Received: 30 April 2024 | Accepted: 10 October 2024

DOI: 10.1111/2041-210X.14452

Methods in Ecology and Evolution

APPLICATION

calibrar: An R package for fitting complex ecological models

Ricardo Oliveros-Ramos[1,2,3] | Yunne-Jai Shin[2,3]

[1]Instituto del Mar del Perú (IMARPE), Callao, Peru

[2]MARBEC, Institut de Recherche Pour le Développement (IRD), Université de Montpellier, IFREMER, CNRS, Montpellier, France

[3]Department of Biological Sciences, Private Bag X3, Marine Research (MA-RE) Institute, University of Cape Town, Rondebosch, South Africa

Correspondence
Ricardo Oliveros-Ramos
Email: ricardo.oliveros@ird.fr

Funding information
Biodiversa and Belmont Forum, Grant/Award Number: ANR-18-EBI4-0003-01(SOMBEE); European Union's Horizon 2020 Research and Innovation Programme, Grant/Award Number: 869300 (FutureMARES) and 817578 (TRIATLAS); Horizon Europe Research and Innovation Programme, Grant/Award Number: 101060072 (ActNow); France Filière Pêches; Pew Marine Fellows Programme; Fondation Pour la Recherche sur la Biodiversité, Grant/Award Number: APP-SCEN-2010-II (EMIBIOS); "Support and training of scientific communities of the South" Department of IRD

Handling Editor: Robin James Boyd

Abstract

1. The fitting or parameter estimation of complex ecological models is a challenging optimisation task, with a notable lack of tools for fitting complex, long runtime or stochastic models.
2. calibrar is an R package that is dedicated to the fitting of complex models to data. It is a generic tool that can be used for any type of model, especially those with non-differentiable objective functions and long runtime, including individual or agent based models.
3. calibrar supports multiple phases and constrained optimisation, includes 20 optimisation algorithms, including derivative-based and heuristic ones. It supports any type of parallelisation, the capability to restart interrupted optimisations for long runtime models and the combination of different optimisation methods during the multiple phases of a calibration.
4. User-level expertise in R is necessary to handle calibration experiments with calibrar, but there is no need to modify the model's code, which can be programmed in any language. It implements maximum likelihood estimation methods and automated construction of the objective function from simulated model outputs. For more experienced users, calibrar allows the implementation of user-defined objective functions.
5. The package source code is fully accessible and can be installed directly from CRAN.

KEYWORDS
black-box optimisation, calibration, evolutionary algorithms, individual based model, inverse problem, parameter estimation, stochastic model

1 | INTRODUCTION

The ability to achieve accurate parameter estimation has been used as a criterion to assess the usefulness of ecological models (Bartell, 2003). Given a model, the criterion for the selection of the best possible parameter set is the optimisation of an *objective function* with respect to the model parameters (Bolker et al., 2013; Walter & Pronzato, 1997). This objective function is scalar and provides a measure of the error of the model in comparison to the data (e.g. residual sum of squares, log-likelihood), integrating all datasets and considering uncertainty and possible biases in the observations. Once the objective function is properly defined, parameter estimation is essentially an optimisation problem. Parameter estimation, calibration or fitting of ecological models (Jorgensen &

Bendoricchio, 2001) can be a challenging task for optimisation algorithms when dealing with complex models characterised by non-linearity and high dimensionality, as well as low quantity and quality of observed data (Tashkova et al., 2012). To date, there has been limited development of optimisation algorithms and calibration methodologies for complex ecological models that are sufficiently flexible, generic and well-documented for users (Bolker et al., 2013). Additionally, complex ecosystem models can be numerically intensive and require long simulation runs, adding an extra layer of difficulty to their fitting process.

There are some dedicated tools for non-linear parameter estimation, AD Model Builder (ADMB; Fournier et al., 2012) being one of the most robust and fast (Bolker et al., 2013). Among other advantages, ADMB provides support for parameter estimation in multiple phases or "masks", with some parameters temporarily fixed (Nash & Walker-Smith, 1987), which can be of great interest when dealing with complex ecosystem models (Oliveros-Ramos et al., 2017). It also provides support for bound or box constraints (independent fixed bounds on each parameter), which can be helpful for regularising hard optimisation problems (Bolker et al., 2013). However, the model and the objective function itself need to be coded in C++ (using a scripting language), which can be an obstacle for fitting models which have been already implemented in other languages (e.g. Fortran, Java). In addition, it is based in automatic differentiation (AD), which allows to provide accurate estimates of derivatives (Griewank & Corliss, 1992), but does not handle the estimation of parameters of stochastic models for which derivatives cannot be computed.

Parameter estimation methods have been developed for stochastic non-linear models, such as continuous time, finite state Markov models and individual-based models (IBMs), for which the probability of state transitions or the master equation can be written (Ionides et al., 2006; Newman et al., 2009; Ross et al., 2009; Walker et al., 2006). However, many stochastic models at the individual level can only be simulated numerically and are too complex for mathematical analysis and explicit parameter estimation (Black & McKane, 2012). In this case, when it is not possible to compute the likelihood but the model is easy to simulate, Approximate Bayesian Computation (ABC) has been used to estimate parameters of complex models (Csilléry et al., 2012; Sunnåker et al., 2013), but the number of simulations needed is prohibitive for long runtime models. As a result, more attention has been given to the exploration of model behaviour than to a rigorous confrontation with data for complex and computationally intensive models. To solve these issues, metaheuristic algorithms (e.g. evolutionary algorithms, see Wong & Ming, 2019 and Rajwar et al., 2023) have been used (Cropper Jr. & Anderson, 2004; Duboz et al., 2010; Poovathingal & Gunawan, 2010; Tashkova et al., 2012; Travers-Trolet et al., 2013), and have in some cases shown better performance than derivative-based optimisation methods (Tashkova et al., 2012). However, the scientific community lacks generic, open and flexible enough tools for the parameter estimation of different types of models with different degrees of complexity and computational power requirements.

Here we present a novel R package, calibrar, designed for parameter estimation for a wide range of ecological models, including complex and stochastic models. The package combines various optimisation functionalities in a single interface, enabling the implementation of the latest advancements in complex model calibration. The package provides support for multiple sequential phases and box constrained optimisation with the possibility of using several algorithms available in R. In particular, by using a "black-box" approach, the package allows the calibration of models implemented in any programming language. It provides a generic interface with models and allows the construction of the objective function, within R, without requiring any changes in the models' code. Parallel support for computationally intensive models is also provided, and can be used with high performance computing systems in a simple manner, including the capability to restart an unfinished optimisation for models with a long runtime.

2 | GENERAL DESCRIPTION OF THE PACKAGE

The calibrar package is written in R (R Core Team, 2023), and can be installed from CRAN. The purpose of the package is handling the frequentist parameter estimation of complex models, for example using a maximum likelihood approach. However, it can be used for fitting any model, and can also be used with user-defined objective functions. The package can be used for the optimisation of "black-box" scalar functions (Jones et al., 1998), where analytical information about the function to be optimised and the model source code are assumed to be unavailable or impractical to modify (Rios & Sahinidis, 2013). Our approach is hence "non-intrusive", making the model interact with the optimiser, that is the calibrar package, in two ways: (i) receiving the model's parameters to run the model, and (ii) providing the model outputs to be confronted with observed data. The package also helps in the construction of the objective function to be optimised in order to estimate model's parameters (Figure 1).

The package works in a way that minimal expertise in R is necessary to handle the model fitting and the main functionalities are embedded in three functions: calibrate, calibration_data and calibration_ObjFn (see Table 1). The user intervention is mainly required in the construction of the function to run the model and to retrieve the simulation outputs within R (run_model function, Figure 1). However, given R's flexibility and features for data manipulation, it is rather straightforward to develop such a function. Some complex ecological models have dedicated packages oriented to the analysis and simulation of their outputs that could be used to link with calibrar, for example, RNetLogo (Thiele et al., 2012) for IBMs implemented in netLogo or osmose (www.osmose-model.org) for the OSMOSE ecosystem model (Shin & Cury, 2001, 2004). The main function of the package is calibrate, which performs minimisation of the objective function and returns the optimal parameters. It has a similar syntax as the base R optimisation function optim (see Table 2), and is also similar to most

