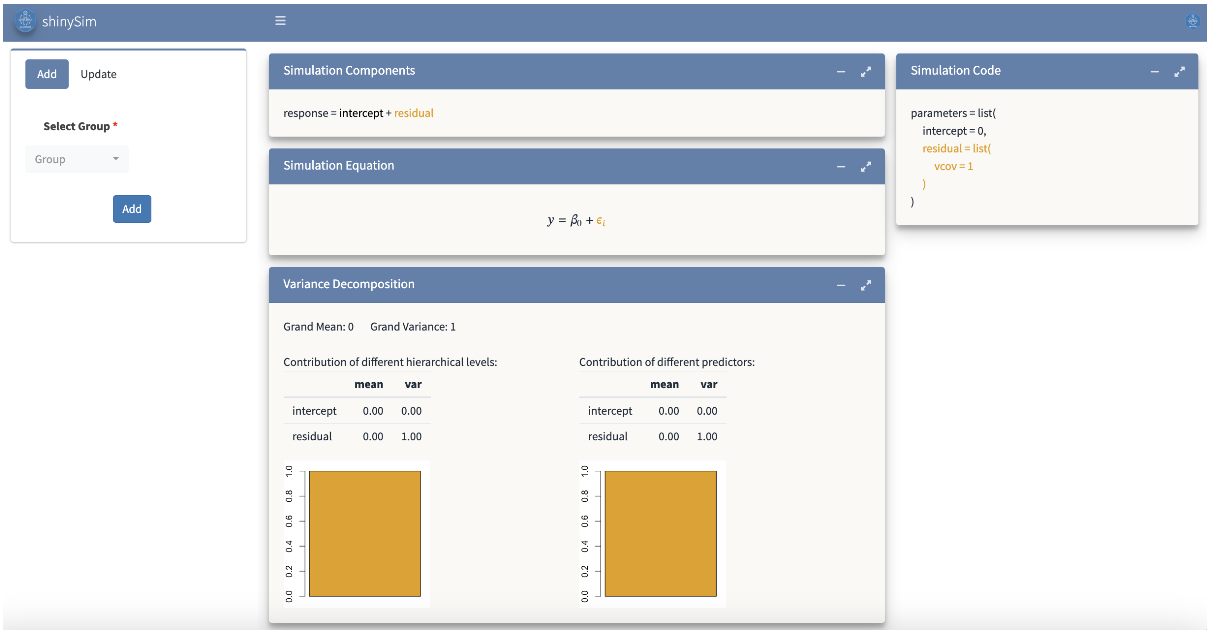


Figure 3: The shinySim interface

statistics workshops with diverse attendees worldwide, specifically using `squidSim` in ten. These workshops were designed to teach students that statistical models are a way to represent hypotheses about specific biological processes. In the workshops, we taught linear mixed models and simulation simultaneously. Use of simulations allow us to cycle through model equations, generating the data, graphical representations of the data, statistical models and outputs. As `squidSim` has an intuitive structure that mirrors statistical equations, students see the same concepts in multiple ways, and learn how model outputs reflect the parameters they have used to generate the data.

A second use of simulated data in our workshops was to create practicals where students either simulate data or were provided with a simulated dataset, and then were challenged to understand what happens when their analysis models were misspecified. This is probably always the case with real data and using simulated data can focus attention on specific types of problems and their solutions. `squidSim` has allowed us and students to easily create complex datasets that allow targeted lessons to be learned, such as the impact of different sampling designs, or leaving predictors out of a model. These practicals helped students gain a richer and more intuitive understanding of what different components of the models are doing.

A final application of squidSim has been to challenge students to simulate data from their own study systems, perhaps as a prelude to assessing sample sizes needed to address their own research questions. One major benefit of this is that abstract notions of the statistical models the students have been learning about immediately gain more traction when linked to their own system. The students also gain experience thinking about statistics in the context of their own research question. Hierarchical models have many moving parts, and simulating data along with retrieving parameter values when the system is their own leads to deeper intuition about how models behave under different conditions and what factors may be limiting their interpretation of their models.

9 Conclusions

In summary, we have shown that squidSim can simulate a variety of data structures and types to address an array of useful problems encountered in E&E. It has flexibility, yet is intuitive in structure, and the addition of the shinySim interface makes doing many types of simulations easier for beginners. A key element is that the coding is standardised and reproducible. We therefore believe that squidSim provides a valuable tool for researchers of all levels of familiarity with simulations and a helpful teaching resource.

