***Bristlecone*: An F# library for model-fitting and model-selection of ecological time-series models**

Andrew C Martin, Long-Term Ecology Lab, University of Oxford

Abstract:

Environmental archives such as sediment cores and tree rings provide important insights on the timing and rates of change in biodiversity and ecosystem function over the long-term. Such datasets are often analysed using empirical methods, which limits their ability to address ecological questions that seek to understand underlying ecological mechanisms and processes. Top down modelling approaches – where data is confronted with simple ecological models – can be used to infer the presence, form, and strength of mechanisms of interest. To aid adoption of time-series mechanistic modelling for long-term ecology, we created a F# library, *Bristlecone*, that can be used to apply this approach using a Model-Fitting and Model-Selection workflow. Our objective with *Bristlecone* was to create a library that could be used to efficiently and effectively conduct a full MFMS analysis for long-term ecological problems. We incorporated techniques to address specific challenges with environmental archives, including uneven time steps from age-depth models (for sediment cores), and allometry and seasonality (for tree rings). We include an example analysis to demonstrate functionality of *Bristlecone*. Our solution presents a straightforward, repeatable, and highly parallel method for conducting inference for long-term ecological problems.

Keywords: ecological models; dendroecology; dendrochronology; long-term ecology; palaeoecology; mechanistic modelling; model-fitting and model-selection.

1.1 Introduction

Time-series observations of ecological phenomena and the environment provide essential insights into rates of change, the order of processes, and the resilience or otherwise of ecological systems (Willis et al. 2010). Common sources of long-term environmental time-series are sediment cores, tree rings, and historical records. Analyses that utilise time-series overcome limitations in space-for-time and chronosequence approaches, where the order and relative magnitudes of processes cannot necessarily be determined (Johnson & Miyanishi 2008).

Time-series data of long-term environmental processes are often interrogated using empirical methods. For example, tree rings provide direct and annually resolved observations of plant growth through time. Standard dendrochronological techniques involve correlation to assess the sensitivity of ring-width series to environmental variables such as air temperature and precipitation (Fritts 1976). Similar correlative approaches may also be applied to time-series of direct and indirect proxies measured in sedimentary sequences; for example, pollen counts may be correlated with paleoclimate reconstructions to determine which plant taxa occur with which climate conditions (Binney et al. 2017). Alternatively, palaeoenvironment may be reconstructed by applying modern analogues to sedimentary time-series such as charcoal or geochemistry; statistical analysis is then performed to identify the relationship between biodiversity and the environment (Blaauw et al. 2019; Birks et al. 2014).

There is an increasing importance for rich sedimentary and tree-ring datasets to be able to contribute towards understanding the underlying ecological processes that give rise to observed patterns and trends. Top-down modelling approaches – where simple theoretical models are applied to complex systems (Poorter et al. 2013) – can provide insights into the mechanisms operating within a study system, and their rates. This approach is unlike forward- or projection-modelling where current knowledge is projected forward in time; rather here we aim to infer the structure and parameters of processes that drive previous observations (Peng et al. 2011). Model-fitting and model-selection (MFMS) strategies may be implemented to identify the most appropriate model structure and model parameters to best explain observed time-series data (Bonsall & Hastings 2004; Archontoulis & Miguez 2015). MFMS approaches have been previously applied to proxy information from sediment cores to assess the underlying causes of the rise and decline of key plant taxa (Jeffers et al. 2011); and to wood rings to identify the strength of limitations on plant growth (Martin et al. n.d.).

Here we describe a library called ***Bristlecone*** that enables the full model-fitting and model-selection workflow, and has been previously used to infer the nature and strength of plant growth processes in dendroecology (Martin et al. n.d.). It is implemented in F#, a strongly typed functional programming language, which is being increasingly used within data-science and machine learning domains. Our objective with *Bristlecone* was to create a library that could be used to efficiently and effectively conduct

a full MFMS analysis for long-term ecological problems. To achieve this, we implemented a software library to enable three fundamental concerns:

1. Definition of models, data, and MFMS method using a clear domain-specific language;
2. Minimisation of error in model definition and data before any MFMS analysis is started; and
3. Application of a pre-built workflow for problems that use wood ring time-series.