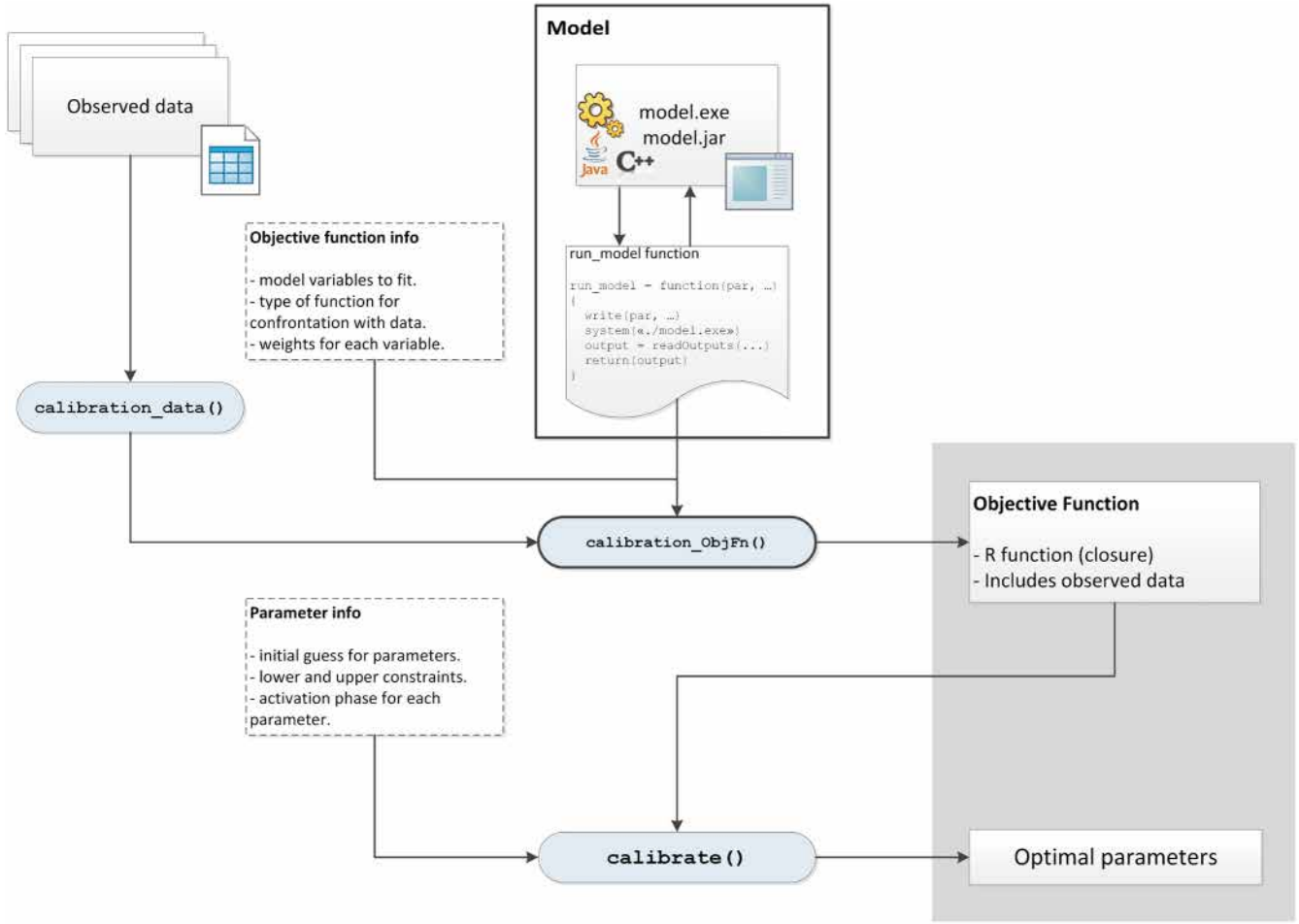

FIGURE 1 Diagram representing the functioning of the calibrar package. The grey area groups the outputs produced by the package (the objective function and the optimal parameters of the model). Rectangles with broken border lines show user inputs which are needed to configure the model fitting process. Blue rounded rectangles show main package functions.

optimisation related functions in other R packages. Additionally, the calibration_data and calibration_ObjFn are provided to simplify the organisation of the observed data and construct the objective function for the calibration, respectively.

In order to create the objective function, the users need to specify some information about the model outputs used for the calibration and how to combine them (Figure 1). More experienced users can create the objective function by directly integrating the run of the model and its comparison with observed data. The details for the creation of the objective function are explained in the next section.

Additionally, the user needs to specify information on the parameters to calibrate. Lower and upper bounds can be provided for each parameter for box constrained optimisation, and this often improves the parameter estimation (Bolker et al., 2013) but unconstrained optimisation is also supported. In case of a multiple phase calibration, the user must indicate the phase of the calibration where the estimation of a parameter must be included. The implementation of multiple phases in the calibration is detailed in the next section.

The calibrar package allows to perform the calibration with a total of 20 different optimisation algorithms allowing box constraints (Table 3), including the ones available in the package stats (R Core Team, 2023) and optimx (Nash & Varadhan, 2011), among others (see Table 3 for details, and Nash, 2014 for a comprehensive introduction to nonlinear parameter optimisation in R). The calibration in multiple phases is available for all the optimisation methods through a sequential update of the objective function for each phase, built-in within the calibrar package. Different optimisation methods can be used for each estimation phase, allowing the combination of different methods during the course of a multi-phase calibration. The default algorithm for deterministic optimisation problems is a version of the L-BFGS-B algorithm implemented in the package Rvmmin (Nash, 2021). For stochastic optimisation problems, the default algorithm is the AHR-ES (Adaptative Hierarchical Recombination Evolutionary Strategy), which designates a novel evolutionary algorithm included in this package. Another focus of this package is the handling of computationally intensive objective functions (e.g. >60 s for evaluation time), including the possibility to resume interrupted optimisation runs from intermediate "restart" files. Long runtime models are not necessarily more complex from the optimisation point of view, but they are more prone to computational errors during their

TABLE 1 Main functions of the calibrar package.

Function	Description	Input arguments	Returned objects
calibrate	Performs a parameter estimation of a model using multiple sequential phases	Starting point for the optimisation (par) and objective function (fn), see Table 2 for more details	An object summarising the calibration results, including convergency status. See the help of the function for more details
calibration_setup	A wrapper for read.csv checking column names and data types for the table with the calibration information	A data.frame with the calibration setup (see Figure 2 and the text for details) and Supporting Information for examples	A data.frame with the necessary information to create the objective function using calibration_objFn. See the help of the function for more details
calibration_data	Creates a list of the observed data with the information provided by its main argument	The returned object from calibration_setup(), plus an optional absolute path to search for the data	A list of the observed values (the data). See the help of the function for more details
calibration_objFn	Creates a new function, to be used as the objective function in the calibration	A function suited to run the model to be calibrated, plus the outputs from calibration_setup() and calibration_data().	A function, integrating the run of the model and the comparison of outputs with observed data. See the help of the function for more details
coef, summary, predict	R S3 methods for visualising the results of the calibration	The object returned by the calibrate() function	

TABLE 2 Main arguments of the calibrate function.

Argument	Description
par	A numeric vector or a list containing numeric vectors. The length of the par argument defines the number of parameters to be estimated (i.e. the dimension of the problem)
fn	The function to be minimised
gr	A function specifying the gradient of fn. Some optimisation methods do not need the gradient, so it will be ignored if provided. If needed and not provided, numerical gradients will be computed
method	The optimisation method to be used, 20 methods are currently available (see Table 3). Different methods are allowed for each calibration phase, see 'phases' argument
upper	Upper bound value(s) for parameters. One value or a vector of the same length as par. If one value is provided, it is used for all parameters. NA means Inf. By default, Inf is used (unconstrained)
lower	Lower bound value(s) for parameters. One value or a vector of the same length as par. If one value is provided, it is used for all parameters. NA means -Inf. By default -Inf is used (unconstrained)
phases	An optional vector of the same length as par, indicating the phase at which each parameter becomes active for the optimisation. If omitted, default value is 1 for all parameters. Negative integers and NA are accepted for phases, both meaning that the parameter will never be active, so it will remain constant throughout the calibration
control	Fine control of the optimisation, see function help for details and Table 3
hessian	Logical value. Should a numerically differentiated Hessian matrix be returned?
replicates	The number of replicates of model run to evaluate the objective function in case of stochastic models. One value or a vector of length max(phases), to specify a different number of replicates for each phase. The default value is 1
parallel	Logical. Use parallel computation? (e.g. for numerical gradients)

execution (e.g. breakdowns or stoppages of calculation servers), so the ability to restart an interrupted optimisation is a very useful feature for this type of problems. This feature is provided for the two algorithms used by default (L-BFGS-B and AHR-ES) and for the Hooke-Jeeves derivative-free minimisation algorithm (HJK), a pattern search type algorithm similar to the Nelder–Mead algorithm used by default in optim(). Additionally, parallel computation of numerical derivatives has been implemented for all derivative-based optimisation methods, greatly improving the time needed for the optimisation.