Acknowledgements

This is the eighth contribution of the Statistical Quantification of Individual Differences (SQuID) working group, and we'd like to thank the other members of SQuID. JLP and the SQuID group meetings in 2022-24 were funded by a Research Council of Norway INTPART project number 309356 grant to JW. YGA was supported by the Research Council of Norway project number 325826. JW and YGA were also partially supported by the Research Council of Norway (SFF-III 223257/ F50). DR was supported by Discovery grant from the Natural Science and Engineering Council of Canada (NSERC: RGPIN-2020-04745). NAD was supported by NSF award (IOS-2222929) and KLL by NSF award (IOS-2100625). HA was supported by the Natural Sciences and Engineering Research Council of Canada (CGSD3-504399-2017 and BP-598736-2025) and the Fond de Recherche du Québec - Nature et Technologies (FRQNT; 283511). NJD was supported by the German Science Foundation (DI 1694/1-2, DI 1694/5-1).

Author Contributions

Conceptualization: JLP, HA, YAA, DFW

Software: JLP, EIC

Methodology: JLP

Writing - Original Draft: JLP

Writing - Review & Editing: All authors

Supervision: DFW, JW, NJD

Funding Acquisition: JW

10 Conflict of Interest statement

The authors declare no conflict of interest.

11 Data and code availability

All code for the simulated examples are deposited in https://github.com/squidgroup/squidSim_manuscript

References

Allegue, H., Araya-Ajoy, Y.G., Dingemanse, N.J., Dochtermann, N.A., Garamszegi, L.Z., Nakagawa, S., Réale, D., Schielzeth, H. & Westneat, D.F. (2017) Statistical Quantification of Individual Differences (SQuID): an educational and statistical tool for understanding multilevel phenotypic data in linear mixed models. *Methods in Ecology and Evolution*, **8**, 257–267. https://dx.doi.org/10.1111/2041-210X.12659.

Culina, A., Berg, I.v.d., Evans, S. & Sánchez-Tójar, A. (2020) Low availability of code in ecology: A call for urgent action. *PLOS Biology*, **18**, e3000763. https://dx.doi.org/10.1371/journal.pbio.3000763.

de Villemereuil, P., Morrissey, M.B., Nakagawa, S. & Schielzeth, H. (2018) Fixed-effect variance and the estimation of repeatabilities and heritabilities: issues and solutions. *Journal of Evolutionary Biology*, **31**, 621–632. https://dx.doi.org/10.1111/jeb.13232.

DeBruine, L. (2023) faux: Simulation for factorial designs.

Dingemanse, N.J. & Dochtermann, N.A. (2013) Quantifying individual variation in behaviour: mixed-effect modelling approaches. *Journal of Animal Ecology*, **82**, 39–54. https://dx.doi.org/10.1111/1365-2656.12013.

DiRenzo, G.V., Hanks, E. & Miller, D.A.W. (2023) A practical guide to understanding and validating complex models using data simulations. *Methods in Ecology and Evolution*, **14**, 203–217. https://dx.doi.org/10.1111/2041-210X.14030.

Gelman, A. & Hill, J. (2007) *Data Analysis Using Regression and Multilevel Hierarchical Models*. Cambridge University Press, Cambridge.

Green, P. & MacLeod, C.J. (2016) SIMR: an R package for power analysis of generalized linear mixed models by simulation. *Methods in Ecology and Evolution*, **7**, 493–498. https://dx.doi.org/10.1111/2041-210X.12504.

Ihle, M., Pick, J.L., Winney, I.S., Nakagawa, S. & Burke, T. (2019) Measuring Up to Reality: Null Models and Analysis Simulations to Study Parental Coordination Over Provisioning Offspring. *Frontiers in Ecology and Evolution*, **7**, 142. https://dx.doi.org/10.3389/fevo.2019.00142.

Kain, M.P., Bolker, B.M. & McCoy, M.W. (2015) A practical guide and power analysis for GLMMs: detecting among treatment variation in random effects. *PeerJ*, **3**, e1226. https://dx.doi.org/10.7717/peerj.1226.

Kellner, K.F., Doser, J.W. & Belant, J.L. (2025) Functional R code is rare in species distribution and abundance papers. *Ecology*, **106**, e4475. https://dx.doi.org/10.1002/ecy.4475.

Kimmel, K., Avolio, M.L. & Ferraro, P.J. (2023) Empirical evidence of widespread exaggeration bias and selective reporting in ecology. *Nature Ecology & Evolution*, **7**, 1525–1536. https://dx.doi.org/10.1038/s41559-023-02144-3.

Kontiainen, P., Pietiäinen, H., Huttunen, K., Karell, P., Kolunen, H. & Brommer, J.E. (2009) Aggressive Ural owl mothers recruit more offspring. *Behavioral Ecology*, **20**, 789–796. https://dx.doi.org/10.1093/beheco/arp062.

Kruuk, L.E.B. (2004) Estimating genetic parameters in natural populations using the 'animal model'. *Philosophical Transactions of the Royal Society of London Series B: Biological Sciences*, **359**, 873–890. https://dx.doi.org/10.1098/rstb.2003.1437.