The full source code for *Bristlecone* is available at <https://github.com/AndrewIOM/bristlecone>. The paper is structured as follows: in Section 1.2 we outline the MFMS method, in 1.3 we discuss the *Bristlecone* framework; in Section 1.4 we provide an example use-case; and in Section 1.5 we discuss extensibility.

1.2 Method Overview

In *Bristlecone* we provide tools to infer the structure and magnitude of processes from long-term ecological data using an information-theoretic approach of model-fitting and models-selection (Burnham & Anderson 2007). The confrontation of data with models was demonstrated for broad ecological questions in ‘The Ecological Detective’ (Hilborn & Mangel 1997). There are four key stages to the model-fitting and model-selection process. First, the hypotheses under interrogation must be represented as mathematical models. An appropriate measure for goodness-of-fit must also be identified at this stage. Second, verification is required that it is possible to infer parameter values successfully for any given mathematical model and method of inference. If this is not the case, there may be problems with the model or the appropriateness of the selected inference method. Third, the real observations must be organised, and the parameter set optimised for each model hypothesis to gain a maximum likelihood estimate. Finally, the most appropriate model must be determined using model-selection criteria.

There are specific challenges when applying this approach to long-term ecological data. Ecological time-series – such as data from sedimentary archives – are often short and noisy posing challenges for model inference. In addition, theoretical models in ecology can be highly non-linear (Bolker 2007), requiring the application of robust model-fitting algorithms and diagnostics. We include suitable model-fitting heuristics in *Bristlecone*. In addition, theoretical models are often described as time-dependent ordinary differential equations (ODEs): these require integration in continuous time, which adds complexity to model-fitting. *Bristlecone* handles both continuous-time and discrete-time model formulations.

Bristlecone includes methods adapted specifically for long-term ecological data: additional considerations exist for tree ring and sedimentary data. Sediment cores rely on an age-depth model where the resultant time-series does not have common timesteps; we therefore included the ability to run models in uneven time intervals. Models of tree ring growth rely on allometric relations that dictate

the relationship between different size variables of the plant (e.g. height and stem width). We have included a customisable workflow for analysing ring increment time-series with ecological models.

1.3 *Bristlecone* Framework

Bristlecone is a .NET Standard 2.0 library, which can be used in cross-platform F# scripting or embedded within cross-platform web or desktop applications. We chose to implement our solution in the F# language. F# is a functional, strongly typed language that is being increasingly used for data-science, machine learning and web applications. We chose F# to make use of particular language features. First, *type providers* enable data-driven programming by inferring the structure of datasets as F# types (Petricek et al. 2016). Second, the expressive F# type system allows meta-programming of custom domain-specific languages. Third, F# can be scripted and is cross-platform. In addition, the *RProvider* type provider enables interoperability with R, for uses such as the creation of *ggplot* graphics.

Figure 1 Types involved in the model-fitting and model-selection process in *Bristlecone*. Input types (yellow) represent the observation data, model hypotheses, and methodologies to be used. *Bristlecone* functions (orange) yield results (green) that can be exported to files on disk (blue) using the *Bristlecone.Data* namespace.

Bristlecone was primarily designed for time-series models, which can be defined in continuous or discrete time. *Bristlecone* contains functions for data access, time series manipulation, optimisation, integration of ordinary differential equations (ODEs), and diagnostic measures. We represented the flow of a MFMS analysis via core types (**Figure 1**). A standard MFMS procedure shares a common

abstraction: (1) a series of [nested or non-nested] mechanistic models are developed to represent the hypotheses under study; (2) likelihood functions are chosen that represent the error structure of the time-series being analysed; (3) observational time-series data are imported and arranged to a format suitable for model-fitting step; (4) model-fitting is conducted for every hypothesis independently; and (5) the maximum likelihood estimates for every model are compared using a model-selection criterion. If there are multiple subjects in a study (e.g. individual trees), then steps 1 – 5 are repeated for each. *Bristlecone* enables definition and execution of the above pipeline using a domain specific language (DSL). The full definition of steps 1 – 5 is described in the rest of this section.