3 | CREATING THE OBJECTIVE FUNCTION

The main purpose of the package is to fit complex models to data. In order to solve a calibration problem, we first need to define the objective function. While more experienced users can write explicitly the objective function for their model fitting, the calibrar package provides help functions aimed at simplifying the process of creating the objective function. A non-intrusive black-box optimisation approach is adopted, which means that the computer code of the model to be calibrated does not need modification, but the model can be evaluated

TABLE 3 Optimisation methods available within the calibrar package.

Algorithm	Package	Reference	Method	Notes
I. Derivative-based (local) methods				
L-BFGS-B 3.0	lbfgsb3c	Fidler et al. (2020)	'lbfgsb3'	Default for deterministic objective functions. R interface to L-BFGS-B.3.0 (Morales & Nocedal, 2011) Fortran library
Algorithm 21	Rvmmin	Nash (2021)	'Rvmmin'	Variable metric method with box constraints
Algorithm 22	Rcgmin	Nash (2022)	'Rcgmin'	Conjugate gradient method with box constraints
ppg	BB	Varadhan and Gilbert (2009)	'spg'	Spectral projected gradient method
nlminb	Stats	R Core Team (2023)	'nlminb'	Quasi-newton method with box constraints, R implementation of PORT routines
L-BFGS-B	Stats	R Core Team (2023)	'L-BFGS-B'	Original implementation in the stats package
II. Derivative-free (local) methods				
nmkb	Dfoptim	Varadhan et al. (2023)	'nmkb'	Nelder–Mead algorithm with box constraints
hjkb	Dfoptim	Varadhan et al. (2023)	'hjkb'	Hooke-Jeeves derivative-free minimisation algorithm
mads	Dfoptim	Varadhan et al. (2023)	'mads'	Mesh Adaptive Direct Searches (MADS) algorithm for derivative-free and black-box optimisation
hjn	Optimx	Nash and Varadhan (2011)	'hjn'	Hooke and Jeeves Pattern Search Optimisation
BOBYQA	Minqa	Bates et al. (2023)	'bobyqa'	Implementation of the BOBYQA algorithm trusted-region method
III. Heuristic (global) methods				
AHR-ES	Calibrar	This paper	'AHR-ES'	Default for stochastic objective functions. Adaptative hierarchical recombination evolutionary strategy
CMA-ES	Cmaes	Trautmann et al. (2011)	'CMA-ES'	Covariance matrix adaptation evolutionary strategy (Hansen & Ostermeier, 2001)
SANN	Stats	R Core Team (2023)	'SANN'	Simulated Annealing implemented in stats::optim (method='SANN')
genSA	GenSA	Xiang et al. (2013)	'genSA'	Generalised Simulated Annealing
DE	DEoptim	Mullen et al. (2011)	'DE'	Differential evolution
soma	Soma	Clayden (2022)	'soma'	Implementation of the Self-organising Migrating Algorithm
genoud	Rgenoud	Mebane Jr. and Sekhon (2011)	'genoud'	Genetic optimisation using derivatives
pso	Psoptim	Ciupke (2016)	'PSO' 'PSO2007' 'PSO2011'	Particle swarm optimisation (PSO), it includes two algorithms, 'PSO2007' and 'PSO2011'. Using method='PSO' will use the default 'PSO2007' (see package help for details) but both can be specified
hybridPSO	Psoptim	Ciupke (2016)	'hybridPSO'	The method='hybridPSO' will use the hybrid method combining PSO and BFGS implemented in the pso package

for a given set of parameters. For this purpose, users need to write an R function to (i) write a set of parameters in the format the model is able to read, (ii) run the model with this set of parameters and (iii) read the model outputs back into R (Figure 2a). The output of this run_model function is expected to be a list, each element being one of the variables to calibrate. If the model is already implemented in R, the construction of this function is very simple. Additionally, R facilities to process and analyse data in different formats allow model outputs to be handled independently of the language used for coding the model.

After the construction of the run_model function, the second step consists in providing information for the construction of the objective function. Each variable listed in the output of run_model needs to be documented in the objective function (e.g. providing the name of 'variable' and 'type', Figure 2b). This information should be provided as a data.frame, and will be used as an argument for the functions calibration_data and calibration_ObjFn. The calibration_data function is expected to read data from the disk, to produce a list with the same structure as the outputs of the run_model function. The function calibration_ObjFn will combine the observed data and the run_model function to create the objective function for the calibration problem (Figure 2b), which in turn will be the fn argument for the calibrate function.

To build the objective function, the 'type' selected for each variable is the function that will combine the observed and simulated

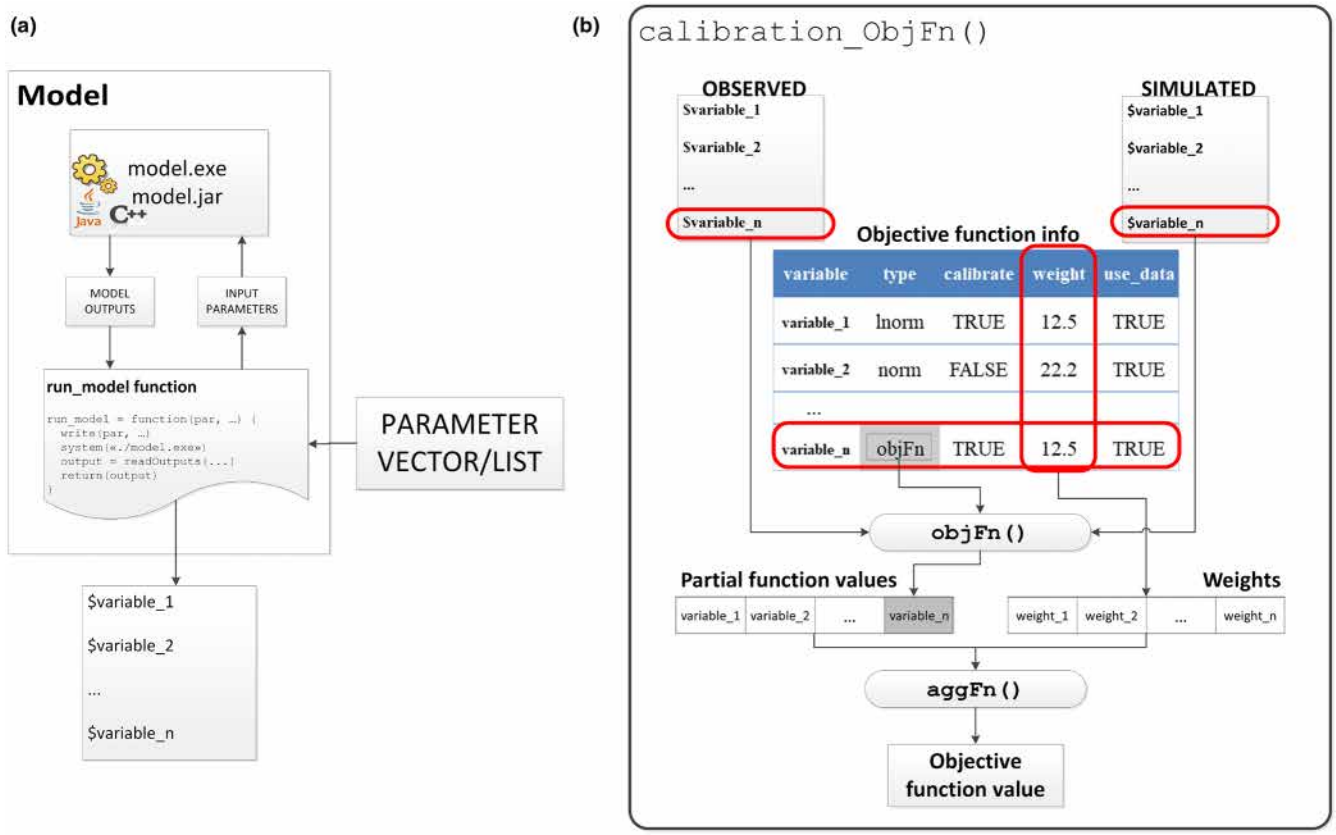

FIGURE 2 (a) Scheme of the link between the model and the calibration. The R function run_model receives a vector or list of parameters to test, writes the parameters in a form that is readable for the model (e.g. via txt or csv files), runs the model (possibly via system) then captures and processes the model outputs. The result of the function is a 'list' object with all the variables to be confronted with observed data. (b) Scheme of the calculation of the value of the objective function for a given set of parameters. For each variable, a partial value of the objective function is calculated by applying the function specified in the column 'type' to observed and simulated values. The final value of the objective function is calculated by applying the aggFn to the partial function values and the weights specified in the 'objective function info' table.