Kéry, M. & Royle, J.A., eds. (2020) *Applied Hierarchical Modeling in Ecology: Analysis of Distribution, Abundance and Species Richness in R and BUGS*. Academic Press.

Kéry, M. & Schaub, M., eds. (2012) *Bayesian Population Analysis using WinBUGS*. Academic Press, Boston.

Lotterhos, K.E., Fitzpatrick, M.C. & Blackmon, H. (2022) Simulation Tests of Methods in Evolution, Ecology, and Systematics: Pitfalls, Progress, and Principles. *Annual Review of Ecology, Evolution, and Systematics*, **53**, 113–136. https://dx.doi.org/10.1146/annurev-ecolsys-102320-093722.

Martin, J.G.A., Nussey, D.H., Wilson, A.J. & Réale, D. (2011) Measuring individual differences in reaction norms in field and experimental studies: a power analysis of random regression models. *Methods in Ecology and Evolution*, **2**, 362–374. https://dx.doi.org/10.1111/j.2041-210X.2010.00084.x.

Morris, T.P., White, I.R. & Crowther, M.J. (2019) Using simulation studies to evaluate statistical methods. *Statistics in Medicine*, **38**, 2074–2102. https://dx.doi.org/10.1002/sim.8086.

Nakagawa, S. & Schielzeth, H. (2013) A general and simple method for obtaining R2 from generalized linear mixed-effects models. *Methods in Ecology and Evolution*, **4**, 133–142. ISBN: 2041-210X, https://dx.doi.org/10.1111/j.2041-210x.2012.00261.x.

O'Hara, R.B. (2009) How to Make Models Add Up — A Primer on GLMMs. *Annales Zoologici Fennici*, **46**, 124–137. https://dx.doi.org/10.5735/086.046.0205.

Pick, J.L., Kasper, C., Allegue, H., Dingemanse, N.J., Dochtermann, N.A., Laskowski, K.L., Lima, M.R., Schielzeth, H., Westneat, D.F., Wright, J. & Araya-Ajoy, Y.G. (2023a) Describing posterior distributions of variance components: Problems and the use of null distributions to aid interpretation. *Methods in Ecology and Evolution*, **14**, 2557–2574. https://dx.doi.org/10.1111/2041-210X.14200.

Pick, J.L., Khwaja, N., Spence, M.A., Ihle, M. & Nakagawa, S. (2023b) Counter culture: causes, extent and solutions of systematic bias in the analysis of behavioural counts. *PeerJ*, **11**, e15059. https://dx.doi.org/10.7717/peerj.15059.

Pick, J.L., Walling, C.A. & Kruuk, L.E.B. (2024) Simple maternal effect animal models provide biased estimates of additive genetic and maternal variation. *EcoEvoRxiv*. https://dx.doi.org/https://doi.org/10.32942/X2V33J.

Pinheiro, J. & Bates, D. (2025) nlme: Linear and Nonlinear Mixed Effects Models.

Schielzeth, H., Dingemanse, N.J., Nakagawa, S., Westneat, D.F., Allegue, H., Teplitsky, C., Réale, D., Dochtermann, N.A., Garamszegi, L.Z. & Araya-Ajoy, Y.G. (2020) Robustness of linear mixed-effects models to violations of distributional assumptions. *Methods in Ecology and Evolution*, **11**, 1141–1152. https://dx.doi.org/10.1111/2041-210X.13434.

Stoffel, M.A., Nakagawa, S. & Schielzeth, H. (2017) rptR: repeatability estimation and variance decomposition by generalized linear mixed-effects models. *Methods in Ecology and Evolution*, **8**, 1639–1644. https://dx.doi.org/10.1111/2041-210X.12797.

van Benthem, K.J., Bruijning, M., Bonnet, T., Jongejans, E., Postma, E. & Ozgul, A. (2017) Disentangling evolutionary, plastic and demographic processes underlying trait dynamics: a review of four frameworks. *Methods in Ecology and Evolution*, **8**, 75–85. https://dx.doi.org/10.1111/2041-210X.12627.

Westneat, D.F., Araya-Ajoy, Y.G., Allegue, H., Class, B., Dingemanse, N., Dochtermann, N.A., Garamszegi, L.Z., Martin, J.G.A., Nakagawa, S., Réale, D. & Schielzeth, H. (2020) Collision between biological process and statistical analysis revealed by mean centring. *Journal of Animal Ecology*, **89**, 2813–2824. https://dx.doi.org/10.1111/1365-2656.13360.

Wolak, M.E. (2012) nadiv : an R package to create relatedness matrices for estimating non-additive genetic variances in animal models. *Methods in Ecology and Evolution*, **3**, 792–796. https://dx.doi.org/10.1111/j.2041-210X.2012.00213.x.