1.3.1 Inputs

There are four required inputs for a full MFMS analysis: **Bristlecone.fit** requires hypotheses to test, input time-series, and a model-fitting protocol; and **Bristlecone.select** requires a model-selection protocol.

Each hypothesis is encapsulated by a **ModelSystem** type ('model system'). A model system comprises a list of equations to be computed (either ordinary differential equations or discrete-time equations), model parameters, and a likelihood function. When the model hypotheses are nested, partial application can be used to define a base model and pluggable model components. Within the model system, the components are as follows:

- **Equation<'State> = Time -> 'State -> Environment -> 'State**

An equation in *Bristlecone* has access to the current **Environment**, **ParameterPool**, and time t . The environment contains the current value of all model variables, as well as any environmental forcing variables that have been included.

- **ParameterPool = CodedMap<Parameter>**

The parameter pool contains all of the parameters to be estimated for the model system.

Parameters of a model system are created as a map, where each **Parameter** requires a short code (for reference), bounds, and an optional constraint. The bounds for a parameter may be used differently between different estimation engines. Constraints are required when certain parameter values are not biologically meaningful.

- **LikelihoodFn**

A likelihood function represents the goodness-of-fit of a model hypothesis to data. A function of **LikelihoodFn** type has access to the observed and estimated dataset. *Bristlecone* will minimise this function; therefore, a negative log likelihood or sum-of-squares measure are appropriate.

- **Measure<'State> = 'State -> Environment -> Environment -> 'State**

Measures are optional outputs of the model equations that transform the computed time-series into an additional form. Measures are computed before goodness-of-fit.

Input data sources are represented by the **TimeSeries** type. *Bristlecone*'s time-series representation ensures that observations are always correctly ordered. It also ensures that data adheres to any temporal resolution constraints in modelling approach, such as requiring a fixed time increment. The module contains functions for manipulating the temporal extent and resolution of time-series where, for example, there is a mismatch in the resolution of input datasets.

We represent protocols for MFMS through an **EstimationEngine** type, which can be re-used across many analyses. The engine allows configuration of outputs, time mode, optimisation method, and start value conditioning. Model selection is configured in a similar approach with a **SelectionEngine**.

1.3.2 *Orchestration and Workflow*

For a single research question there can often be many competing model hypotheses (and in some cases more than one subject). We included orchestration tools to run many analyses at once and in parallel. An **OrchestrationAgent** is available in the **Bristlecone.Workflow** namespace. The agent is queue-based: it enqueues **WorkParcel** items and only runs a set number simultaneously. The agent manages shared concerns such as real-time graphing and logging across threads and analyses. We provide simple work parcel functions that iterate hypotheses and subjects, saving results once complete to disk. In addition, advanced scenarios are supported such as the co-ordination of co-dependent workloads: for example, it allows multiple Monte Carlo chains to end when converged according to a convergence statistic.

1.3.3 *Specialist Methods for Long-Term Ecology*

We developed *Bristlecone* to address specific needs for tree ring and sediment data. To address the requirement for uneven timesteps derived from sediment age-depth models, *Bristlecone*'s estimation functions are aware of temporal-resolution and will solve continuous-time problems accordingly. If optional environmental forcing data is provided on a different timeline, interpolation or another specified method can be used to align time-series.

We included a number of adaptations suitable for tree ring research. The *Bristlecone.Dendro* library contains pre-defined workflows for ring width data. In particular, we have included a model template that fits to ring width data by modelling whole-plant biomass dynamics; translating between biomass and stem width using customisable allometric relations; setting the starting biomass using the allometric relation between stem radius and biomass; and outputting wood ring increments.