data to produce a scalar value, measuring the fit between the model and the observations. One scalar objective function is produced as in a multi-objective optimisation problem there is no guarantee that a single solution simultaneously optimises each objective and, given the data has observation errors and biases and models can be misspecified, most likely each partial objective function will be conflicting with each other (Maunder & Piner, 2017). As a result, a multi-objective optimisation approach will produce many different equivalent solutions, while we should expect that only one set of parameters has physical, biological or ecological meaning given our model is an appropriate representation of the system under study. By combining the likelihood of each data set, and weighting the datasets by their assumed uncertainty, we are able to find a solution that integrates all the data. Some negative log-likelihood functions are already implemented and proposed for common distributions (e.g. normal, lognormal, multinomial, Poisson; type?objFn to see the available functions). User defined functions can be used, as long as they accept two arguments (obs and sim) and return a scalar value, including ad-hoc distance measures as used in ABC methods. For example, to implement a calibration using the least squares method, we can write the following function to calculate the residual sum of squares:

```
RSS = function(obs, sim, ...) {
 value = sum((obs-sim)^2, na.rm=TRUE)
 return(value)
 }
```

The 'calibrate' column in the configuration of the objective function provides a flag to select the variables to be used for the calibration. The 'use_data' column indicates whether data are read from the disk. If use_data=TRUE, the file specified in the calibration settings will be used. If use_data=FALSE, the observed value is set to NULL, and the type function is expected to use simulated data only. The latter option can be particularly useful to set penalties in the model outputs or parameters, where no observed data are needed. Finally, the 'weight' column provides the relative weights to combine the values obtained for each variable. A more detailed illustration of this process is provided in the vignettes and demos of the package.

4 | RUNNING A CALIBRATION

The calibrate function takes a list as a control argument, where fine control options are provided, for example for the parallelisation of the optimisation (based on the foreach package, Revolution Analytics & Weston, 2014). Before using the parallel implementation, a parallel 'cluster' should be created, which can be easily done using the parallel or snow R packages (see vignettes). This allows full control of the configuration of the parallel runs, making the calibration work in different computer systems, from computers with multicore processors to high-performance supercomputers. Once the cluster is created, the parallel=TRUE argument must be included in the call, and the ncores control option should indicate the number of cores chosen (see Table 4 for details). Additionally, since each model run could require files to be written to the disk (which will be read by the run_model function after the simulation), a different folder needs to be assigned for each parameter combination that is tested by the optimisation algorithm or for the computation of numerical gradients. For this purpose, the run control option allows a directory to be specified where all the simulations are run (subfolders named i0, i1, ..., in-1 will be automatically created as needed). By default, no folders are created, so a path should be specified if the model needs to write files to the disk. All the parameter input files (Figure 2a) will be written in temporary folders (e.g. run/i0). The control option master allows a folder to be specified, the full content of which will be copied to these temporary folders. Since the calibration of numerically intensive models can run for a long time, a 'restart' option is also available, allowing an interrupted calibration to be continued.

5 | THREE APPLICATION EXAMPLES

To illustrate the main functionality of the package, we estimated the parameters for a predator–prey model using the calibrate function. The model was defined by a system of ordinary differential equations for the abundance of prey *N* and predator *P*:

$$\begin{aligned}\frac{dN}{dt} &= rN\left(1-\frac{N}{K}\right)-\alpha NP\\ \frac{dP}{dt} &= -lP+\gamma\alpha NP\end{aligned}.$$

The parameters to estimate were the prey's growth rate *r*, the predator's mortality rate *l*, the carrying capacity of the prey *K* and α and γ for the predation interaction. To start, we created the demonstration data for this model using the function calibrar_demo function (see help in the package for details on this function used to illustrate the functionality of the package) with T=100 as an additional argument to specify the time horizon.

```
LV = calibrar_demo(path=path, model='PredatorPrey', T=100)
setup = calibration_setup(file = LV$setup)
observed = calibration_data(setup=setup, path=LV$path)
run_model = calibrar:::.PredatorPreyModel
obj = calibration_objFn(model=run_model, setup=setup, ob-
served=observed, T=LV$T,
 aggregate=TRUE)
```

To run the calibration, we needed to specify the initial guess for the parameter values (par), the objective function to minimise (fn) and optionally the lower and upper thresholds for the parameters (lower and upper) and the phase number at which each parameter needs to be estimated (phases). See the Supporting Information for the full description of the objective function and calibration setup.

```
calibrate(par=LV$guess, fn=obj, lower=LV$lower, upper=LV$upper,
phases=LV$phase)
```

The argument method can be used to change the default optimisation algorithm. We calibrated the model using five different optimisation algorithms with their default parameters: (i) derivative-based: Algorithm 21 (L-BFGS-B, default for deterministic functions, from package Rvmmin), the conjugate gradient

TABLE 4 Some options for the control argument of the function calibrate.

Option	Description
maxit	Maximum number of iterations of the algorithm
ncores	The number of cores available in the parallel cluster for the active session. If parallel=TRUE, the default is to get the number of cores of the system
run	An optional folder path to create all the temporary files needed to run the simulations from the objective function for each parameter combination tested by the optimisation algorithms
master	An optional folder path. All the contents of the master folder will be copied to the run folder, one copy in one subfolder per parameter combination tested at each iteration
REPORT	Number of iterations after saving a new restart object, which contains all the information necessary to restart the calibration at that point. The default is NULL, and no restart files are created
restart.file	Basename for the restart file to be created

method (CG from the Rcgmin package); (ii) derivative free: Nelder–Mead (default R optimiser, stats::optim()); (iii) heuristic: AHR-ES (default for stochastic functions) and the CMA-ES (from cmaes package). As shown by the optimisation results (Figure 3 and Table 5), the algorithm Nelder–Mead and CG could not find the solution using the default parameters provided in the original optimisation functions (while some improvements may be obtained after some tuning). The full code can be found in the vignettes of the package.

A second example involves the calibration of a Poisson Autoregressive Mixed model for the dynamics of a population in different sites:

$$\log(\mu_{i,t+1}) = \log(\mu_{i,t}) + \alpha + \beta X_{i,t} + \gamma_t,$$

where $\mu_{i,t}$ is the size of the population in site i at year t, $X_{i,t}$ is the value of an environmental variable in site i at year t. The parameters to estimate were α, β, and γ_t, the random effects for each year ($\gamma_t \sim N(0, \sigma^2)$), and the initial population at each site $\mu_{i,0}$. We assumed that the observations $N_{i,t}$ follow a Poisson distribution with mean $\mu_{i,t}$. We could also create the data for this model using the function calibrar_demo, with the additional arguments L=5 (five sites) and T=100 (one hundred years):

```
ARPM = calibrar_demo(path=path, model="PoissonMixedModel",
L=5, T=100)
setup = calibration_setup(file=ARPM$setup)
observed = calibration_data(setup=setup, path=ARPM$path)
forcing = as.matrix(read.csv(file.path(ARPM$path, "master", "environment.csv"),
 row.names=1))
```

For this example, the run_model function also returns the time series of γ_t that will be used to add a penalty when constructing the

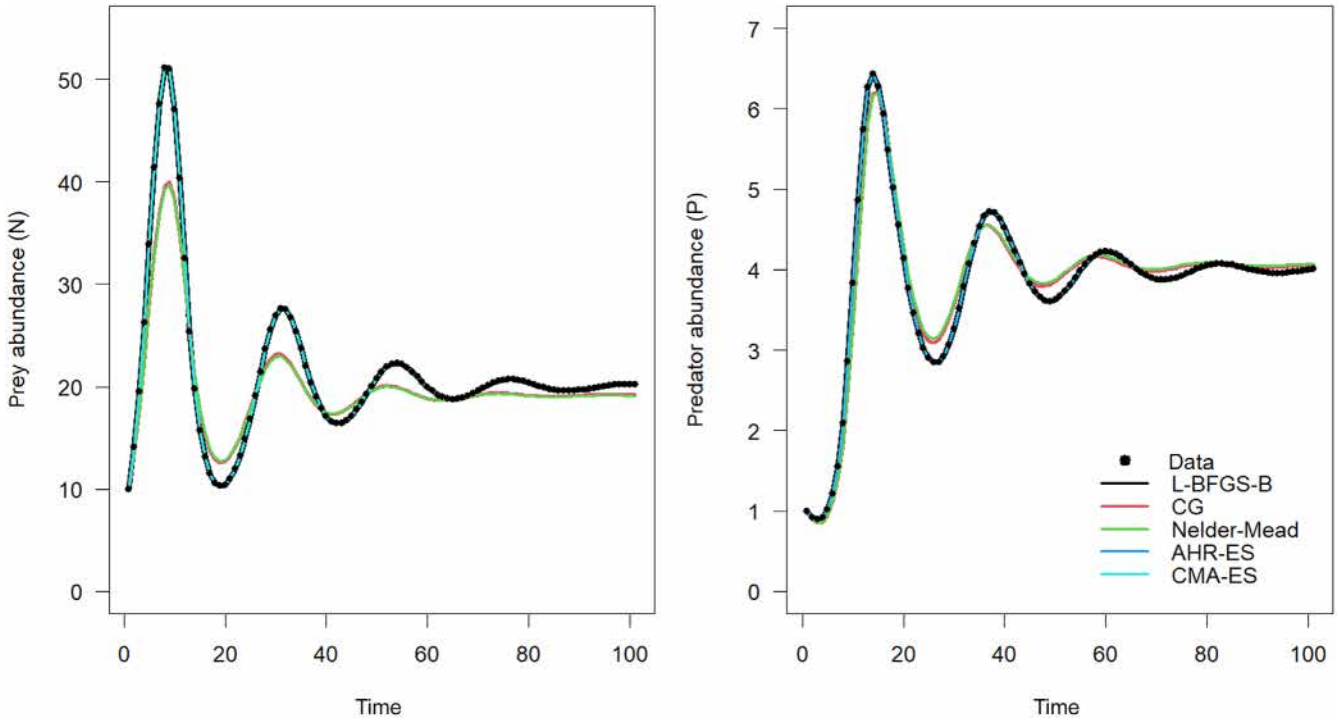

FIGURE 3 Results of the calibration of the predator–prey model using different optimisation methods. The simulated data (points) and model fits (lines) are shown. For the optimisation methods "AHR-ES", "L-BFGS-B" and "CMA-ES" there are no visual differences and the lines merge.

Method	Objective function value	Parameter				
		R	L	K	α	γ
Data	4.96E-07	0.5	0.2	100	0.1	0.1
L-BFGS-B (Rvmmin)	5.61E-07	0.4999	0.2000	99.9960	0.1000	0.1000
CG (Rcgmin)	1.29961642	0.4489	0.2746	67.4768	0.0796	0.1802
Nelder–Mead	1.41516908	0.4500	0.2751	66.2434	0.0788	0.1827
AHR-ES	5.61E-07	0.4999	0.2000	99.9960	0.1000	0.1000
CMA-ES	5.61E-07	0.4999	0.2000	99.9960	0.1000	0.1000

TABLE 5 Summary of the calibration results for the predator–prey model using five different optimisation methods with default parameters.

objective function. This function also includes an additional argument named 'forcing' that is used to pass additional forcing data (environmental conditions at each site and time step in this example) that is needed to run the model:

```
run_model = function(par, forcing) {
 output = calibrar:::.PoissonMixedModel(par=par, forcing=forcing)
 output = c(output, list(gammas=par$gamma))
 return(output)
}

obj = calibration_objFn(model=run_model, setup=setup, observed=observed, forcing=forcing, aggregate=TRUE)
```

The calibration was run as in the previous example calling the calibrate function:

```
calibrate(par=ARPM$guess, fn=obj, lower=ARPM$lower, upper=ARPM$upper, phases=ARPM$phase)
```

Here we calibrated the model using four different optimisation algorithms: (i) derivative-based: L-BFGS-B (stats) and L-BFGS-B v3, (ii) derivative free: HJKB; (iii) heuristic: AHR-ES and SANN (Figure 4 and Table 6). In this case, the SANN algorithm could not find the solution despite an increased number of maximum iterations. The best solution found was using HJKB algorithm, closely followed by the AHR-ES and L-BFGS-B algorithms (Table 6 and Figure 4). The full code can be found in the vignettes of the package.

For a given model, such as the autoregressive Poisson mixed model, the calibration can take up to several hours depending on the optimisation method used (Table 6). Therefore, the possibility to restart an interrupted optimisation can be useful. To do this, the calibrate function must be called with the additional control argument 'restart.file', which indicates the name of the file storing the information needed to restart the optimisation (see Table 4):

```
calibrate(par=ARPM$guess, fn=obj, lower=ARPM$lower,
upper=ARPM$upper, phases=ARPM$phase,
 control=list(restart.file="arpm"))
```

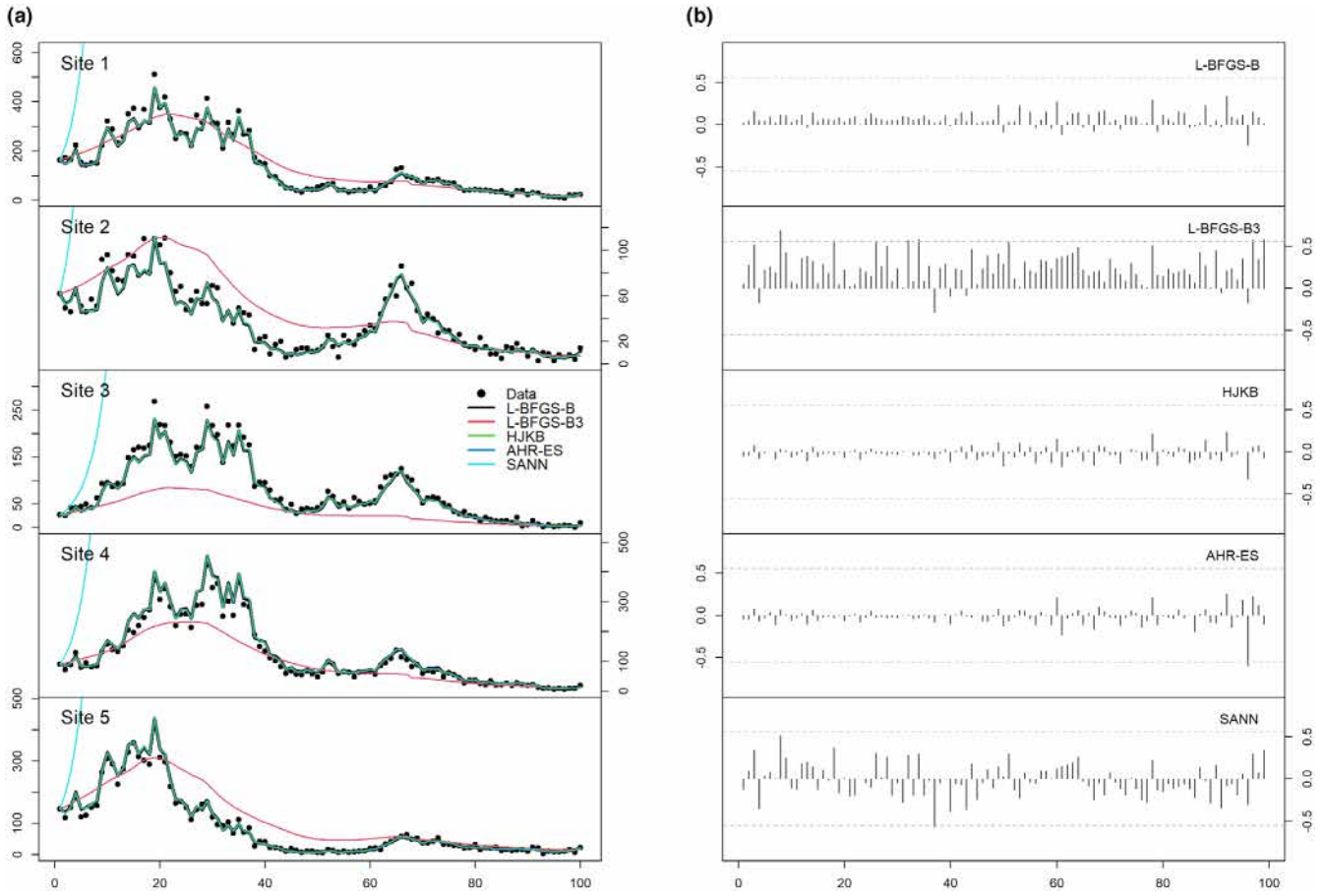

FIGURE 4 Results of the calibration of the autoregressive Poisson mixed model using different optimisation methods. (a) The simulated data (points) and model fits (lines) are shown. For the optimisation methods "HJKB", "AHR-ES" and "L-BFGS-B" there are no visual differences and the lines merge. (b) Time series of differences between the γ_t estimated by each algorithm and the real parameter values. For each case, the dotted lines represent the 95% limits for the differences.

TABLE 6 Summary of the calibration results for the autoregressive Poisson mixed model using five different optimisation methods.