Seasonality of environmental conditions is an important consideration within tree ring research. At an annual resolution, time-series of environmental conditions must be sliced and generalised to a lower resolution for use as an input variable, for example by constructing a mean June – August air temperature. This approach is appropriate when models are representative of annual processes but may not fully capture the influence of seasonality through time. We include a method for modelling annual

wood production but taking account of higher-resolution variability in the environment, which uses a greater proportion of the underlying data to infer model parameters. Internal high-resolution time-stepping is applied whereas external low-resolution wood increment and other variable data is provided to the likelihood function. *Bristlecone* also includes functions to compute day length given latitude, longitude, and time zone (Meeus 1998), which can be directly accessed within tree ring models to capture the role of seasonal light availability.

1.4 Example

Here we outline a standard *Bristlecone* procedure for a classic ecological problem: predator-prey dynamics. The example code can be placed in an F# script file or run directly using F# interactive. Scripting with *Bristlecone* requires .NET Core 3.0 or greater. Install the *Bristlecone* package into the working directory:

```
$ dotnet tool install -g paket
$ paket init
$ paket add bristlecone
```

After installing .NET Core F# interactive can be run with the **dotnet fsi** command in your working directory. You can either run the below code directly or add it to a script file such as *analysis.fsx* and pass that file to the previous command. First, import and load *Bristlecone*:

```
#load "packages/Bristlecone/bristlecone.fsx"
open Bristlecone
open Bristlecone.ModelSystem
```

Next define a hypothesis (**ModelSystem**). Here we simply define predator-prey dynamics:

```

let ``predator-prey`` =

  /// Number of snowshoe hares
  let dhdt' hare lynx alpha beta =
    alpha * hare - beta * hare * lynx

  /// Number of lynx
  let dldt' lynx hare delta gamma =
    - gamma * lynx + delta * hare * lynx

  let dhdt p t hares env =
    dhdt' hares (lookup env "lynx") (lookup p "alpha") (lookup p "beta")

  let dldt p t lynx env =
    dldt' lynx (lookup env "hare") (lookup p "delta") (lookup p "gamma")

  { Equations = [ code "hare", dhdt
                  code "lynx", dldt ] |> Map.ofList
    Measures = [ ] |> Map.ofList
    Parameters = [ code "alpha", parameter PositiveOnly 0.10 0.60
                   code "beta",  parameter PositiveOnly 0.001 0.0135
                   code "delta", parameter PositiveOnly 0.001 0.0135
                   code "gamma", parameter PositiveOnly 0.10 0.60
                   code "sigmax", parameter Unconstrained -0.2 0.2
                   code "sigmay", parameter Unconstrained -0.2 0.2
                   code "rho",   parameter Unconstrained -0.2 0.2
                 ] |> Map.ofList
    Likelihood = ModelLibrary.Likelihood.sumOfSquares ["hare"; "lynx"] }

```

Here predator-prey dynamics are defined as a system of two ODEs. The functions `dhdt` and `dldt` connect the respective pure mathematical functions into *Bristlecone* by specifying which parameter or variable to use for each input to the ODEs. The next step is to define an **EstimationEngine** with which to fit our model.

```

let engine =
  Bristlecone.mkContinuous
  |> Bristlecone.withGradientDescent
  |> Bristlecone.withConditioning RepeatFirstDataPoint

```

Engines are specified through chained builder functions. Here we have started with *Bristlecone*'s basic continuous-time engine and further specified that gradient-descent (an Amoeba algorithm) should be used for optimisation, and the variables at time-zero should be repeats of the first observation. To assess the performance of this engine with our model, we use *Bristlecone*'s test function:


```
open Bristlecone.Test

let endCondition = Optimisation.EndConditions.afterIteration 100000
let testSettings = {
    TimeSeriesLength = 50
    StartValues = [ code "lynx", 30.09
                    code "hare", 19.58 ] |> Map.ofList
    EndCondition = endCondition
    GenerationRules = []
    NoiseGeneration = fun p ts -> ts
    EnvironmentalData = [] |> Map.ofList
    Resolution = 1 |> Years
    Random = System.Random()
    StartDate = System.DateTime.Create 01 01 2012
    Attempts = 10000 }

let testResult = ``predator-prey`` |> Bristlecone.testModel engine testSettings
```