Method	Elapsed time (seconds)	Objective function value	Parameters					
			α	β	γ_t			
					Mean	SD	Correlation	Bias
Data	—	−186178.145	0.4000	−0.4000	−0.0187	0.1952	1.000	0.0000
L-BFGS-B	186	−186081.484	0.4846	−0.4157	−0.0867	0.1923	0.904	0.0680
L-BFGS-B 3.0	14	−180576.287	0.3700	−0.1389	−0.2528	0.0460	0.311	0.2341
HJKB	222	−186094.819	0.3923	−0.4145	0.0039	0.1869	0.908	−0.0226
AHR-ES	612	−186085.887	0.3967	−0.4143	−0.0009	0.1940	0.862	−0.0178
SANN	43,004	1.5117E+16	0.2000	0.1000	0.0000	0.0000	—	−0.0187

If an appropriate restart file is found in the working directory, the calibration will be restarted from the last saving point.

A third and more complex application involved the calibration of the stochastic individual-based model OSMOSE, a multispecies spatially explicit ecosystem model (Shin & Cury, 2001, 2004). It has been applied in the Northern Peru Current Ecosystem (Oliveros-Ramos et al., 2017) to model the life history and spatiotemporal dynamics of nine interacting species, between 1992 and 2008. The model was confronted with time series of abundance indices, fisheries landings and catch-at-length composition data (see Hilborn & Mangel, 1997; a comprehensive introduction to the subject). The objective function used a penalised likelihood approach, combining log-normal and multinomial likelihoods (see the Supporting Information for the full description of the objective function and calibration setup). A total of 307 parameters were estimated in four sequential phases. The OSMOSE model is implemented in Java, and it was not an option to recode it in another language for parameter estimation purposes. Each calibration trial lasted 5 days using a High-Performance Computing (HPC) cluster under the Portable Batch System (PBS) for jobs scheduling, used 64 cores and needed to be relaunched every 24 h due to system restrictions. The objective function for this model is stochastic and was optimised using the AHR-ES algorithm. The calibration in sequential phases improved the final parameter estimates as the calibration in only one phase could not converge to an acceptable solution (see Oliveros-Ramos et al., 2017 for details on how to set the parameters' phases). The calibrar package greatly simplified the task of the calibration of a complex model like OSMOSE, which otherwise can be highly technical, if not impossible, with other optimisation packages. The scripts used for this calibration that may be adapted for applications of the calibrar package in other HPC systems are available in the Supporting Information. For other applications, the calibrar package has been used in the model fitting of some complex models like in Oliveros-Ramos et al. (2010), Oliveros-Ramos and Peña-Tercero (2011), Grüss et al. (2015, 2016), Dueri et al. (2016), Halouani et al. (2016), Travers-Trolet et al. (2019), Moullec et al. (2019), Bănaru et al. (2019), Xing et al. (2020) and Morell et al. (2023). More examples can be found in the vignette of the package 'Parameter estimation for ODE systems', that illustrates how the calibrar package can be used for the calibration of models where ABC methods have been used previously (e.g. see Minter & Retkute, 2019), but with considerably less computational cost.

6 | COMPARISON WITH OTHER SOFTWARE

Implementation of general-purpose optimisers can be found in R (see Optimisation and Mathematical Programming Task View at CRAN: http://cran.r-project.org/web/views/Optimisation.html). Two very useful features for model calibration are the performance of constrained optimisation (limiting the search to a box by defining lower and upper boundaries to parameter values) and the calibration in multiple phases (to improve the search of the global minimum by performing a sequential approximation). The former option is implemented in several R packages, including the optim function (providing the "L-BFGS-B" method, Byrd et al., 1995) and several others wrapped in the optimx package (Nash & Varadhan, 2011). The latter option is available in some other R packages (e.g. Rcgmin and Rvmmin) for a single optimisation, but a sequential calibration, as described here, would have to be performed manually. Additionally, the calibrar package allows the choice of different optimisation methods for each calibration phase, allowing the combination of heuristic global optimisation methods and local optimisation ones to improve the performance of a multiple phase calibration.

For the particular purpose of the calibration of stochastic models, several meta-heuristic and non-derivative based algorithms are now available in R, from EAs (e.g. genalg, DEoptim and cmaes packages) to other nature-inspired algorithms (e.g. Simulated Annealing 'SANN' method in optim and the Particle Swarm Optimisation (PSO) algorithm in the hydroPSO package, Zambrano-Bigiarini & Rojas, 2013). However, while all of these packages and algorithms provide support for constrained optimisation, none of them provides support for keeping fixed parameters during the course of a single optimisation, and a multiple phase calibration would have to be performed manually by modifying the objective function for each trial. Furthermore, from the implementation point of view, a very important feature for the calibration of

complex models is the parallel implementation of the optimisation routine. The PSO algorithm in the hydroPSO has its parallel implementation tied to the core of the function and does not allow its use in high-performance clusters, especially under a queue system, and only the DEoptim package provides a more flexible externally configured parallelisation.

Additionally, in the construction of the objective function, calibrar allows an easy transferability of the calibration problem to other general-purpose optimisers, which can be useful under certain circumstances (e.g. see Bolker et al., 2013). There is indeed "no free lunch" in optimisation, and no optimisation algorithm will perform better than all others for every type of optimisation problems (Wolpert & Macready, 1997) and testing multiple optimisation algorithms is recommended (Bolker et al., 2013). Other calibration-oriented packages like hydroPSO provide functions to write parameters and read outputs, but this approach breaks the 'objective function' approach for the optimisation, by not allowing the transfer of the objective function to other optimisers, and while the hydromad package (Andrews et al., 2011) offers support for the automated construction of an objective function in a standard way, it is restricted to some particular cases useful in hydrological modelling. In these regards, the package calibrar is meant to be generic enough to be used in a variety of optimisation problems, including the calibration of complex (i.e. non-linear, with a lot of parameters and long runtime) and stochastic models. Three features of calibrar render it particularly useful for the calibration of computationally intensive stochastic models: the parallelisation of the simulations, the ability to handle replicate simulations in the evaluation of the objective function and the 'restart' option, which allows the calibration of complex models to be handled under restricted access to high performance resources (e.g. clusters with queue systems and fixed wall time).

7 | CONCLUSIONS AND PERSPECTIVES

A successful model calibration implies several computational, theoretical and practical challenges. The calibrar package is intended to provide a framework to simplify the calibration of models, in particular complex and stochastic ones (e.g. individual-based models), for which there have been fewer developments compared to those for deterministic and differentiable models. To our knowledge, the calibrar package is the first one wrapping several global search optimisation methods, facilitating their use and testing for parameter estimation. We adopted a 'black-box' and 'non-intrusive' approach, since most complex models are often computationally intensive and most likely implemented in fast low-level languages and recoding for calibration purposes is not the best option. The restart functionality can help in the optimisation of computationally intensive problems, since most of the computing time is spent evaluating the objective function (Nash, 2014) and the ability to restart an interrupted optimisation could save significant amounts of time. By using the generic approach of separating the optimiser from the objective function, users can benefit from the optimisation capabilities of the package while being able to write complex objective functions that deal with the uncertainties and biases typically found in ecological datasets. This also facilitates the good practice of testing several optimisers during the parameter estimation process (Bolker et al., 2013) and additional state-of-the-art optimisation methods are planned to be added in future versions. The use and testing of the calibrar package with several real-world optimisation problems for the calibration of complex ecological models has directed the developments of the package and its current flexibility, while we expect future applications will help to continue improving the package given its open-source nature.

AUTHOR CONTRIBUTIONS

Ricardo Oliveros Ramos and Yunne-Jai Shin conceived the ideas and designed methodology; Ricardo Oliveros Ramos collected the data; analysed the data and led the writing of the manuscript. All authors contributed critically to the drafts and gave final approval for publication.

ACKNOWLEDGEMENTS

This research has been funded by the Biodiversa and Belmont Forum project SOMBEE (BiodivScen ERA-Net COFUND programme, ANR contract n° ANR-18-EBI4-0003-01), the European Union's Horizon 2020 Research and Innovation Programme under grant agreements no 869300 (FutureMARES), no 817578 (TRIATLAS), the Horizon Europe Research and Innovation Programme under grant agreement no 101060072 (ActNow), France Filière Pêches, the Pew fellows program in Marine Conservation at the Pew Charitable Trusts, and the Fondation Pour la Recherche sur la Biodiversité (EMIBIOS, FRB contract no. APP-SCEN-2010-II). R.O.-R. was supported by an individual doctoral research grant (BSTD) from the "Support and training of scientific communities of the South" Department of IRD, managed by Campus France. We thank David Kaplan, Coleen Moloney, and Arnaud Bertrand for their advice during the development of this research. We acknowledge Philippe Verley, Nicolas Barrier, Morgane Travers-Trolet, Laure Velez, Criscely Luján and Alaia Morell for testing and helping to improve the calibrar package. We also thank the Pôle de Calcul et de Données Marines (PCDM) for providing access to their DATARMOR computing resources (https://wwz.ifremer.fr/pcdm/Equipement). This work is a contribution from the Cooperation agreement between the Instituto del Mar del Peru (IMARPE), the University of Cape Town (UCT) and the Institut de Recherche pour le Développement (IRD), through the LMIs DISCOH and ICEMASA.