As we used sum-of-squares to assess goodness of fit here, `testResult` should have a likelihood of zero and minimal difference in the parameter estimates. The test function is flexible allowing noise to be generated on individual expected time-series of variable length. After a successful test, a real model-fitting step can be run as follows:

```
// Here, we use the FSharp.Data CSV type provider to read in a CSV file.
type PopulationData = FSharp.Data.CsvProvider<"data/lynx-hare.csv">
let data =
    let csv = PopulationData.Load "data/lynx-hare.csv"
    [ code "hare", TimeSeries.fromObservations
      (csv.Rows |> Seq.map(fun r -> float r.Hare, r.Year))
      code "lynx", TimeSeries.fromObservations
      (csv.Rows |> Seq.map(fun r -> float r.Lynx, r.Year))
    ] |> Map.ofList

let result = ``predator-prey`` |> Bristlecone.fit engine endCondition data
```

The resultant **EstimationResult** contains the maximum likelihood estimate (MLE), estimated parameters, and a full trace of the optimisation process. The results can be saved to a file using the **Data.EstimationResult.saveAll** function.

When multiple hypotheses are present, model-selection methods can identify the most appropriate model structure for the observed data. In *Bristlecone* we have included Akaike Information Criterion (AIC) model selection using Akaike Weights (Wagenmakers & Farrell 2004), although other model-selection statistics can be substituted such as the Bayesian Information Criterion (BIC) depending on specific research objectives (Burnham & Anderson 2007).

1.5 Interchangeable Components

Bristlecone was designed as a generic workflow, where custom components that match the type signatures in *Bristlecone* can be readily substituted. These include integration and optimisation methods, graphics and real-time logging, and end conditions. We have provided built-in components that have worked well for our research in long-term ecology, but others may substitute their own.

1.5.1 Optimisation

An optimisation technique is responsible for minimising the negative log likelihood to identify the maximum likelihood. The procedure ends in conformance with an **EndCondition**: a function depending on the procedure's history that returns true if the condition is met. Any optimisation technique can be used within a *Bristlecone* analysis, so long as it matches the appropriate type signature:

```
type Optimise<'d> = WriteOut -> EndCondition<'d> -> Domain -> Objective ->
Solution<'d> list

and WriteOut = LogEvent -> unit
and EndCondition<'d> = Solution<'d> list -> bool
```

The most appropriate optimisation technique depends on the nature of the problem with extensive reviews available elsewhere (Reali et al. 2017; Rios & Sahinidis 2012). We focus on stochastic gradient descent algorithms that may be applied to any simple or complex function to minimise. Algorithms included in *Bristlecone* are shown in **Table 1**:

Table 1 Summary of included optimisation techniques with reference implementation

Family	Technique	Description	Reference
Amoeba	Single	A single Nelder-Mead simplex	(Press et al. 1992)
	Swarm	Many layers of n Nelder-Mead simplexes	None
Monte Carlo	Classical Simulated Annealing	Proposal jumps randomly drawn from normal distribution. Exponential cooling.	(Corana et al. 1987; Tsallis & Stariolo 1996; Sharman 1988)
	Fast Simulated Annealing	Proposal jumps randomly drawn from Cauchy distribution with a faster cooling schedule than classical simulated annealing.	(Szu & Hartley 1987; Lee 2015)
	Random Walk	Proposal jumps randomly drawn from multivariate normal distribution. <i>Suitable as a final MCMC algorithm.</i>	(Reali et al. 2017)
	Adaptive Metropolis	Weights covariance structure of proposal distribution towards recent chain history. Tunes scale of distribution based on acceptance rate.	(Haario et al. 2001)
	Generalised MCMC	A metaheuristic that conducts rough and fine tuning, before running a 'clean' random walk MCMC.	(Yang & Rosenthal 2016)
	Filzbach	An implementation of the 'Filzbach' algorithm from Microsoft Research.	(Microsoft Corporation 2011)

Metropolis-within-Gibbs	Proposes jumps for each parameter sequentially.	(Yang & Rosenthal 2016)
Adaptive Metropolis-within-Gibbs	Tunes the scale of each parameter separately, based on acceptance rates.	(Yang & Rosenthal 2016)

1.5.1.1 Amoeba Algorithms

Amoeba methods provide a local search. The Nelder-Mead simplex expands and contracts an n -dimensional geometry within an unconstrained parameter space, following a ‘downhill’ gradient (Press et al. 1992). Our implementation draws a random point from a normal distribution, informed by the starting bounds for each parameter. Although simple and relatively fast, as a local search it may be inappropriate for more complex likelihood surfaces that have more than one local minima. To address concerns of local minima, we included a swarm mode. The swarm mode creates an ensemble of 20 amoeba, which all crawl the likelihood surface. After n iterations, we discard all amoeba that have values above the 80th percentile of the negative log likelihood values returned. We run this procedure for five levels but end early if all values returned are identical.

1.5.1.2 Monte Carlo Algorithms

Monte Carlo Markov Chain (MCMC) methods have two key advantages over amoeba solvers: first, given infinite time, a homogenous MCMC chain is guaranteed to identify the lowest likelihood region of parameter space (Chib & Greenberg 2012). Second, the sampled posterior distribution provides an indication of uncertainty for each parameter without the need for additional calculations. Of the included methods, simulated annealing methods are designed specifically for minimisation whereas other methods are distribution-sampling methods and thus relatively inefficient for identifying the minimum.

Monte Carlo approaches may be combined using tuning and sampling periods to tackle the trade-offs of convergence guarantees, generalisability, and speed of convergence. Random walk MCMC is a Metropolis-Hastings algorithm that draws from a Gaussian proposal distribution generating a homogenous MCMC chain, but its efficiency is reliant on the starting parameter bounds (which form the multivariate Gaussian prior distribution).

Simulated annealing (SA) models a minimisation problem in terms of particle entropy, where the function value is analogous to energy (Du & Swamy 2016). At high temperatures, particles are in a high-energy state, thus can move readily. As temperature gradually decreases, particles are less able to move to high energy states, with particles eventually arranging into the ‘ground state’ of a solid material. When Temperature (T) = 1, the simulated annealing algorithm is identical to a Metropolis-Hastings MCMC algorithm. We have provided two of the most common SA algorithms for continuous systems: Gaussian (GSA) and Cauchy (CSA). In GSA, candidates are drawn from a temperature-dependent normal distribution, with an exponential cooling schedule (Corana et al. 1987). In CSA, a semi-local search is enabled by drawing candidates from a Cauchy distribution, which enables a faster cooling

schedule. In these schemes, the distributions ‘tighten’ as the temperature is lowered, allowing the search to become more focused on the identified minimum (Szu & Hartley 1987). Our implementation roughly tunes the temperature-dependent scale (Cauchy) or standard deviation (Gaussian) for each individual parameter using a Metropolis-within-Gibbs function to improve sampling efficiency. The acceptance rate for each parameter is tuned to between 0.2 and 0.4.

1.5.2 Real-Time Diagnostics

Bristlecone includes tools for monitoring and diagnosing the model-fitting process. Each **EstimationEngine** has a connector for a **WriteOut** function, which consumes **LogEvents**. *Bristlecone* includes threading-aware console logging by default, and this may be extended or replaced. For instance, we have implemented a multi-chain parallel and real-time graph of each parameter trace within the fitting process. This is implemented through interoperability with the R programming language: plots are rendered by passing real-time data to the *ggplot* R package (**Figure 2**).