CONFLICT OF INTEREST STATEMENT

The authors declare no conflict of interest.

PEER REVIEW

The peer review history for this article is available at https://www.webofscience.com/api/gateway/wos/peer-review/10.1111/2041-210X.14452.

DATA AVAILABILITY STATEMENT

All the data used in this article can be obtained with the code already included in the manuscript, supplementary material or the vignettes of the package. The code associated to the package is available via https://doi.org/10.32614/CRAN.package.calibrar (Oliveros-Ramos, 2024), and via https://github.com/roliveros-ramos/calibrar.

ORCID

Ricardo Oliveros-Ramos https://orcid.org/0000-0002-8069-2101

REFERENCES

Andrews, F. T., Croke, B. F. W., & Jakeman, A. J. (2011). An open software environment for hydrological model assessment and development. *Environmental Modelling & Software*, *26*, 1171–1185.

Bănaru, D., Diaz, F., Verley, P., Campbell, R., Navarro, J., Yohia, C., Oliveros-Ramos, R., Mellon-Duval, C., & Shin, Y.-J. (2019). Implementation of an end-to-end model of the Gulf of lions ecosystem (NW Mediterranean Sea). I. Parameterization, calibration and evaluation. *Ecological Modelling*, *401*, 1–19. https://doi.org/10.1016/j.ecolmodel.2019.03.005

Bartell, S. M. (2003). Effective use of ecological modeling in management: The toolkit concept. In V. Dale (Ed.), *Ecological modeling for resource management* (pp. 211–220). Springer Verlag.

Bates, D., Mullen, K. M., Nash, J. C., & Varadhan, R. (2023). minqa: Derivative-free optimization algorithms by quadratic approximation. R package version 1.2.6. https://CRAN.R-project.org/package=minqa

Black, A. J., & McKane, A. J. (2012). Stochastic formulation of ecological models and their applications. *Trends in Ecology & Evolution*, *27*(6), 337–345. https://doi.org/10.1016/j.tree.2012.01.014

Bolker, B. M., Gardner, B., Maunder, M., Berg, C. W., Brooks, M., Comita, L., Crone, E., Cubaynes, S., Davies, T., de Valpine, P., Ford, J., Gimenez, O., Kéry, M., Kim, E. J., Lennert-Cody, C., Magnusson, A., Martell, S., Nash, J., Nielsen, A., … Zipkin, E. (2013). Strategies for fitting nonlinear ecological models in R, AD model builder, and BUGS. *Methods in Ecology and Evolution*, *4*, 501–512.

Byrd, R. H., Lu, P., Nocedal, J., & Zhu, C. (1995). A limited memory algorithm for bound constrained optimisation. *SIAM Journal on Scientific Computing*, *16*, 1190–1208.

Ciupke, K. (2016). psoptim: Particle Swarm Optimization. R package version 1.0. https://CRAN.R-project.org/package=psoptim

Clayden, J. (2022). soma: General-purpose optimisation with the self-organising migrating algorithm. R package version 1.2.0. https://CRAN.R-project.org/package=soma

Cropper, W. P., Jr., & Anderson, P. J. (2004). Population dynamics of a tropical palm: Use of a genetic algorithm for inverse parameter estimation. *Ecological Modelling*, *177*, 119–127.

Csilléry, K., François, O., & Blum, M. G. B. (2012). abc: An R package for approximate Bayesian computation (ABC). *Methods in Ecology and Evolution*, *3*(3), 475–479. https://doi.org/10.1111/j.2041-210X.2011.00179.x

Duboz, R., Versmisse, D., Travers, M., Ramat, E., & Shin, Y.-J. (2010). Application of an evolutionary algorithm to the inverse parameter estimation of an individual-based model. *Ecological Modelling*, *221*(5), 840–849.

Dueri, S., Guillotreau, P., Jiménez-Toribio, R., Oliveros-Ramos, R., Bopp, L., & Maury, O. (2016). Food security or economic profitability? Projecting the effects of climate and socioeconomic changes on global skipjack tuna fisheries under three management strategies. *Global Environmental Change*, *41*, 1–12.

Fidler, M. L., Nash, J. C., Zhu, C., Byrd, R., Nocedal, J., & Morales, J. L. (2020). lbfgsb3c: Limited memory BFGS minimizer with bounds on parameters with optim() 'C' Interface. R package version 2020-3.2. https://CRAN.R-project.org/package=lbfgsb3c

Fournier, D. A., Skaug, H. J., Ancheta, J., Ianelli, J., Magnusson, A., Maunder, M. N., Nielsen, A., & Sibert, J. (2012). AD model builder: Using automatic differentiation for statistical inference of highly parameterized complex nonlinear models. *Optimisation Methods and Software*, *27*(2), 233–249. https://doi.org/10.1080/10556788.2011.597854

Griewank, A., & Corliss, G. F. (1992). *Automatic differentiation of algorithms: Theory, implementation, and application*. SIAM.

Grüss, A., Schirripa, M. J., Chagaris, D., Drexler, M., Simons, J., Verley, P., Shin, Y.-J., Karnauskas, M., Oliveros-Ramos, R., & Ainsworth, C. H. (2015). Evaluation of the trophic structure of the West Florida shelf in the 2000s using the ecosystem model OSMOSE. *Journal of Marine Systems*, *144*, 30–47.

Grüss, A., Schirripa, M. J., Chagaris, D., Velez, L., Shin, Y.-J., Verley, P., Oliveros-Ramos, R., & Ainsworth, C. H. (2016). Estimating natural mortality rates and fishing scenarios for Gulf of Mexico red grouper (*Epinephelus morio*) using the ecosystem model OSMOSE-WFS. *Journal of Marine Systems*, *154*, 264–279.

Halouani, G., Lasram, F. B. R., Shin, Y.-J., Velez, L., Verley, P., Hattab, T., Oliveros-Ramos, R., Diaz, F., Ménard, F., Baklouti, M., Guyennon, A., Romdhane, M. S., & Le Loc'h, F. (2016). Modelling food web structure using an end-to-end approach in the coastal ecosystem of the Gulf of Gabes (Tunisia). *Ecological Modelling*, *339*, 45–57. https://doi.org/10.1016/j.ecolmodel.2016.08.008

Hansen, N., & Ostermeier, A. (2001). Completely de-randomized self-adaptation in evolution strategies. *Evolutionary Computation*, *9*(2), 159–195.

Hilborn, R., & Mangel, M. (1997). *The ecological detective: Confronting models with data (MPB-28)*. Princeton University Press.

Ionides, E. L., Breto, C., & King, A. A. (2006). Inference for nonlinear dynamical systems. *PNAS*, *103*(49), 18438–18443.

Jones, D. R., Schonlau, M., & Welch, W. J. (1998). Efficient global optimisation of expensive Black-box functions. *Journal of Global Optimisation*, *13*, 455–492.

Jorgensen, S. E., & Bendoricchio, G. (2001). *Fundamentals of ecological modelling* (3rd ed., p. 530). Elsevier.

Maunder, M. N., & Piner, K. R. (2017). Dealing with data conflicts in statistical inference of population assessment models that integrate information from multiple diverse data sets. *Fisheries Research*, *192*, 16–27. https://doi.org/10.1016/j.fishres.2016.04.022

Mebane, W. R., Jr., & Sekhon, J. S. (2011). Genetic optimization using derivatives: The rgenoud package for R. *Journal of Statistical Software*, *42*(11), 1–26. https://www.jstatsoft.org/v42/i11/

Minter, A., & Retkute, R. (2019). Approximate Bayesian computation for infectious disease modelling. *Epidemics*, *29*, 100368. https://doi.org/10.1016/j.epidem.2019.100368

Morales, J. L., & Nocedal, J. (2011). Remark on "Algorithm 778: L-BFGS-B: Fortran subroutines for large-scale bound constrained optimization". *ACM Transactions on Mathematical Software (TOMS)*, *38-1*, 1–4. https://doi.org/10.1145/2049662.2049669

Morell, A., Shin, Y.-J., Barrier, N., Travers-Trolet, M., Halouani, G., & Ernande, B. (2023). Bioen-OSMOSE: A bioenergetic marine ecosystem model with physiological response to temperature and oxygen. BioRxiv, 2023.01.13.523601 https://doi.org/10.1101/2023.01.13.523601

Moullec, F., Velez, L., Verley, P., Barrier, N., Ulses, C., Carbonara, P., Esteban, A., Follesa, C., Gristina, M., Jadaud, A., Ligas, A., Díaz, E. L., Maiorano, P., Peristeraki, P., Spedicato, M. T., Thasitis, I., Valls, M., Guilhaumon, F., & Shin, Y.-J. (2019). Capturing the big picture of Mediterranean marine biodiversity with an end-to-end model of climate and fishing impacts. *Progress in Oceanography*, *178*, 102179. https://doi.org/10.1016/j.pocean.2019.102179

Mullen, K., Ardia, D., Gil, D., Windover, D., & Cline, J. (2011). 'DEoptim': An R package for global optimization by differential evolution.