Figure 2 Real-time traces of parameter estimates output by the *Bristlecone* library. These charting capabilities demonstrate *Bristlecone* interoperability with the R programming language using the F# R type provider. Here we use the *ggplot2* R package to generate graphics.

1.6 Discussion

We have described a library called *Bristlecone* that may be used to understand and interrogate ecological mechanism from long-term ecological datasets. It can, however, be used within other fields such as

econometrics if desired. The library yields information on the most appropriate model structure and parameters with uncertainty estimates to explain observed data.

We sought to create a straightforward and reproducible workflow to define and conduct the interrogation of ecological time-series using competing theoretical models. Our objective was not to design the most efficient and generalisable global optimisation algorithm. Although other libraries such as Stan (Gelman et al. 2015) may be used to conduct interrogation of ecological time-series with mathematical models, *Bristlecone* handles all considerations to conduct a full model-fitting and model-selection analysis including domain-specific requirements for long-term ecology: handling of time-series data; and complex non-linear models written as ordinary differential equations.

For long-term ecological research we provide tools specifically geared towards interrogating tree ring and sediment core data. Similar top-down modelling approaches exist such as the VS-lite model, which incorporates climate-limitations to wood growth using the principle of limiting factors (Tolwinski-Ward et al. 2010). Various models have included allometric relations (Pommerening & Muszta 2016) and day length (Tolwinski-Ward et al. 2013) into tree ring modelling approaches. We wish to encourage the further interrogation of tree ring datasets using top-down modelling approaches with *Bristlecone*.

1.7 Funding

This work was completed with funding from the Natural Environment Research Council of the United Kingdom (grant number NE/L002612/1).

1.8 References

- Archontoulis, S.V. & Miguez, F.E., 2015. Nonlinear Regression Models and Applications in Agricultural Research. *Agronomy Journal*, 107(2), pp.786–798.
- Binney, H. et al., 2017. Vegetation of Eurasia from the last glacial maximum to present: Key biogeographic patterns. *Quaternary Science Reviews*, 157, pp.80–97.
- Birks, J. et al., 2014. Quantitative Palaeoenvironmental Reconstructions from Holocene Biological Data. In *Global Change in the Holocene*. Routledge, pp. 123–139.
- Blaauw, M., Christen, J.A. & Aquino-López, M.A., 2019. A Review of Statistics in Palaeoenvironmental Research. *Journal of Agricultural, Biological and Environmental Statistics*, pp.1–15.
- Bolker, B., 2007. *Deterministic Functions for ecological modeling*,
- Bonsall, M.B. & Hastings, A., 2004. Demographic and environmental stochasticity in predator-prey metapopulation dynamics. *Journal of Animal Ecology*, 73(6), pp.1043–1055.
- Burnham, K.P. & Anderson, D.R., 2007. *Model Selection and Multimodel Inference*, Springer Science & Business Media.
- Chib, S. & Greenberg, E., 2012. Understanding the Metropolis-Hastings Algorithm. *The American Statistician*, 49(4), pp.327–335.
- Corana, A. et al., 1987. Minimizing multimodal functions of continuous variables with the “simulated annealing” algorithm—Corrigenda for this article is available here. *ACM Transactions on Mathematical Software (TOMS)*, 13(3), pp.262–280.

- Du, K.-L. & Swamy, M.N.S., 2016. Simulated Annealing. In *Search and Optimization by Metaheuristics*. Cham: Springer International Publishing, pp. 29–36.
- Fritts, H.C., 1976. *Tree Rings and Climate*, Academic Press.
- Gelman, A., Lee, D. & Guo, J., 2015. Stan: A Probabilistic Programming Language for Bayesian Inference and Optimization. *Journal of Educational and Behavioral Statistics*, 40(5), pp.530–543.
- Haario, H., Saksman, E. & Tamminen, J., 2001. An adaptive Metropolis algorithm. *Bernoulli*, 7(2), pp.223–242.
- Hilborn, R. & Mangel, M., 1997. *The Ecological Detective*, Princeton University Press.
- Jefferies, E.S. et al., 2011. Climate change impacts on ecosystem functioning: evidence from an Empetrum heathland. *New Phytologist*, 193(1), pp.150–164.
- Johnson, E.A. & Miyanishi, K., 2008. Testing the assumptions of chronosequences in succession. *Ecol Letters*, 11(5), pp.419–431.
- Lee, C.-Y., 2015. Fast simulated annealing with a multivariate Cauchy distribution and the configuration's initial temperature. *Journal of the Korean Physical Society*, 66(10), pp.1457–1466.
- Martin, A.C. et al., **Nitrogen limitation of Arctic shrub growth over three decades: inference from mechanistic modelling of dendroecological data, Yamal, Russia.**
- Meeus, J., 1998. *Astronomical Algorithms*,
- Microsoft Corporation, 2011. *Filzbach User Guide*.
- Paine, C.E.T. et al., 2011. How to fit nonlinear plant growth models and calculate growth rates: an update for ecologists. *Methods in Ecology and Evolution*, 3(2), pp.245–256.
- Peng, C. et al., 2011. Integrating models with data in ecology and palaeoecology: advances towards a model-data fusion approach. *Ecology Letters*, 14(5), pp.522–536.
- Petricek, T., Guerra, G. & Syme, D., 2016. Types from data: making structured data first-class citizens in F#. In the 37th ACM SIGPLAN Conference. New York, New York, USA: ACM Press, pp. 477–490.
- Pommerening, A. & Muszta, A., 2016. Relative plant growth revisited: Towards a mathematical standardisation of separate approaches. *Ecological Modelling*, 320, pp.383–392.
- Poorter, H., Anten, N.P.R. & Marcelis, L.F.M., 2013. Physiological mechanisms in plant growth models: do we need a supra-cellular systems biology approach? *Plant, Cell & Environment*, 36(9), pp.1673–1690.
- Press, W.H. et al., 1992. *Numerical Recipes in C*, Cambridge University Press.
- Real, F., Priami, C. & Marchetti, L., 2017. Optimization Algorithms for Computational Systems Biology. *Frontiers in Applied Mathematics and Statistics*, 3, p.80.
- Rios, L.M. & Sahinidis, N.V., 2012. Derivative-free optimization: a review of algorithms and comparison of software implementations. *Journal of Global Optimization*, 56(3), pp.1247–1293.
- Sharman, K.C., 1988. Maximum likelihood parameter estimation by simulated annealing. In ICASSP-88., International Conference on Acoustics, Speech, and Signal Processing. IEEE, pp. 2741–2744.
- Szu, H. & Hartley, R., 1987. Fast simulated annealing. *Physics Letters A*, 122(3-4), pp.157–162.
- Tolwinski-Ward, S.E. et al., 2010. An efficient forward model of the climate controls on interannual variation in tree-ring width. *Climate Dynamics*, 36(11-12), pp.2419–2439.
- Tolwinski-Ward, S.E., Anchukaitis, K.J. & Evans, M.N., 2013. Bayesian parameter estimation and interpretation for an intermediate model of tree-ring width. *Climate of the Past*, 9(4), pp.1481–1493.
- Tsallis, C. & Stariolo, D.A., 1996. Generalized simulated annealing. *Physics A*.
- Wagenmakers, E.-J. & Farrell, S., 2004. AIC model selection using Akaike weights. *Psychonomic Bulletin & Review*, 11(1), pp.192–196.
- Willis, K.J. et al., 2010. Biodiversity baselines, thresholds and resilience: testing predictions and assumptions using palaeoecological data. *Trends in Ecology & Evolution*, 25(10), pp.583–591.
- Yang, J. & Rosenthal, J.S., 2016. Automatically tuned general-purpose MCMC via new adaptive diagnostics. *Computational Statistics*, 32(1), pp.315–348.