Journal of Statistical Software, *40*(6), 1–26. https://doi.org/10.18637/jss.v040.i06
Nash, J. C. (2014). *Nonlinear parameter optimization using R tools* (p. 287). Wiley and Sons.
Nash, J. C. (2021). Rvmmin: Variable metric nonlinear function minimization. R package version 2018-4.17.1. https://CRAN.R-project.org/package=Rvmmin
Nash, J. C. (2022). Rcgmin: Conjugate gradient minimization of nonlinear functions with box constraints. R package version 2022-4.30. https://CRAN.R-project.org/package=Rcgmin
Nash, J. C., & Varadhan, R. (2011). Unifying optimisation algorithms to aid software system users: Optimx for R. *Journal of Statistical Software*, *43*(9), 1–14. http://www.jstatsoft.org/v43/i09/
Nash, J. C., & Walker-Smith, M. (1987). *Nonlinear parameter estimation: An integrated system in BASIC* (p. 493). Marcel Dekker.
Newman, K. B., Fernández, C., Thomas, L., & Buckland, S. T. (2009). Monte Carlo inference for state–space models of wild animal populations. *Biometrics*, *65*, 572–583.
Oliveros-Ramos, R. (2024). calibrar: Automated parameter estimation for complex models. R package version 1.0. https://doi.org/10.32614/CRAN.package.calibrar
Oliveros-Ramos, R., Guevara-Carrasco, R., Simmonds, J., Cirske, J., Gerlotto, F., Peña-Tercero, C., Castillo, R., & Tam, J. (2010). Modelo de evaluación integrada del stock norte-centro de la anchoveta peruana Engraulis ringens. *Boletin Instituto del Mar del Perú*, *25*(1–2), 49–55.
Oliveros-Ramos, R., & Peña-Tercero, C. (2011). Modeling and analysis of the recruitment of Peruvian anchovy (*Engraulis ringens*) between 1961 and 2009. *Ciencias Marinas*, *37*(4B), 659–674.
Oliveros-Ramos, R., Verley, P., & Shin, Y.-J. (2017). A sequential approach to calibrate ecosystem models with multiple time series data. *Progress in Oceanography*, *151*, 227–244. https://doi.org/10.1016/j.pocean.2017.01.002
Poovathingal, S. K., & Gunawan, R. (2010). Global parameter estimation methods for stochastic biochemical systems. *BMC Bioinformatics*, *11*, 414.
R Core Team. (2023). *R: A language and environment for statistical computing*. R Foundation for Statistical Computing. http://www.R-project.org/
Rajwar, K., Deep, K., & Das, S. (2023). An exhaustive review of the metaheuristic algorithms for search and optimization: Taxonomy, applications, and open challenges. *Artificial Intelligence Review*, *56*(11), 13187–13257. https://doi.org/10.1007/s10462-023-10470-y
Revolution Analytics, & Weston, S. (2014). foreach: Foreach looping construct for R. R package version 1.4.2. http://CRAN.R-project.org/package=foreach
Rios, L. M., & Sahinidis, N. V. (2013). Derivative-free optimisation: A review of algorithms and comparison of software implementations. *Journal of Global Optimisation*, *56*, 1247–1293.
Ross, J. V., Pagendam, D. E., & Pollet, P. K. (2009). On parameter estimation in population models II: Multi-dimensional processes and transient dynamics. *Theoretical Population Biology*, *75*, 123132.
Shin, Y.-J., & Cury, P. (2001). Exploring fish community dynamics through size-dependent trophic interactions using a spatialized individual-based model. *Aquatic Living Resources*, *14*(2), 65–80.
Shin, Y.-J., & Cury, P. (2004). Using an individual-based model of fish assemblages to study the response of size spectra to changes in fishing. *Canadian Journal of Fisheries and Aquatic Sciences*, *61*, 414–431.
Sunnåker, M., Busetto, A. G., Numminen, E., Corander, J., Foll, M., & Dessimoz, C. (2013). Approximate Bayesian computation. *PLoS Computational Biology*, *9*(1), e1002803. https://doi.org/10.1371/journal.pcbi.1002803
Tashkova, K., Silc, J., Atanasova, N., & Dzeroski, S. (2012). Parameter estimation in a nonlinear dynamic model of an aquatic ecosystem with meta-heuristic optimisation. *Ecological Modelling*, *226*, 36–61.
Thiele, J. C., Kurth, W., & Grimm, V. (2012). RNetLogo: An R package for running and exploring individual-based models implemented in NetLogo. *Methods in Ecology and Evolution*, *3*, 480–483.
Trautmann, H., Mersmann, O., & Arnu, D. (2011). cmaes: Covariance Matrix Adapting Evolutionary Strategy. R package version 1.0–11. https://CRAN.R-project.org/package=cmaes
Travers-Trolet, M., Coppin, F., Cresson, P., Cugier, P., Oliveros-Ramos, R., & Verley, P. (2019). Emergence of negative trophic level-size relationships from a size-based, individual-based multispecies fish model. *Ecological Modelling*, *410*, 108800. https://doi.org/10.1016/j.ecolmodel.2019.108800
Travers-Trolet, M., Shin, Y.-J., & Field, J. G. (2013). An end-to-end coupled model ROMS-N2P2Z2D2-OSMOSE of the southern Benguela food web: Parameterisation, calibration and pattern-oriented validation. *African Journal of Marine Science*, *36*(1), 11–29. https://doi.org/10.2989/1814232X.2014.883326
Varadhan, R., & Gilbert, P. (2009). BB: An R package for solving a large system of nonlinear equations and for optimizing a high-dimensional nonlinear objective function. *Journal of Statistical Software*, *32*(4), 1–26. https://doi.org/10.18637/jss.v032.i04
Varadhan, R., Johns Hopkins University, Borchers, H. W., Research, A. C., Bechard, V., & Montreal, H. (2023). dfoptim: Derivative-free optimization. R package version 2023.1.0. https://CRAN.R-project.org/package=dfoptim
Walker, D. M., Pérez-Barbería, F. J., & Marion, G. (2006). Stochastic modelling of ecological processes using hybrid Gibbs samplers. *Ecological Modelling*, *198*, 40–52.
Walter, E., & Pronzato, L. (1997). *Identification of parametric models from experimental data* (p. 413). Springer Masson.
Wolpert, D. H., & Macready, W. G. (1997). No free lunch theorems for optimisation. *IEEE Transactions on Evolutionary Computation*, *1*(1), 67–82.
Wong, W., & Ming, C. I. (2019). A review on metaheuristic algorithms: Recent trends, benchmarking and applications. In *2019 7th International Conference on Smart Computing & Communications (ICSCC)* (pp. 1–5). IEEE. https://doi.org/10.1109/ICSCC.2019.8843624
Xiang, Y., Gubian, S., Suomela, B., & Hoeng, J. (2013). Generalized simulated annealing for efficient global optimization: The GenSA package for R. *The R Journal*, *5/1*, 13–28. https://journal.r-project.org
Xing, L., Chen, Y., Zhang, C., Li, B., Tanaka, K. R., Boenish, R., & Ren, Y. (2020). Evaluating impacts of imprecise parameters on the performance of an ecosystem model OSMOSE-JZB. *Ecological Modelling*, *419*, 108923. https://doi.org/10.1016/j.ecolmodel.2019.108923
Zambrano-Bigiarini, M., & Rojas, R. (2013). A model-independent particle swarm optimisation software for model calibration. *Environmental Modelling & Software*, *43*, 5–25.

SUPPORTING INFORMATION

Additional supporting information can be found online in the Supporting Information section at the end of this article.

Supporting Information S1. Code and data required to run the examples.

How to cite this article: Oliveros-Ramos, R., & Shin, Y.-J. (2025). calibrar: An R package for fitting complex ecological models. *Methods in Ecology and Evolution*, *16*, 507–519. https://doi.org/10.1111/2041-210X.14452